Graphite: Graph-Induced feaTure Extraction for Point Cloud Registration

Mahdi Saleh[1], Shervin Dehghani[1], Benjamin Busam[1], Nassir Navab[1], Federico Tombari[1,2]
[1] Technische Universität München, [2] Google
{m.saleh, shervin.dehghani, b.busam, nassir.navab}@tum.de, tombari@in.tum.de

Abstract

3D Point clouds are a rich source of information that enjoy growing popularity in the vision community. However, due to the sparsity of their representation, learning models based on large point clouds is still a challenge. In this work, we introduce Graphite, a GRAPH-Induced feaTure Extraction pipeline, a simple yet powerful feature transform and keypoint detector. Graphite enables intensive downsampling of point clouds with keypoint detection accompanied by a descriptor. We construct a generic graph-based learning scheme to describe point cloud regions and extract salient points. To this end, we take advantage of 6D pose information and metric learning to learn robust descriptions and keypoints across different scans. We Reformulate the 3D keypoint pipeline with graph neural networks which allow efficient processing of the point set while boosting its descriptive power which ultimately results in more accurate 3D registrations. We demonstrate our lightweight descriptor on common 3D descriptor matching and point cloud registration benchmarks [76, 71] and achieve comparable results with the state of the art. Describing 100 patches of a point cloud and detecting their keypoints takes only 0.018 seconds with our proposed network.

1. Introduction

Point clouds play an indispensable role in modern 3D computer vision applications. LiDAR-equipped cars sense distances as sparse point clouds and mobile robots as well as modern hand-held devices measure depth with RGB-D and Time-of-Flight hardware. These devices essentially see the world in point cloud form. Thus, efficient processing of point clouds is an integral part to equip agents with 3D perception. At the same time, modern sensors are capable to provide large quantities of point cloud data at high frame rates. However, processing high numbers of sparse and unordered points can be computationally demanding. While trivial sub-sampling methods can lead to increased sparsity and loss of fine local information, perceptive sub-sampling of point clouds is very much structure dependant.

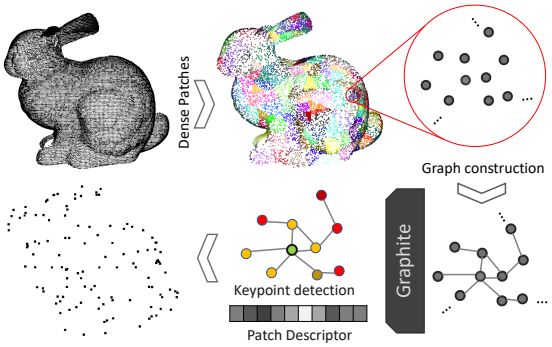


Figure 1. Graphite helps representing a source point cloud (top-left) densely by describing each patch with a small descriptor and detecting a keypoint per patch (down-left). Fast calculation with lightweight model enables real-time point cloud applications.

Keypoints and Feature Description. To distill the essential information in such rich vision data, classical methods represent image and scenes regions using feature descriptors. Extracted keypoints indicate salient landmarks and descriptors encapsulate local information in different formats. For example for the application of 6D pose estimation, a generic descriptor is oftentimes preferred over learning each object specifically with a deep neural network. In the image domain, handcrafted features [36, 6, 1, 32] and learned descriptors [75, 67] are employed to match regions or objects under certain levels of 3D rotation change and various illuminations. Recently with the popularity of self-supervised learning, it is possible to improve the quality of image-based features across multiple frames when homography or correspondence information is available. This has shown to improve matching rate when dense keypoints are exploited [17, 15]. In the 3D domain and specifically on point clouds, descriptors are used for surface registration, multi-view reconstruction and 3D object pose estimation. Registration of 3D points clouds normally involves three steps: 1) find a set of salient points 2) describe each salient point with a descriptor 3) discover correspondences through descriptor matching.

Neural Point Processing. Point clouds are by nature unordered sets and processing models thus require per-

mutation invariance. PointNet [48] pioneered the field by suggesting a deep network that could segment or classify objects from their point clouds. In their work, building blocks such as T-Nets are utilized to force transformation invariance while permutation invariance is gained via max-pooling. Although PointNet and its successor PointNet++ [49] are widely used, their design choice limits the inclusion of local neighborhoods and geometrical embeddings. Later works such as [33, 69] suggests the use of graphs to introduce point connectivity and vicinity information. Although they manage to learn local geometries by connecting points and learning node level features, they lack notion of metric scale through construction of graphs using k-NNs. After every step in processing node-level features, they change edge links and attributes. The inherent edge permutation of this process inhibits a consistent information flow. To this end, we propose a novel graph-based model capable of retaining geometrical shape under varying scale and sampling conditions. To describe point clouds, graphs are well suited as to capture relative information with their connections and can be designed to be invariant to transformations.

The Graphite Model. A classical approach for point cloud registration is the two stage paradigm of initial sampling of points to find keypoints and a subsequent feature description [56, 25]. A keypoint is matched with another by finding the closest point in feature space. If insufficient salient keypoints are matched or mismatches occur, the registration fails. Besides sub-sampling a point cloud to a set of interest points, we propose to describe point clouds on dense patches. However forcing keypoints per patch might not be ideal for some tasks such as registration. Therefore we define soft keypoint saliency through a scoring mechanism. We first form patches from a point cloud uniformly and then convert every patch to a graph structure similar to the connected structure of carbon atoms in the crystalline form of graphite. These graphs are learned to represent Graphite features which can be used for matching. We train our model in two stages. Firstly, we initialize our network to learn keypoints by supervision. In the second stage, we let our network optimize the descriptor and detector by performing metric learning on object point clouds obtained from various 6D poses.

In short, our contribution is two-fold: Firstly, we provide a lightweight model that can detect salient keypoints in point cloud patches and regions taking advantage of a novel graph processing architecture. Secondly, the same model is used to describe the patch with a feature descriptor which can be used for correspondence matching. Our patch-based representation and reduction model is a major advantage compared to dense descriptors per point approach. We evaluate our descriptor on 3DMatch [76], a commonly used benchmark for descriptor matching. Furthermore, we evaluate our dense descriptor and the extracted keypoints for the applications of point cloud registration on ModelNet Object Registration [71].

2. Related Work

This section reviews related work in the area of feature extraction in 2D to 3D from both handcrafted and learned approaches, and discusses state of the art for point cloud learning and registration.

