IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received 21 December 2022, accepted 13 January 2023, date of publication 23 January 2023, date of current version 26 January 2023.

Digital Object Identifier 10.1109/ACCESS.2023.3238901

RESEARCH ARTICLE

D3GATTEN: Dense 3D Geometric Features Extraction and Pose Estimation Using Self-Attention

BENJAMIN KELENYI AND LEVENTE TAMAS, (Member, IEEE)
Automation Department, Technical University of Cluj-Napoca, 400114 Cluj-Napoca, Romania

Corresponding author: Levente Tamas (levente.tamas@aut.utcluj.ro)

This work was supported in part by the Romanian National Authority for Scientific Research, CNCS-UEFISCDI, under Project PN-III-P2-2.1-PED2021-3120; in part by the European Union's Horizon 2020 Research and Innovation Program under Grant 871295; in part by the Project Network of Excellence in Applied Research and Innovation for Doctoral and Postdoctoral Programs/InoHubDoc; and in part by the European Social Fund under Agreement POCU/993-6-13/153437.

ABSTRACT Point-cloud processing for extracting geometric features is difficult due to the highly non-linear rotation variance and measurement noise corrupting the data. To address these challenges, we propose a new architecture, called Dense 3D Geometric Features Extraction And Pose Estimation Using Self-Attention (D3GATTEN), which allows us to extract strong 3D features. Later on these can be used for point-cloud registration, object reconstruction, pose estimation, and tracking. The key contribution of our work is a new architecture that makes use of the self-attention module to extract powerful features. Thoughtful tests were performed on the 3DMatch dataset for point-cloud registration and on TUM RGB-D dataset for pose estimation achieving 98% Feature Matching Recall (FMR). Our results outperformed the existing state-of-the-art in terms of robustness specification for point-cloud alignment and pose estimation. Our code and test data can be accessed at link: https://github.com/tamaslevente/trai/tree/master/d3gatten.

INDEX TERMS 3D point-clouds registration, self-attention, geometric features extraction, pose estimation.

I. INTRODUCTION

Point-cloud alignment is an important task in computer vision as this task forms the basis for many problems. It is a fundamental task for applications such as 3D reconstruction [1], simultaneous localization and mapping (SLAM) [2], tracking [3], flow estimation [4], AR/VR tracking [5], pose estimation for data fusion [6]. Point-cloud registration consists of computing the rigid transformation of two overlapping point-clouds in order to align them in the 3D space. However, given the properties of the point-cloud to be unordered, irregular, and often noisy, extracting pointwise correspondences is not a trivial task. In recent years, many methods have contributed to solving these problems, from hand-crafted methods [7], [8], [9], [10] to more recent deep learning-based approaches [11], [12], [13], [14], [15], [16], [17], [18].

The associate editor coordinating the review of this manuscript and approving it for publication was Yangmin Li.

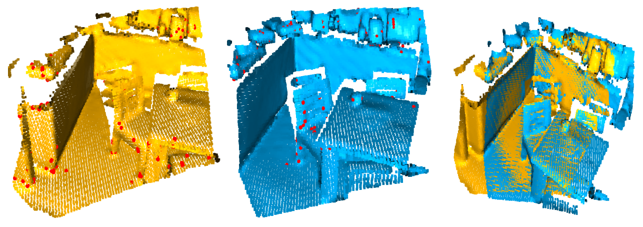


FIGURE 1. D3GATTEN isolates the essential points that can be used to produce a more precise point-cloud registration. The extracted keypoints are shown by red dots on Figure 1a, 1b, and Figure 1c depicts the registration of the two input point-clouds.

Deep learning-based methods that deal with point-cloud registration can be divided into several subcategories. The first class [19], [20], [21], [22] follows the principle of iterative closest-point (ICP) [23] base methods, where pose transformation and correspondence evaluations are performed.