2.1. 2D Feature Extraction & Matching

Image descriptors are functions that map a local region of an image to a feature vector. In order to determine the interest region worth to describe, keypoint detectors are used to find salient areas. The first keypoint extractors [43, 26] rely on image gradients and self-similarity metrics to detect corner like structures. To robustly detect corner points, consecutive methods make use of template-based techniques [62, 52, 53, 38] using machine learning and binary classifiers. The concept of consecutively smoother images in scale space [34, 35] helps to find blob-structured keypoints across various image scales with the Laplacian of Gaussian (LoG) operator and its scale-normalized counterpart. The approximation of its calculation with a differences of Gaussians improves runtime performance in the prominent SIFT [36] pipeline. Further methods [6, 32, 1, 31] focus on advances in repeatability, accuracy, robustness and computational efficiency. These are well studied in extensive comparison and survey papers [60, 66, 39, 40, 30, 58, 4] that investigate differences and performance of feature extraction pipelines. With the advent of deep learning, TILDE [67] tackles the problem of illumination changes and MagicPoint [16] explores the advantages of training with synthetic primitive data while Key.Net [5] combines handcrafted and learnt features.

Some methods combine keypoint extraction directly with a feature description stage. SIFT, for instance, uses a 128-dimensional vector to describe its feature points and a ratio test is proposed to withdraw ambiguous matches. Further descriptors [12, 32, 54, 1] propose binary features to make use of the fact that the matching can be done efficiently via Hamming distance calculation. This allows their use in real-time systems such as 6D pose estimation [11] and SLAM [45] pipelines. Metric learning is a prominent way to leverage data for image description. HARDNet [41] proposes a triplet margin loss with hard negative mining for this task and the advantages of descriptor learning over hand-crafted methods have been shown with L2Net [64]. R2D2 [51] proposes a dense version of it while estimating also repeatability and reliability.

A joint estimation of detection, orientation and description is done in LIFT [75] and LF-Net [46] also integrates

scale space in an unsupervised learning framework driven by structure from motion. Similar to this, SuperPoint [17] uses homography warping as a self-supervision signal to jointly estimate saliency map and a dense descriptor. More recently, D2Net [19] experiments with a single feature map for both detection and description resulting in more robust but less accurate results. The matching stage classically involves a nearest neighbour search [44, 7, 61, 22] accompanied by outlier removal through ratio test, cross check or RANSAC [21] and more elaborate techniques involve motion statistics [8] or temporal constraints [55]. Recently, SuperGlue [59] proposes to use a graph neural network to solve the assignment as an optimal transport problem.

2.2. 3D Descriptors

3D descriptors are relatively less evolved. One reason is varied data representations and the complexity of describing point clouds. In scenarios where RGBD images are available, depth is used as an auxiliary information to find features or templates for matching [27, 28, 70]. Classical point cloud and surface descriptors such as SHOT [65], RoPS [24] and TriSi [25] use a unique local reference frame to explain geometry with rotation invariance. PFH [57] and FPFH [56] use pair-wise point features and surface normals describe curvature. While large scene variability can harm their performance, their classical nature allows for the use on edge devices with hardware constraints. In recent years, scholars have also designed 3D descriptors with deep learning methods. CGF [29] uses supervised learning to map hand-crafted high dimensional features into a lower dimensional vector. PPf-FoldNet [13] combines PPF-Net [14], PointNet [48] and FoldingNet[73] to learn rotation invariant features with self-supervision. 3DFeat-Net [74] also learns to extract sparse features with weak supervision from the tagged geolocation data. In parallel to this also dense voxel based approaches are explored. 3DMatch [76] converts point clouds to truncated distance functions (TDF) and Perfect Match [23] uses voxelized smoothed density value (SDV) to describe local reference frames. Although they achieve substantial results on the 3DMatch Benchmark [76] they do not learn directly from point clouds.

2.3. Deep Point Cloud Processing

While point clouds are used widely in computer vision, learning them with deep models have been far more challenging than 2D images or voxels. The pioneering pipeline PointNet [48] manages to build an architecture which is permutation-invariant and can learn point cloud features. Following that, PointNet++ [49] proposes hierarchical learning to learn larger scale sets. While the contribution of PointNet is without doubt significant, it concentrates on global features. As a result, it fails to employ local features and information on the geometric neighborhood. To this end, also 3D Capsules [77] are used for this task where and auto-encoders and capsule networks learn point features. In an effort to simulate convolutions, researchers incorporated graphs to operate on point clouds [42, 9]. Graphs help connecting points and build structures which can potentially represent surfaces and manifolds. Graph based point cloud approaches such as DGCNN [69] and PointCNN [33] yield better results on segmentation and classification of point clouds. While graphs structures bring rotation invariance (isomorphism), how to define edges without losing metric information and attributing nodes and edges to sustain geometries are still not fully investigated.

2.4. Point Cloud Registration

Registering point clouds is a classic problem with applications such as scene reconstruction or object pose estimation. Conventional iterative methods such as ICP [3] are still commonly used although they are very dependent on initialization. Go-ICP [72] improves ICP by accuracy but at a high computational cost. Soft assignment [50] is another iterative approach which improves initialization by exploring and soft assigning correspondences to estimate 6D poses [10]. Other scholars use conventional 3D descriptors to register clouds [63, 56, 72]. Following the success of PointNet, PointNetLK [2] builds an iterative learning approach on top of PointNet and Lucas & Kanade (LK) algorithm [37] to register two point clouds. Deep Closest Point (DCP) [68] suggests using an attention-based module to find correspondences and a differentiable singular value decomposition to estimate transformation.

3. GRAPHITE

In this section we explain the methodology and pipeline of Graphite. First we discuss how to segment point clouds into patches and to convert them into graphs. We then introduce our graph learning architecture and the training stage. Finally we explain how we perform point cloud warping.

3.1. Point Patches

The input to our pipeline is a point cloud, that can come directly from a sensor such as a LiDAR, can be back-projected from an RGB-D frame or can be sampled from a 3D mesh. A point cloud P with m points is an unordered set of points $\{p_1, p_2, p_3, ..., p_m\}$, with each p_j consisting of coordinates and normal/color information $p_j = (x_j, y_j, z_j)$. We want to break our point cloud P into small patches C_i each holding n points. $C_i = \{p_1, p_2, p_3, ..., p_n\}$ with $\bigcup C_i = P$. In order to find clusters we perform random sampling and build clusters of n points around each centroid. If a projected depth frame is available, this can be a rectangular patch of size $w \times w = m$ in the image domain, which represents a small frustum in the point cloud. Now

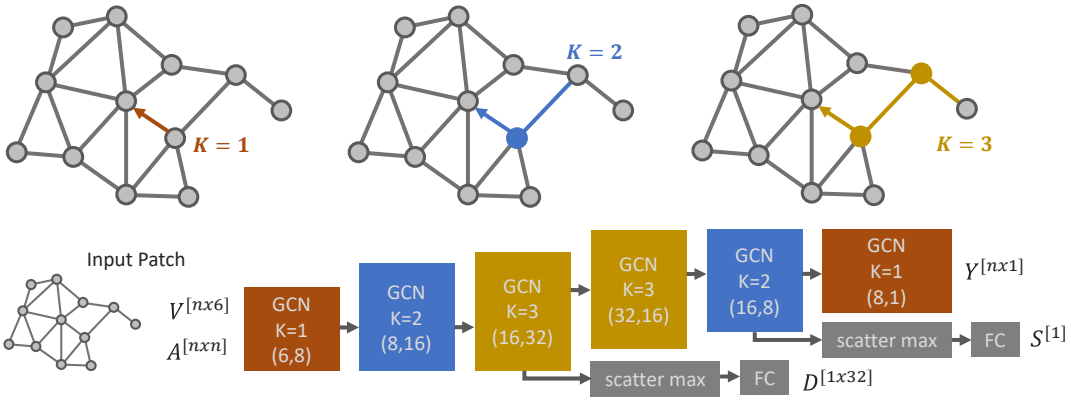


Figure 2. Top: The schematic of node level feature propagation given the edge connectivites of graph and different hops values K=(1,2,3). Bottom: The Graphite architecture, consisting of GCN layers with different hops (increasing and then decreasing) to estimate a patch descriptor (D) from the middle stage and keypoint localization (Y) and scoring (S) at the end.

suppose we have a function $F(C_i) = Y_i, F : \mathbb{R}^{n\times 3} \rightarrow \mathbb{R}^n$ that, for each point p in patch C_i, finds a value y representing the saliency of that point. The point with maximum value in each patch would be our point of interest or keypoint k_i:

$$k_i = \underset{p}{\text{argmax}}(F(C_i)) = \underset{p}{\text{argmax}}(Y_i) \tag{1}$$

Every patch is thus associated with a salient point (keypoint). Furthermore, we transform every patch to a descriptor vector D_i of size l. A function $G(C_i) = D_i, G : \mathbb{R}^{n\times 3} \rightarrow \mathbb{R}^l$ maps a patch to the descriptor. Using the functions F and G, one can transform every patch to $k_i \in \mathbb{R}^3$ and $D_i \in \mathbb{R}^l$ respectively.

3.2. Graph Definition

Given a set of points $C_i = \{p_1, p_2, p_3, ..., p_n\}$ in a patch as described in the previous section, we want to create a graph $X_i = (V_i, E_i)$. Graphs are constructed from a set of vertices V_i and edges E_i. As a common practice, we assume that every point in the patch C_i can be considered as a node in the graph X_i. A node is represented by its coordinates and normal vectors in local reference frame (x, y, z, a, b, c). Thus $V_i \in \mathbb{R}^{n\times 6}$.

In addition to nodes, edges would demonstrate the connectivity of nodes and hold geometrical embedding. An edge $e_{j,k}$ connects vertices v_j and v_k. In previous works such as [69], edge connectivities are associated with K-nearest neighbors. The k-NN constraint would force a fixed number of edges into every node ultimately losing metric information. In contrast to this, we consider the unit ball with radius r around each node's positional coordinates, and connect them to other nodes when their distance falls bellow r. We calculate the parameter r based on average point cloud resolution. In addition to this, we attribute a weight to every edge to give a higher weight to close neighbors (and vice-versa) as follows:

$$e(j,k) = \begin{cases} \frac{r}{r+||p_j - p_k||}, & \text{if } ||p_j - p_k|| < r. \\ 0, & \text{otherwise.} \end{cases} \tag{2}$$

Taking the function $F(C_i)$ from Section 3.1, we can input a graph to our model $F(C_i) = F'(X_i) = Y_i,\ F' : \mathbb{R}^{n\times 6} \rightarrow \mathbb{R}^n$ and predict a value per point/node of the patch. We can define this value as the inverse of the shortest path to the desired keypoint k_i. $y_{k_i} = max(F'(X_i))$. During supervised training explained in Section 3.7, we assign the distance to a known keypoint to the node values.

3.3. GCN Architecture

In this section we introduce our graph convolutional network (GCN) architecture as depicted in Figure 2. As mentioned previously we build graphs consisting of edges and nodes to represent a point cloud. We associate node features to estimate the value function $F(X)$ which is used for keypoint detection discussed in Section 3.2. Moreover, we construct a function $G(X_i)$ that estimates descriptors from the graph, which should be invariant to node orders. We therefore use the scatter max operation which is a symmetric function for our descriptor. We build upon Topology Adaptive Graph Convolutions [18] which will give us a framework to digest node and edge level information under different topologies.

We simulate the notion of multi-scale processing in classical feature descriptor and modern 2D convolutional networks by stacking GCN layers with different hops as illustrated in Figure 2. Hops define how many nodes the information passes by on its way. By letting increasing and then decreasing hops (K=1,2,3) in our filters we increase the receptive field of nodes to capture patch-wide features and therefore manage to describe a global representation of the graph. Moreover, in contrast to previous

works that uses two heads for F and G [5, 17], we model both $F, G(X_i) = (Y_i, D_i)$ with the very same network. An illustration of hops and the Graphite model can be found in Figure 2. In the next section, we describe the design and process steps of our three concepts for description, keypoint localization and scoring.

3.4. Descriptor

Starting with the input patch graph X we apply 3 layers of graph convolutions at first. Every graph convolution module is taking into account the adjacency matrix $A \in \mathbb{R}^{n \times n}$ built using Eq. 2 and its diagonal degree matrix $D' \in \mathbb{R}^{n \times n}$ to propagate node features across the graph. As in [18], we update node level information X_i' by propagating features using edge level information as follows:

$$X_i{}' = \sum_{k=0}^{K} \mathbf{D'}^{-1/2} \mathbf{A}^k \mathbf{D'}^{-1/2} \mathbf{X_i} \mathbf{\Theta}_k, \tag{3}$$

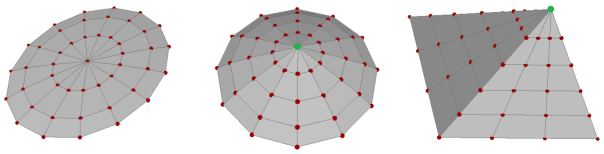
where Θ_k is the matrix to be learned for hop iteration k. After each layer we increase the depth of features from 6 to 8, 8 to 16 and 16 to 32. After three layers, the updated node features encapsulate features from the surrounding points and therefore describe the considered patch. To force descriptor invariance with respect to both point and node permutation, a scatter max aggregation function is utilized similar to PointNet [48]. The features follows linear fully connected layers and a normalization layer to project the feature on a unit sphere. The final descriptor vector is 32-dimensional ($l = 32$).