The second category is correspondence-based methods [13], [15], [16], [17], [24], [25], where a neural network extracts the correspondences between two point-clouds, based on a random sampling technique such as RANSAC [26] for filtering and initial alignment. Most of these methods are based on the detection of keypoints [15], [16], [24], [27], [28]. These approaches may be divided into two categories based on how they extract correspondence [29]. The first class [15], [16], is based on extracting as many repeatable points as possible while the second class [24], [27], focuses on extracting all possible matches without detecting keypoints. Another category [24], [30], [31] first extracts the global feature vector for each point-cloud and then returns the transformation using the global feature vector. A different class [11], [13], [14], [15] is the attention-based methods [32], which are used to encode contextual information to register point-clouds.

We follow this path and introduce D3GATTEN: **D**ense **3**D **G**eometric Features Extraction And Pose Estimation Using Self-**Atten**tion for superior feature extraction using the self-attention module [32]. Our architecture uses D3Feat: Joint Learning of Dense Detection and Description of 3D Local Features [16], which adds a salient point detector to a fully convolutional feature descriptor. In addition to this, we implemented a self-attention module for feature extraction. Our algorithm is significantly more noise resistant after the addition of this module. This is a crucial consideration in order to apply this algorithm in real world. The results obtained for point-cloud registration can be seen in Figure 1c.

In summary, the key contributions of our work are as follows.

1) Based on D3Feat [16] we extend with a separate self-attention mechanism for selecting the most reliable keypoints for the registration of point-clouds;
2) the thorough analysis of different variants for the proposed extension with self-attention module based on the various backbones, module location in the pipeline, and profiling with respect to robustness against noise and runtime;
3) analysis and description of existing methods based on both hand-crafted and deep-learning methods.

Furthermore, to demonstrate that our algorithm performs well, we conducted several tests to evaluate its performance especially for robustness against noise. The public dataset used for testing was 3DMatch [33]. As a limitation of the proposed method, we observed the lack of outlier rejection ability which is the inherited characteristic from the base [16]. Nevertheless, according to our investigation, our method achieves results comparable to the state-of-the-art in terms of Feature Matching Recall (FMR) which is close to 98% with faster runtimes at the same time.

II. RELATED WORK

A. CLASSICAL METHODS

First, we briefly describe the hand-crafted keypoint feature-based methods for point-cloud registration. The early point-cloud local descriptors used hand-made features to describe local geometry. For a better understanding of these methods, we present them briefly.

1) USING SPIN IMAGES FOR EFFICIENT OBJECT RECOGNITION IN CLUTTERED 3D SCENES [7]

Andrew E. Johnson first proposed the spin image as a surface representation approach in [34], and it is used for surface matching and object detection in 3D images. In an object-oriented coordinate system rather than a viewer-oriented one, spin images encode the global attributes of any surface. Spin images: a representation for 3D surface matching (SPIN) [34] takes advantage of a projection of adjacent points on the tangential plane. The Spin Image descriptor is then calculated by adding the points in the support area to each bin of the 2D array, as shown in Figure 2a.

2) UNIQUE SIGNATURES OF HISTOGRAMS FOR SURFACE AND TEXTURE DESCRIPTION (SHOT) [10]

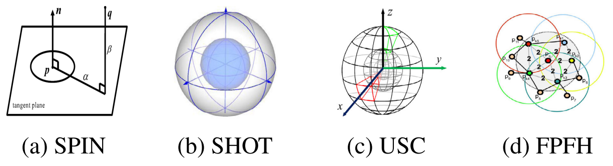
The surface normal histograms at distinct spatial regions are encoded by the descriptor. To begin with, a local reference frame (LRF) is built for the keypoint and its nearby points in the support area are aligned with it. As illustrated in Figure 2b, the support zone is then separated into various volumes along the radial, azimuth, and elevation axes. A local histogram is constructed for each volume by collecting point counts in bins based on the angles between the normals at nearby points inside the volume and the normal at the keypoint. Finally, by concatenating all of the local histograms, the SHOT [10] descriptor is created.

3) UNIQUE SHAPE CONTEXT FOR 3D DATA DESCRIPTION (USC) [8]

This method is a 3DSC [35] enhancement that eliminates the need to compute multiple descriptors at a single keypoint. For each keypoint, first, an LRF is created. The local surface is then aligned with the LRF to guarantee rigid transformation invariance. The keypoint's support area is then separated into numerous bins, as illustrated in Figure 2c. Finally, a USC [8] descriptor is constructed by adding the total number of points for each bin.

4) FAST POINT FEATURE HISTOGRAM (FPFH) [9]

As illustrated in Figure 2d, the first step is to construct a Simplified Point Feature Histogram (SPFH) for each point by computing the associations between the point and its neighbors. FPFH is calculated as the weighted sum of the



(a) SPIN (b) SHOT (c) USC (d) FPFH

FIGURE 2. Geometric feature-based methods [36].

SPFH of the feature point and the SPFH of the points in the support area.

B. LEARNING-BASED METHODS

The introduction of deep learning approaches in image processing has also greatly benefited this field. The learned 3D feature descriptors have recently taken over and now outperform the hand-crafted alternatives. Gojcic et al. [37] developed an end-to-end framework for multiview point cloud registration by directly learning to register every view of a scene in a uniform manner across all views. Teaser [38] resolves the rotating sub-problem via graded non-convexity. This technique makes effective use of Douglas-Rachford Splitting to confirm global optimality. The considerable computational cost in SDP relaxation is resolved by this technique. Using a set of criteria, Serafin and Grisetti [39] improved the convergence of the minimization function by pruning problematic correspondences using normal and tangent information, while also generating an extended evaluation of each point correspondence.

Many new methods have also been introduced as this area has become very challenging. In the following, we analyze the newest and most effective methods for point-cloud registration.

1) LEARNING COMPACT GEOMETRIC FEATURES [40]

A description has been established that can be applied immediately to a set of unordered points. This design has the advantage that no surface parameterization, volumetric representation, or additional depth image synthesis is required. This feature facilitates nearest-neighbor searches in Euclidean space, allowing dense mappings between point sets in near-linear time. Finally, this design relies on multilayer perceptrons (MLPs) to extract manually created features and transfer them to a compact feature space.

2) LEARNING THE MATCHING OF LOCAL 3D GEOMETRY IN RANGE SCANS [33]

is built on a fully convolutional siamese network architecture that learns rotation invariant 3D local feature descriptors. The 3DMatch extracts a feature for each interest point in a 3D point-cloud to integrate the local structure near the interest point. The 3D point-cloud must be converted into 3D volumetric data in 3DMatch [33], and the local representation is extracted by feeding the 3D volumetric data into the neural network.

3) GLOBAL CONTEXT AWARE LOCAL FEATURES FOR ROBUST 3D POINT MATCHING [17] AND UNSUPERVISED LEARNING OF ROTATION-INVARIANT 3D LOCAL DESCRIPTORS [41]

PPFNet [17] uses the PointNet [42] architecture to learn 3D patch descriptors by combining the characteristics of the pair of points. PPFNet, on the other hand, is not truly rotation-invariant. To solve this problem, PPF-FoldNet [41] uses mainly point pair features (PPF) as input, which is inherently rotation invariant, and integrates a FoldingNet [43] architecture to enable unsupervised training of rotation invariant descriptors.

4) FULLY CONVOLUTIONAL GEOMETRIC FEATURES (FCGF) [24]

This approach, as the name suggests, is full convolution. FCGF [24] generates high-resolution features rapidly and does not require low-level preprocessing or 3D patching as input. To expand the receptive field and extract geometric information, a fully convolutional 3D network with metric learning is used. It recovers the transformation using robust pose estimators after extracting correspondences between two point-clouds.

5) D3Feat: JOINT LEARNING OF DENSE DETECTION AND DESCRIPTION OF 3D LOCAL FEATURES [16]

The D3Feat [16] design is mainly based on the KPConv [44] architecture, with the extraction of features from a point-cloud taking over. D3Feat introduced a keypoint selection method that overcomes intrinsic density changes of 3D point clouds, as well as a self-supervised detector loss driven by on-the-fly feature matching findings during training. The incorporation of a descriptor with a salient point detector is D3Feat's contribution to this fully convolutional design.