3.5. Keypoint Localization

Apart from descriptor we need to formulate a value function $G(X_i) = Y_i$ to estimate how salient a point is in comparison with its neighboring nodes. In the context of this work we assume every small patch has at most one salient keypoint. Following the first three GCN layers leading to descriptor branch, we now squeeze back features to estimate per node values (Y_i). Feature depths are decreased from 32 to 16, 16 to 8 and 8 to 1 while the graph formation and node values are constrained.

3.6. Keypoint Scoring

Our target is to improve point cloud registration. However, not every single patch possesses enough structure to represent a salient keypoint. For this reason, we formulate a soft invalidation of our non-salient keypoints through assignment of a global score S per patch. Figure 3 depicts different level of keypoint scoring. We can use mesh surface information from synthetic data to assign scores given their curvature magnitude. To regress this scalar value we perform another scatter max on the output feature of layer five and apply a FC unit on top of that.



$S \approx 0$ $S = 0.3$ $S = 1$

Figure 3. We render point clouds (in red) from different surfaces of 3D primitives. We assign a score measure S to each patch given its taper curvature and height and a keypoint K in green. $S = 1$ infers a surface with potential salient keypoint . $S \approx 0$ infers a non-salient flat patch

3.7. Network Initialization

In order to properly learn our joint model we propose to have a training initialization. Supervising the network in the first stage helps the model to detect corners. As the pointiness of a shape corner is a fundamental geometric property, we let our network first focus on these points. Furthermore, we drive our GCN to predict surface flatness and curvature to infer keypoint scoring.

To fulfill this goal, we create a synthetic dataset, including point clouds of rendered depth maps with known shape and surface. Random primitive corners are generated with different curvature and known corner points. We also render the same 3d primitive (such as cone, box or pyramid) from two different camera views to simulate pose and sampling variations. Keypoint locations are labeled on the point sampled closest to the corner. We then measure ground truth values for all the points in the point cloud given their shortest path to the target keypoint in the constructed graph. Values and scores are learned in a supervised way with MSE loss (Eq. 4) to give the network an early assumption of such salient extreme points. This will be fine-tuned without direct supervision in the following stage (see Section 3.8).

$$\mathcal{L}_V = (Y - \hat{Y})^2, \; \mathcal{L}_S = (S - \hat{S})^2 \tag{4}$$

Alongside keypoint detection, we also predict a descriptor which we train using metric learning with a triplet margin loss. Each of the triplet samples, *i.e.* reference, positive and negative, will have a predicted descriptor (D_r, D_p and D_n). A similarity term $|D_r - D_p|$ pulls together descriptors while a push term $|D_r - D_n|$ increases the distance between non-matching descriptors. The loss is as follows:

$$\mathcal{L}_D = \frac{|D_r - D_p|}{|D_r - D_p| + m \cdot |D_r - D_n|} \tag{5}$$

With this we force the point clouds taken from the same patch but with different samplings to retrieve the same descriptor while descriptors from different patches are pushed apart. The total loss will accumulate all three losse terms $L_T = L_D + L_V + L_V$.

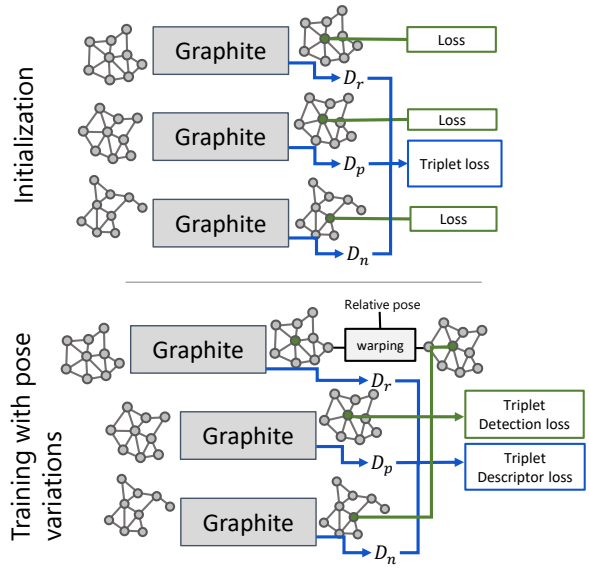


Figure 4. Training happens in two stages. In the initialization stage (top), triplet inputs are fed into networks with shared weights. The detectors are learned with supervised loss and the descriptors are learned through triplet loss. (Bottom) we train using 6D pose with minimum supervision. Both detector and descriptor are trained with triplets. In order to measure similarities on point cloud values we warp one point cloud to another using known relative pose.

3.8. Training with pose variations

After initializing our model on homemade synthetic primitives dataset, we subsequently fine-tune our feature detector with the support of pose annotations on a more variety of geometries and samplings. Descriptors should be rotational and sampling invariant. If we transform and resample a point cloud with a 6D pose, it should not produce a different descriptor or find a different keypoint. Moreover the salient regions should be consistent under different patch samplings to improve matching. This is particularly significant in real applications e.g. when capturing range data, sampling a surface would produce different point clouds with different resolution and noise. To build a robust descriptor and detector, we use the Graphite model to describe the same surface with varied samplings and poses. Similarly, we remove the direct supervision of value estimation to let the network detect optimal and persistent keypoints from different views using metric learning and triplet loss similar to Eq. 5.

3.9. Point cloud warping

Given two point clouds from two different viewpoints with known relative pose, we first tessellate each cloud into small sets of points (patches). In presence of variable sampling rate or noise perturbations transformed points would not lay exactly on corresponding points but rather somewhere on the surface. To facilitate metric learning across different poses we need to warp each patch into one another. With warping, each embedding from the original point set finds its equivalent embedding in the other set to compare.

Lets assume we have a patch C_i from cloud P and we know pose T_{pq} to transform point cloud P to Q. Applying known transformation T to P will move us to P', $T_{pq}P = P'$. We find the corresponding patch of C_i, C_j in Q using nearest neighbors of c' in P' with

$$C_j = \{c \in Q : \exists c' \in P', c \in knn(Q, c')\}. \quad (6)$$

Where $knn(Q, c')$ find k nearest neibors of c' in Q. We finally utilize k-NN again to find the weight combinations for corresponding value assignments. The weights are defined as inverse 3d distance in the k nearest neighbors for value warping.