6) DeepVCP: AN END-TO-END DEEP NEURAL NETWORK FOR 3D POINT CLOUD REGISTRATION [45]

Extracts local features using PointNet++ [46], then filters them using a weighted layer that maintains just the most relevant ones. The characteristics descriptors are computed using a miniPointNet structure and then passed into a corresponding point generation layer, which creates the relevant key points in the target point cloud. In order to regress the final result of the transformation, two loss functions are coupled, hoping to encode both the local similarities and global geometric limitations.

C. ATTENTION BASED METHODS

1) NEIGHBORHOOD-AWARE GEOMETRIC ENCODING NETWORK FOR POINT-CLOUD REGISTRATION [11]

To encode the information of the point-clouds, NgeNet [11] uses siamese shared layers with a new geometric-encoding interaction module applied to the superpoints, as well as multi-scale parallel decoding layers to extract multi-level point-wise characteristics for each point-cloud. On indistinguishable surfaces, a learning-free consistency voting system is designed to choose the feature with the appropriate neighborhood for each point and eliminate erroneous features.

2) CoFiNet: RELIABLE COARSE-TO-FINE CORRESPONDENCES FOR ROBUST POINT-CLOUD REGISTRATION [13]

CoFiNet primarily uses a KPConv-based [44] encoder-decoder architecture. For context aggregation, the authors incorporated two attention-based networks into this design.

In the first phase, the dense points are down-sampled to evenly distributed nodes, and the features are enhanced before being utilized to generate the similarity matrix. The confidence matrix is then used to offer coarse node correspondences.

3) GEOMETRIC TRANSFORMER FOR FAST AND ROBUST POINT-CLOUD REGISTRATION [14]

Learns features at several resolution levels by down-sampling the input point-clouds. Using the Geometric Transformer, which iteratively encodes intra-point-cloud geometric structures and inter-point-cloud geometric consistency, the architecture recovers high-quality superpoint correspondences between the source and target point-cloud. Finally, a local-to-global registration approach is used to compute the transformation among the point-cloud.

4) PREDATOR: REGISTRATION OF 3D POINT-CLOUDS WITH LOW OVERLAP [15]

The network is built using the KPConv [44] architecture and can be split into three main steps. In the first step, the two input point-clouds are converted into smaller sets of super points and encoded with the associated latent features with common weights. Then, using an overlapping attention module to extract co-contextual information. In the last phase, the network anticipates dense overlap scores and indicates the confidence of degree whether the points are in the overlap regions.

III. PRELIMINARIES

A. PROBLEM STATEMENT

Given the point-cloud source $\mathcal{S} = \left\{\mathbf{s}_i \in \mathbb{R}^3\right\}_{i=1,2,\cdots,Y}$ and the point-cloud target $\mathcal{T} = \left\{\mathbf{t}_j \in \mathbb{R}^3\right\}_{j=1,2,\cdots,Z}$, where Y and Z are the cardinality in the point-cloud source and target. The main idea of point-cloud registration is to find a rigid transformation $T \in SE(3)$ that best aligns the source and target point-cloud. The transformation can be recovered by solving:

$$\underset{T}{\arg\min} \frac{1}{|\kappa|} \sum_{(i,j)\in\kappa} \left\| T\left(\mathbf{s}_i\right) - \mathbf{t}_j \right\|_2 \tag{1}$$

where κ is the set of correspondences and $|\cdot|$ denotes the cardinality of the set.

B. SELF-ATTENTION MECHANISM

Attention was first presented as an extension of recurrent neural networks (RNNs) [47]. The transformer mechanism introduced in [32] is a significant advance in attention research, as it reveals that the attention mechanism can obtain state-of-the-art results. The self-attention mechanism was originally developed for natural language processing but was immediately adopted for other tasks, such as processing images [48] and videos [49].