4. Experiments and Evaluations

4.1. Implementation details

Our implementation is done with PyTorch Geometric [20] and PyTorch [47] for graph definition and graph convolutions. To process, sample and operate with point clouds, we leverage the Open3D library [78]. The evaluation follows the example implementation provided by [23] and [68]. We train and evaluate all pipelines on an Intel Core i7-8700K CPU 3.70GHz × 12 and an Nvidia GeForce RTX 2080 Ti GPU. On this hardware average Graphite calculation for 100 patches takes 0.018 seconds on GPU.

4.2. Synthetic Primitive Corners

As introduced in Section 3.7 and inspired by the 2D processing of MagicPoint [16], we start with synthetic data training with the aim to guide the network in order to learn different shapes and primitive differential geometric concepts to locate corners as keypoints. We create a rendering pipeline with a pair of cameras at random poses pointing towards a shape corner from the object. The corner of focus can associate 3-10 faces with varying heights and curvatures. Every instance is then rendered from the camera pairs to simulate different point sampling as it would occur in a natural scene. The depth maps are then back-projected to a point cloud given the known camera intrinsics. In total, we produce 20k random patch pairs. Each pair is grouped with a random instance which together form a triplet. A random patch of size n is sampled in the vicinity of the corner. Every patch is then converted as explained in Section 3.2 with a fixed radius r, and the nodes are annotated with a value Y inverse to the length of their path to a target keypoint node. The target node is valued with 1. In the supplementary material you can find sample triplets from this dataset .

Method		MSE	RMSE	MAE
ICP	R	892.60	29.88	23.63
	t	8.60	2.93	2.52
Go-ICP	R	192.26	13.87	2.91
	t	0.05	0.22	0.06
FGR	R	97.00	9.85	1.45
	t	0.02	0.14	0.02
PointNetLK	R	306.32	17.50	5.28
	t	0.08	0.28	0.07
DCP-v1	R	19.20	4.38	2.68
	t	**<0.01**	0.05	0.04
DCP-v2	R	9.92	3.15	2.01
	t	**<0.01**	**0.05**	**0.03**
Ours	R	7.44	2.73	1.49
	t	0.31	0.56	0.38
Ours + ICP	R	**0.75**	**0.86**	**0.11**
	t	0.09	0.30	0.07

Table 1. Point cloud registration comparison on ModelNet objects from unseen categories.

Method		MSE	RMSE	MAE
ICP	R	882.56	29.71	23.56
	t	8.45	2.91	2.49
Go-ICP	R	131.18	11.45	2.53
	t	0.05	0.23	0.042
FGR	R	607.69	24.65	10.06
	t	1.19	1.09	0.27
PointNetLK	R	256.16	16.00	4.60
	t	0.047	0.216	0.057
DCP-v1	R	6.93	2.63	1.52
	t	**<0.01**	**0.02**	0.02
DCP-v2	R	**1.17**	**1.08**	0.74
	t	**<0.01**	**0.02**	**0.01**
Ours	R	17.76	4.21	2.35
	t	0.39	0.63	0.44
Ours + ICP	R	4.03	2.01	**0.31**
	t	0.10	0.31	0.08

Table 2. Effect of Gaussian noise on point cloud registration on ModelNet40 dataset

4.3. ModelNet Object Registration

Model40 [71] is a dataset consisting of 3D meshes in 40 different categories. For each category it contains synthetic CAD models. We sample random point clouds uniformly from the CAD model. For a fair comparison we follow the repository of [68] to sample 1024 points on the surface. For registration based on ModelNet objects, we randomly generate a rotation and translation (pose) and apply it to the source point cloud as in [68]. A random permutation is applied to the resulting list of target points. We then generate uniformly distributed patches with random seeds on three different scales of the point cloud to learn robust descriptors across different scales. It is worth mentioning that patches may not have a (fully) corresponding patch on the other point cloud. Each patch is then converted to a graph and sets of graphs are stored for each pose. We normalize the cloud into unit cube (1m) and apply perturbation augmentation by applying Gaussian noise ($\sigma = 0.1cm$) on point coordinates and normals. For both training and testing, we generate 70 patches per object to describe.

We compare our registration performance on the ModelNet40 [71] dataset based on the evaluation criteria provided in [68]. We train our model with the first 20 categories and evaluate with the 20 unseen categories. We first calculate Graphite features and keypoints per patch in each frame and then find correspondences based on Euclidean feature distance. Pairs of matched keypoints from our detected pool are then used to calculate a pose with an SVD-based pose estimation. We then calculate the Mean Average Error (MAE), Mean Squared Error (MSE) and Root Mean Squared Error (RMSE) on each rotation (in degrees) and translation (in cm) component. Table 1 compares our registration errors with state of the art registration methods on this dataset.

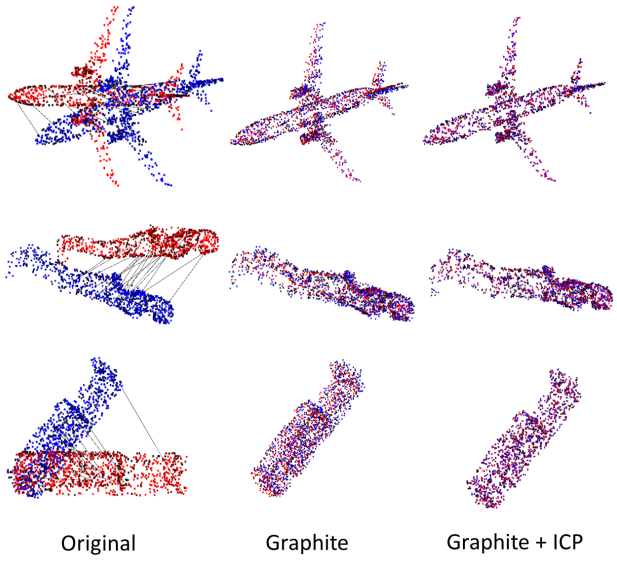


Figure 5. Qualitative evaluation on ModelNet40 [71] dataset registration in presence of noise. Graphite matching followed by RANSAC-based pose estimation provide an almost perfect initial pose for follow-up ICP refinement.

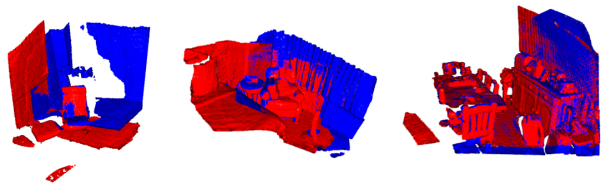


Figure 6. Qualitative evaluation on 3DMatch [76] dataset registration. Random registration examples from three scenes MIT, Home 1 and Hotel 1.