Given 3 matrices Q (queries), K (keys) and V (values), which are projections of the input of the layer, the output of the attention mechanism is the weighted sum of the values multiplied by the compatibility score between queries and keys. Scores indicate how much attention should be paid to other locations or words in the input sequence. Once we determined our keys (K), queries (Q), and values (V), we can compute attention as follows:

$$Attention(Q, K, V) = Softmax\left(\frac{Q \cdot K^T}{\sqrt{d_k}}\right) V \tag{2}$$

First, the dot product of the query and the key are computed. If we execute this on a large number of queries and keys at the same time, we can express the dot products as matrix multiplication, $Q \cdot K^T$, where Q is a vector of queries, and K is a vector of keys. Then it is divided by the square root of the dimensions of the key vector $\sqrt{d_k}$, to prevent the dot products from being too large as the vector length grows. The softmax function rescales the values between 0 and 1 and regularizes them. Finally, the result (weights) is multiplied by the value (all words) to lower the relevance of irrelevant terms and focus only on the most significant ones.

IV. PROPOSED METHOD

A. NETWORK ARCHITECTURE

Figure 3 depicts the architecture of our method. The segmentation part of KPConv [44] has been adapted. The network consists of 5 convolutional layers. Two convolutional blocks make up each layer, with the exception of the first layer's first block being strung together. Our convolutional blocks are constructed similarly to bottleneck ResNet blocks [50] with batch normalization and leaky ReLu activation. The layers of the encoding and decoding parts are connected by skip connections. Extracting the most accurate and powerful features is done using the suggested self-attention module. In the sections below, we present our architecture and the proposed self-attention module. To better understand our approach, we start with a diagram, which describes the architecture of the proposed self-attention-based model and can be seen in Figure 4.

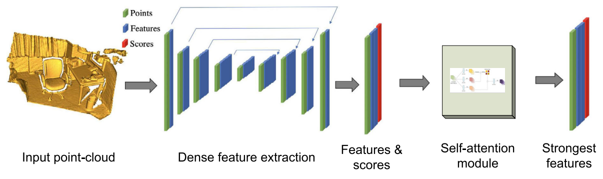


FIGURE 3. D3GAtten's network architecture. Each block is a ResNet [50] block that uses KPConv to replace image convolution. Except for the last layer, all layers are followed by batch normalization and ReLU.

B. KEYPOINT DETECTION PIPELINE

Inspired by D2-Net, a recent technique for 2D image matching [51], we develop a single neural network that serves two functions: a dense feature descriptor and a feature detector. However, because of the inconsistent structure and variable sparsity of point clouds, applying the D2-Net concept to the

3D world is not straightforward. Following that, we outline the basic procedures for performing feature description and detection on 3D point clouds with irregular shapes before outlining how the sparsity variation in the 3D domain is handled.

1) DENSE FEATURE EXTRACTION

To perform dense feature extraction, we use KPConv [44] as our backbone network. We will first go over the KPConv formulas briefly.

Given a set of points $P \in \mathbb{R}^{N\times 3}$ and a set of features $F \in \mathbb{R}^{N\times D}$, let x_i and f_i denote the i-th point in P and its corresponding feature in F. The definition of the generic convolution by kernel g at point x is as follows:

$$(F * g) = \sum_{x_i \in N_x} g(x_i - x) f_i, \tag{3}$$

where N_x denotes the radius neighborhood of point x and x_i denotes a supporting point inside this radius neighborhood. The kernel function is defined as follows:

$$g(x_i - x) = \sum_{k=1}^{K} h\left(x_i - x, \hat{x}_k\right) W_k, \tag{4}$$

where h represents the correlation function between the kernel point x_k and the supporting point x_i, K is the number of kernel points, and Wk is the weight matrix of the kernel point $\hat{x}_k$. To guarantee that convolution is sparsity invariant, a density normalization factor is added to Eq. 3 that adds up the number of supporting points close to x:

$$(F * g) = \frac{1}{|N_x|} \sum_{x_i \in N_x} g(x_i - x) f_i \tag{5}$$

Our network produces a dense feature map as a two-dimensional matrix F as its output.

2) DENSE KEYPOINT DETECTION

In D2-Net [51] the local maximums within and across the deep feature maps channels are defined as keypoints, using the same maps that were used for descriptors. To address the non-uniform sample setup of point clouds, this procedure might be replaced with a radius neighborhood to expand their approach to 3D. Local regions with few points (for example, regions near to the limits of interior scenes or far from the Lidar center of outdoor scenes) would have higher scores if we used a softmax function to estimate the local maximum in the spatial dimension. To address this issue, a density-invariant saliency score to assess a point's saliency in relation to its local neighbourhood is provided.

Using the dense feature map $F \in \mathbb{R}^{N\times c}$ as input, the following 3D responses are possible $D^k (k = 1, \ldots, c)$:

$$D^k = F_{:k}, \quad D^k \in \mathbb{R}^N \tag{6}$$

where $F_{:k}$ is used to indicate the k-th column of the two-dimensional matrix F. Therefore, to discover a keypoint x_i, we need:

$$x_i \iff k = \arg\max_t D_i^t \text{ with } i = \arg\max_{j \in N_{x_i}} D_j^k \tag{7}$$

where N_{x_i} is the radius neighborhood of x_i.

The local-max score in D2-Net [51] is defined as:

$$\alpha_i^k = \frac{\exp\left(D_i^k\right)}{\sum_{x_j \in N_{x_i}} \exp\left(D_j^k\right)} \tag{8}$$

This formulation is not sparsity-invariant. Because the ratings are adjusted by sum, sparse locations naturally receive higher scores than dense ones. As a result, the following density-invariant saliency score is created:

$$\alpha_i^k = \ln\left(1 + \exp\left(D_i^k - \frac{1}{|N_{x_i}|} \sum_{x_j \in N_{x_i}} D_j^k\right)\right) \tag{9}$$

According to this method, a point's saliency score is determined by subtracting its characteristic from the mean characteristic of its local neighborhood.

To pick up the most preeminent channel for each point, a channel max score formula was designed:

$$\beta_i^k = \frac{D_i^k}{\max_t \left(D_i^t\right)} \tag{10}$$

The final keypoint detection score is calculated by combining the two scores:

$$s_i = \max_k \left(\alpha_i^k \beta_i^k\right) \tag{11}$$

We choose the points with the highest scores as keypoints after obtaining the keypoint score map of an input point cloud.

C. PROPOSED SELF-ATTENTION

The fastai implementation of the self-attention layer described in the SAGAN [52] paper is modified and simplified in our approach. We first briefly outline the current approach and then describe our method.

According to the SAGAN [52] paper, to calculate attention, the image features of the previously hidden layer $F \in \mathbb{R}^{C\times N}$ are first translated into two feature spaces, f and g, where $f(F) = W_f F$ and $g(F) = W_g F$ with C being the number of channels, and N denotes the number of feature locations of features from the previously hidden layer. For self-attention, $\beta_{j,i}$ is computed as:

$$\beta_{j,i} = \frac{\exp\left(s_{ij}\right)}{\sum_{i=1}^{N} \exp\left(s_{ij}\right)} \tag{12}$$

where: $s_{ij} = f\left(F_i\right)^T g\left(F_j\right)$ and $\beta_{j,i}$ specifies how much attention the model pays to the i^{th} location while synthesizing the j^{th} region.

The attention layer's output is:

$$O = (O_1, O_2, \ldots, O_j, \ldots, O_N) \in \mathbb{R}^{C\times N} \tag{13}$$

where:

$$O_j = v\left(\sum_{i=1}^{N} \beta_{j,i} h\left(F_i\right)\right), h\left(F_i\right) = W_h F_i, v\left(F_i\right) = W_v F_i \tag{14}$$

In the preceding formulation W_f, W_g, W_h and W_v are the learned weight matrices, which are realized by $1 * 1$ convolutions. In addition, the output of the attention layer is multiplied by a scale parameter and added back to the input feature map.