Methods such as ICP, Go-ICP and DCP are iterative optimization methods. Their convergence is sensitive to the ini-

Method	Handcrafted		Trained on 3DMatch						
	FPFH (33 dim)	SHOT (352 dim)	3DMatch (512 dim)	PPFNet (64 dim)	PPF-FoldNet (512 dim)	Perfect-Match (16 dim)	Perfect-Match (32 dim)	CGF (32 dim)	Graphite (32 dim)
Kitchen	43.1	74.3	58.3	89.7	78.7	93.1	97.0	60.3	64.82
Home 1	66.7	80.1	72.4	55.8	76.3	93.6	95.5	71.1	83.97
Home 2	56.3	70.7	61.5	59.1	61.5	86.5	89.4	56.7	68.26
Hotel 1	60.6	77.4	54.9	58.0	68.1	95.6	96.5	57.1	80.77
Hotel 2	56.7	72.1	48.1	57.7	71.2	90.4	93.3	53.8	80.53
Hotel 3	70.4	85.2	61.1	61.1	94.4	98.2	98.2	83.3	96.29
Study	39.4	64.0	51.7	53.4	62.0	92.8	94.5	37.7	82.53
MIT Lab	41.6	62.3	50.7	63.6	62.3	92.2	93.5	45.5	76.62
Average	54.3	73.3	57.3	62.3	71.8	92.8	94.7	58.2	79.22
STD	11.8	7.7	7.8	11.5	9.9	3.4	2.7	14.2	9.10

Table 3. Results on 3DMatch Geometric Registration Benchmark.

tialization. Therefore, the translation components is minimized comparably well based on the centroid of the full cloud, while the rotation estimate is not very robust in cases of minor overlap or for partial scans. We also add ICP as a consecutive refinement stage after calculating our pose. We reach state of the art rotation error with a significant margin on all estimated metrics.

4.4. Registration under Noise

We also study the robustness of our pipeline in presence of noise. We add Gaussian noise to the target point cloud coordinates with a standard deviation of 1 cm. Similar to [68], we test our approach with unseen test instances from all trained categories. We detect local keypoints per patch and measure descriptors to match them. We use an SVD based solver to predict the pose. Table 2 shows error results in comparison to the state of the art method on the ModelNet40 dataset. While the squared errors reflect some minor outliers, we keep being the method with best performance on MAE error with 0.31 degrees. Figure 5 shows sample registration results on ModelNet40[71] dataset. For extensive evaluations you can refer to supplementary material.

4.5. 3DMatch Descriptor Matching

The 3DMatch benchmark [76] is a 3D descriptor and geometric registration benchmark. It consists of 7 indoor scenes with multiple point cloud frames each. The point cloud instances are partial views of a fixed scene captured with an RGB-D sensors. We evaluate on this benchmark to assess the real-world applicability of Graphite. For evaluation, we follow the official repositories of [23, 76] where 5k keypoint coordinates are provided per frame. As the scans include a huge set of points, we first sub-sample points with voxel based sub-sampling and then form patches around the list of given seed coordinates. We form patches with $n = 225$ points to describe the same vicinity used to describe other features.

Contrary to the state of the art methods such as [76, 13, 14, 23] which have trained their models on 3dMatch data or other realistic scans, we have trained our model on synthetic point clouds only. This test demonstrates the transfer and generalization capabilities of our method applied on real data registration task.

Given the pool of stored locations from 3dMatch[76], we describe each local patch and perform matching with our descriptors, we then match them based on their Euclidean distance and perform RANSAC based registration. We take the same RANSAC iteration and settings as used in 3DSmoothNet [23]. We calculate recall values instructed in [13, 23] with $\tau_1 = 0.1m$ and $\tau_2 = 0.05$.

In Table 3 we present our results for the 3DMatch benchmark. We achieve a high recall rate in most of the scenes while having a dense representation (with 32 dimension) and a super lightweight model. In Figure 6 some example registrations drawn from the benchmark fragments are shown. Graphite demonstrate satisfying results even in cases with very small overlap. Moreover, in contrast to Perfect Match [23], we do not use a memory-hungry voxelization representation, but rely on computationally more efficient graph operations on point clouds through simple matrix multiplications presented in Eq. 3. For extensive qualitative evaluation of 3DMatch we refer the interested reader to our supplementary material.

5. Conclusion

We propose A lightweight patch descriptor which can represent point clouds ideal for expensive problems. Our graph-based model efficiently learns shape features and can detect salient keypoints given synthetic prior training followed by self-supervised metric learning. The extracted keypoints alongside the condensed descriptor can be used in the task of point cloud registration. We improve the state of art on object point cloud registration and prove solid performance and generalization on real indoor scans. Graphite can enable fast computation and dense representation of point clouds for modern 3D vision problems, replacing classical descriptors and sampling techniques.

References

[1] A. Alahi, R. Ortiz, and P. Vandergheynst. Freak: Fast retina keypoint. In *2012 IEEE Conference on Computer Vision and Pattern Recognition*, pages 510–517. Ieee, 2012.

[2] Y. Aoki, H. Goforth, R. A. Srivatsan, and S. Lucey. PointNetLK: Robust & efficient point cloud registration using PointNet. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 7163–7172, 2019.

[3] K. S. Arun, T. S. Huang, and S. D. Blostein. Least-Squares Fitting of Two 3-D Point Sets. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-9(5):698–700, 1987.

[4] V. Balntas, K. Lenc, A. Vedaldi, and K. Mikolajczyk. Hpatches: A benchmark and evaluation of handcrafted and learned local descriptors. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5173–5182, 2017.

[5] A. Barroso-Laguna, E. Riba, D. Ponsa, and K. Mikolajczyk. Key. net: Keypoint detection by handcrafted and learned cnn filters. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 5836–5844, 2019.

[6] H. Bay, T. Tuytelaars, and L. Van Gool. Surf: Speeded up robust features. In *European conference on computer vision*, pages 404–417. Springer, 2006.

[7] J. S. Beis and D. G. Lowe. Shape indexing using approximate nearest-neighbour search in high-dimensional spaces. In *Proceedings of IEEE computer society conference on computer vision and pattern recognition*, pages 1000–1006. IEEE, 1997.

[8] J. Bian, W.-Y. Lin, Y. Matsushita, S.-K. Yeung, T.-D. Nguyen, and M.-M. Cheng. Gms: Grid-based motion statistics for fast, ultra-robust feature correspondence. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4181–4190, 2017.