As a result, the final output is provided by: $y_i = \gamma O_i + F_i$, where γ is a learnable scalar that is initially set to 0.

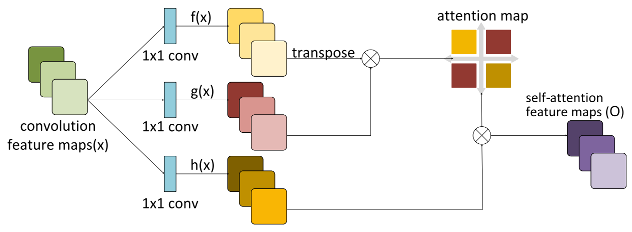


FIGURE 4. Our self-attention module from Figure 3. Matrix multiplication is denoted by the symbol ⊗.

1) OUR SUGGESTED SELF-ATTENTION MODULE

The original layer takes the image features x of shape (C, N) where, $N = H \cdot W$, and transforms them into $f(x) = W_f \cdot x$ and $g(x) = W_g \cdot x$ where, W_f and W_g have shape (C, C'), and C' is chosen to be $C/8$. These matrix multiplications can be expressed as 1×1 convolution layers. Then, we compute $S = f(x)^T \cdot g(x)$. Therefore, $S = (W_f \cdot x)^T \cdot W_g \cdot x = x^T * W_f^T \cdot W_g \cdot x$. Our first proposed simplification is to combine $W_f^T \cdot W_g$ into a single $(C \times C)$ matrix W. Therefore, $S = x^T \cdot W \cdot x = S(x, x)$ (bilinear form) is of shape $(N \times N)$ and will represent the influence of each pixel on the other pixels. As a result, instead of learning weights W_f and W_g for two convolution layers, we just learn weights W for one convolution layer. Advantages are simplicity, removal of one design choice $C' = C/8$, and a matrix W that offers more possibilities than $W_f^T \cdot W_g$. Computing the *softmax* of the matrix S is the following step in the original layer design. We decided to skip this step entirely and operate with unrestricted weights rather than adjusted probability-like weights. The final step in the original version is to calculate $h(x) = W_h \cdot x$ where W_h is of shape $C \times C$, which is also implemented as a convolution layer 1×1. We propose to remove this final convolution layer and have the output be $y_i = \gamma \cdot x \cdot S + x$. If desired, this last convolution can be re-added as a separate layer, implying a new position for the skip connection.

2) SUGGESTED SELF-ATTENTION MODULE EXPERIMENTS

Comparing the enhanced Self-Attention layer to the original self-attention layer, it appears to offer a significant improvement in terms of complexity. The original self-attention layer used *softmax* on a matrix $N \times N$, which is $O(N^2)$, because *softmax* is performed N times and *softmax* is $O(N)$ according to [53]. In terms of a runtime comparison, we put our self-attention model up against ResNet, VGG and AlexNet models. To make an accurate comparison, we evaluated all models with the same configuration(nr. of epochs = 50, dataset=Imagewoof, image size = 128 and nr. of runs = 20). In the last column of Table 1 we reported the wall clock time.

As shown in Table 1, our suggested self-attention module reduces runtime while producing results that are similar to those of the original self-attention.

TABLE 1. Comparison of the original and our self-attention modules. We can see the result of the original architecture on the first line, the original self-attention module on the second line, and the modified self-attention layer on the last line.

Model	Accuracy	Avg. time (m:s)
VGG16	0.9156	21:06
VGG16 + sa	0.9274	23:41
VGG16 + our sa	0.9262	20:54
AlexNet	0.8559	10:06
AlexNet + sa	0.8673	11:48
AlexNet + our sa	0.8794	9:53
xResNet18	0.8497	9:38
xResNet18 + sa	0.8548	11:18
xResNet18 + our sa	0.8565	9:27

D. IMPLEMENTATION DETAILS

In terms of architectural implementation, our network is built in PyTorch [54]. D3Feat [16] served as a starting point for our implementation. We train with momentum SGD, with a fixed learning rate of 0.01, a momentum of 0.98, and a weight decay of $1e - 6$. For each point-cloud fragment, we enhance the data by adding Gaussian noise with a standard deviation of 0.015. Except for the last layer, all layers are followed by batch normalization and ReLU. The encoder section has a fixed number of channels (64, 128, 256, 512, 1024). Between the corresponding layers of the encoder and decoder parts, residual connections are used. To obtain the final 32-dimensional features, the output features are computed using a convolution 1×1.