[9] M. M. Bronstein, J. Bruna, Y. Lecun, A. Szlam, and P. Vandergheynst. Geometric Deep Learning: Going beyond Euclidean data, 7 2017.

[10] B. Busam, M. Esposito, S. Che'Rose, N. Navab, and B. Frisch. A stereo vision approach for cooperative robotic movement therapy. In *Proceedings of the IEEE International Conference on Computer Vision Workshops*, pages 127–135, 2015.

[11] B. Busam, P. Ruhkamp, S. Virga, B. Lentes, J. Rackerseder, N. Navab, and C. Hennersperger. Markerless inside-out tracking for 3d ultrasound compounding. In *Simulation, Image Processing, and Ultrasound Systems for Assisted Diagnosis and Navigation*, pages 56–64. Springer, 2018.

[12] M. Calonder, V. Lepetit, C. Strecha, and P. Fua. Brief: Binary robust independent elementary features. In *European conference on computer vision*, pages 778–792. Springer, 2010.

[13] H. Deng, T. Birdal, and S. Ilic. Ppf-foldnet: Unsupervised learning of rotation invariant 3d local descriptors. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 602–618, 2018.

[14] H. Deng, T. Birdal, and S. Ilic. Ppfnet: Global context aware local features for robust 3d point matching. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 195–205, 2018.

[15] D. DeTone, T. Malisiewicz, and A. Rabinovich. Toward geometric deep slam. *arXiv preprint arXiv:1707.07410*, 2017.

[16] D. DeTone, T. Malisiewicz, and A. Rabinovich. Toward geometric deep slam. *arXiv preprint arXiv:1707.07410*, 2017.

[17] D. DeTone, T. Malisiewicz, and A. Rabinovich. Superpoint: Self-supervised interest point detection and description. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pages 224–236, 2018.

[18] J. Du, S. Zhang, G. Wu, J. M. Moura, and S. Kar. Topology adaptive graph convolutional networks. *arXiv preprint arXiv:1710.10370*, 2017.

[19] M. Dusmanu, I. Rocco, T. Pajdla, M. Pollefeys, J. Sivic, A. Torii, and T. Sattler. D2-net: A trainable cnn for joint description and detection of local features. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 8092–8101, 2019.

[20] M. Fey and J. E. Lenssen. Fast graph representation learning with PyTorch Geometric. In *ICLR Workshop on Representation Learning on Graphs and Manifolds*, 2019.

[21] M. A. Fischler and R. C. Bolles. Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. *Communications of the ACM*, 24(6):381–395, 1981.

[22] K. Fukunaga and P. M. Narendra. A branch and bound algorithm for computing k-nearest neighbors. *IEEE transactions on computers*, 100(7):750–753, 1975.

[23] Z. Gojcic, C. Zhou, J. D. Wegner, and A. Wieser. The perfect match: 3d point cloud matching with smoothed densities. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5545–5554, 2019.

[24] Y. Guo, F. Sohel, M. Bennamoun, M. Lu, and J. Wan. Rotational projection statistics for 3D local surface description and object recognition. *International Journal of Computer Vision*, 105(1):63–86, 10 2013.

[25] Y. Guo, F. A. Sohel, M. Bennamoun, M. Lu, and J. Wan. Trisi: A distinctive local surface descriptor for 3d modeling and object recognition. In *GRAPP/IVAPP*, pages 86–93, 2013.

[26] C. G. Harris, M. Stephens, et al. A combined corner and edge detector. In *Alvey vision conference*, volume 15, pages 10–5244. Citeseer, 1988.

[27] S. Hinterstoisser, V. Lepetit, S. Ilic, S. Holzer, G. Bradski, K. Konolige, and N. Navab. Model based training, detection and pose estimation of texture-less 3d objects in heavily cluttered scenes. In *Asian conference on computer vision*, pages 548–562. Springer, 2012.

[28] W. Kehl, F. Milletari, F. Tombari, S. Ilic, and N. Navab. Deep learning of local RGB-D patches for 3D object detection and 6D pose estimation. In *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, volume 9907 LNCS, pages 205–220. Springer Verlag, 2016.

[29] M. Khoury, Q.-Y. Zhou, and V. Koltun. Learning compact geometric features. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 153–161, 2017.

[30] M. H. Lee and I. K. Park. Performance evaluation of local descriptors for affine invariant region detector. In *Asian Conference on Computer Vision*, pages 630–643. Springer, 2014.

[31] K. Lenc and A. Vedaldi. Learning covariant feature detectors. In *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, volume 9915 LNCS, pages 100–117. Springer Verlag, 2016.

[32] S. Leutenegger, M. Chli, and R. Y. Siegwart. BRISK: Binary Robust invariant scalable keypoints. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2548–2555, 2011.

[33] Y. Li, R. Bu, M. Sun, W. Wu, X. Di, and B. Chen. PointCNN: Convolution On X-Transformed Points. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, editors, *Advances in Neural Information Processing Systems 31*, pages 820–830. Curran Associates, Inc., 2018.

[34] T. Lindeberg. Scale-space theory: A basic tool for analyzing structures at different scales. *Journal of applied statistics*, 21(1-2):225–270, 1994.

[35] T. Lindeberg. Feature detection with automatic scale selection. *International journal of computer vision*, 30(2):79–116, 1998.

[36] D. G. Lowe. Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, 60(2):91–110, 11 2004.

[37] B. D. Lucas and T. Kanade. An iterative image registration technique with an application to stereo vision. In *Proceedings of the 7th International Joint Conference on Artificial Intelligence - Volume 2*, IJCAI'81, page 674–679, San Francisco, CA, USA, 1981. Morgan Kaufmann Publishers Inc.

[38] E. Mair, G. D. Hager, D. Burschka, M. Suppa, and G. Hirzinger. Adaptive and generic corner detection based on the accelerated segment test. In *European conference on Computer vision*, pages 183–196. Springer, 2010.

[39] K. Mikolajczyk, T. Tuytelaars, C. Schmid, A. Zisserman, J. Matas, F. Schaffalitzky, T. Kadir, and L. Van Gool. A comparison of affine region detectors. *International journal of computer vision*, 65(1-2):43–72, 2005.

[40] O. Miksik and K. Mikolajczyk. Evaluation of local detectors and descriptors for fast feature matching. In *Proceedings of the 21st International Conference on Pattern Recognition (ICPR2012)*, pages 2681–2684. IEEE, 2012.

[41] A. Mishchuk, D. Mishkin, F. Radenovic, and J. Matas. Working hard to know your neighbor's margins: Local descriptor learning loss. In *Advances in Neural Information Processing Systems*, pages 4826–4837, 2017.

[42] F. Monti, D. Boscaini, J. Masci, E. Rodola, J. Svoboda, and M. M. Bronstein. Geometric deep learning on graphs and manifolds using mixture model cnns. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5115–5124, 2017.

[43] H. Moravec. Towards automatic visual obstacle avoidance. In *Proc. 5th Int. Joint Conf. Art. Intell., 1977*, 1977.

[44] M. Muja and D. Lowe. Flann-fast library for approximate nearest neighbors user manual. *Computer Science Department, University of British Columbia, Vancouver, BC, Canada*, 2009.

[45] R. Mur-Artal and J. D. Tardós. Orb-slam2: An open-source slam system for monocular, stereo, and rgb-d cameras. *IEEE Transactions on Robotics*, 33(5):1255–1262, 2017.

[46] Y. Ono, E. Trulls, P. Fua, and K. M. Yi. Lf-net: learning local features from images. In *Advances in neural information processing systems*, pages 6234–6244, 2018.

[47] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. In *Advances in neural information processing systems*, pages 8026–8037, 2019.

[48] C. R. Qi, H. Su, K. Mo, and L. J. Guibas. Pointnet: Deep learning on point sets for 3d classification and segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 652–660, 2017.

[49] C. R. Qi, L. Yi, H. Su, and L. J. Guibas. PointNet++: Deep hierarchical feature learning on point sets in a metric space. In *Advances in Neural Information Processing Systems*, volume 2017-December, pages 5100–5109. Neural information processing systems foundation, 2017.

[50] A. Rangarajan, H. Chui, and F. L. Bookstein. The softassign procrustes matching algorithm. In *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, volume 1230, pages 29–42. Springer Verlag, 1997.

[51] J. Revaud, P. Weinzaepfel, C. R. de Souza, and M. Humenberger. R2D2: repeatable and reliable detector and descriptor. In *NeurIPS*, 2019.

[52] E. Rosten and T. Drummond. Machine learning for high-speed corner detection. In *European conference on computer vision*, pages 430–443. Springer, 2006.

[53] E. Rosten, R. Porter, and T. Drummond. Faster and better: A machine learning approach to corner detection. *IEEE transactions on pattern analysis and machine intelligence*, 32(1):105–119, 2008.

[54] E. Rublee, V. Rabaud, K. Konolige, and G. Bradski. ORB: An efficient alternative to SIFT or SURF. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2564–2571, 2011.

[55] P. Ruhkamp, R. Gong, N. Navab, and B. Busam. Dynamite: A dynamic local motion model with temporal constraints for robust real-time feature matching. *arXiv preprint*, 2020.

[56] R. B. Rusu, N. Blodow, and M. Beetz. Fast point feature histograms (fpfh) for 3d registration. In *2009 IEEE international conference on robotics and automation*, pages 3212–3217. IEEE, 2009.

[57] R. B. Rusu, N. Blodow, Z. C. Marton, and M. Beetz. Aligning point cloud views using persistent feature histograms. In *2008 IEEE/RSJ international conference on intelligent robots and systems*, pages 3384–3391. IEEE, 2008.

[58] E. Salahat and M. Qasaimeh. Recent advances in features extraction and description algorithms: A comprehensive survey. In *2017 IEEE international conference on industrial technology (ICIT)*, pages 1059–1063. IEEE, 2017.

[59] P.-E. Sarlin, D. DeTone, T. Malisiewicz, and A. Rabinovich. Superglue: Learning feature matching with graph neural networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4938–4947, 2020.

[60] C. Schmid, R. Mohr, and C. Bauckhage. Evaluation of interest point detectors. *International Journal of computer vision*, 37(2):151–172, 2000.

[61] C. Silpa-Anan and R. Hartley. Optimised kd-trees for fast image descriptor matching. In *2008 IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–8. IEEE, 2008.

[62] S. M. Smith and J. M. Brady. Susan—a new approach to low level image processing. *International journal of computer vision*, 23(1):45–78, 1997.

[63] G. K. Tam, Z. Q. Cheng, Y. K. Lai, F. C. Langbein, Y. Liu, D. Marshall, R. R. Martin, X. F. Sun, and P. L. Rosin. Registration of 3d point clouds and meshes: A survey from rigid to Nonrigid, 2013.

[64] Y. Tian, B. Fan, and F. Wu. L2-net: Deep learning of discriminative patch descriptor in euclidean space. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 661–669, 2017.

[65] 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.

[66] T. Tuytelaars and K. Mikolajczyk. *Local invariant feature detectors: a survey*. Now Publishers Inc, 2008.

[67] Y. Verdie, K. Yi, P. Fua, and V. Lepetit. Tilde: A temporally invariant learned detector. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5279–5288, 2015.

[68] Y. Wang and J. M. Solomon. Deep closest point: Learning representations for point cloud registration. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 3523–3532, 2019.

[69] 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*, 38(5):13, 2019.

[70] P. Wohlhart and V. Lepetit. Learning descriptors for object recognition and 3d pose estimation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3109–3118, 2015.

[71] Z. Wu, S. Song, A. Khosla, F. Yu, L. Zhang, X. Tang, and J. Xiao. 3d shapenets: A deep representation for volumetric shapes. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1912–1920, 2015.

[72] J. Yang, Z. Cao, and Q. Zhang. A fast and robust local descriptor for 3D point cloud registration. *Information Sciences*, 346-347:163–179, 6 2016.

[73] Y. Yang, C. Feng, Y. Shen, and D. Tian. FoldingNet: Point Cloud Auto-Encoder via Deep Grid Deformation. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 206–215. IEEE Computer Society, 12 2018.

[74] Z. J. Yew and G. H. Lee. 3DFeat-Net: Weakly supervised local 3D features for point cloud registration. In *European Conference on Computer Vision*, pages 630–646. Springer, 2018.

[75] K. M. Yi, E. Trulls, V. Lepetit, and P. Fua. LIFT: Learned invariant feature transform. In *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, volume 9910 LNCS, pages 467–483. Springer Verlag, 2016.

[76] A. Zeng, S. Song, M. Nießner, M. Fisher, J. Xiao, and T. Funkhouser. 3dmatch: Learning local geometric descriptors from rgb-d reconstructions. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1802–1811, 2017.

[77] Y. Zhao, T. Birdal, H. Deng, and F. Tombari. 3D point capsule networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1009–1018, 2019.

[78] Q.-Y. Zhou, J. Park, and V. Koltun. Open3D: A modern library for 3D data processing. *arXiv:1801.09847*, 2018.