V. EXPERIMENTAL VALIDATION

A. PAIRWISE REGISTRATION EVALUATION

To validate our proposed method, we used the 3DMatch [33] indoor benchmark to test our model for registration and TUM RGB-D benchmark [55] for pose estimation. Additionally, we compared our method to the most recent point-cloud registration methods. We utilize the same approaches to prepare the training and testing data as in the 3DMatch dataset [21]. There are 62 scenarios, 46 of which are used for training, 8 for validation, and 8 for testing. In 3DMatch, point-cloud pairs have in average 30% overlap.

1) EVALUATION METRICS

We employ three metrics to evaluate the performance under registration for the 3DMatch benchmark: registration recall (RR), feature match recall (FMR), and inlier ratio (IR).

Registration Recall (RR). The proportion of scan pairings for which the appropriate transformation parameters are identified with RANSAC [26]. It computes the root mean square error among Ω^* under the predicted transformation $\hat{\mathbf{T}}$:

$$RMSE = \sqrt{\frac{1}{\Omega^*} \sum_{(\mathbf{x}^*,\mathbf{y}^*)\in\Omega^*} \left\| \hat{\mathbf{T}}_{i,j}\mathbf{x}^* - \mathbf{y}^* \right\|^2} \tag{15}$$

and calculates the fraction of alignments with $RMSE < 0.2m$. Where: Ω^* is a collection of ground truth pairings in fragments $\{i, j\}$ and $\mathbf{x}^*$ and $\mathbf{y}^*$ are the 3D coordinates of the ground truth pair.

Feature Match Recall (FMR): Defined as the fraction of pairs with $\tau_2 = 5\%$ "inlier" matches with a $\tau_1 = 10$ cm residual under the ground truth transformation $\mathbb{1}$ as follows:

$$\frac{1}{M}\sum_{s=1}^{M} \mathbb{1}\left(\left[\frac{1}{|\Omega_s|}\sum_{(i,j)\in\Omega_s} \mathbb{1}\left(\left\|\mathbf{T}^*\mathbf{x}_i - \mathbf{y}_j\right\| < \tau_1\right)\right] > \tau_2\right) \tag{16}$$

where: M is the total pair of point-clouds, Ω_s is the correspondence between a pair of point-clouds (source and target), $\mathbf{x}_i$ and $\mathbf{y}_i$ are the 3D coordinates from the point-cloud source and target, T^* is the estimation of the ground truth.

Inlier Ratio (IR). The proportion of accurate correspondences among the putative matches, in addition to the previous metrics (1) and (2). To estimate the transformation matrix for the metric (2), we use a RANSAC with 50,000 maximum iterations, as described in [21].

2) COMPARISONS WITH THE STATE OF THE ARTS

In Table 2 we compare our method's Feature Matching Recall (FMR) to the state of the art: SPIN [34], SHOT [8], FPFH [9], USC [8], CGF [40], 3DMatch [33], PPFNet [17], PPF-FoldNet [41], DirectReg [56], CapsuleNet [57], FCGF [24], D3Feat [16], PREDATOR [15], GeoTransformer [14], CoFiNet [13] and NgeNet [11]. The authors' existing codes were used to obtain the results of the other methods rather than re-implementing them. FMR is presented for two types of data: original point-cloud fragments (left columns) and randomly rotated $(0-2\pi)$ fragments (right columns). The results suggest that our method outperforms most other methods on this dataset, with a top performance of 98.3 percent. Furthermore, we demonstrate the resilience of our method in FMR by adjusting the error threshold ($\tau_1 = 10$cm and $\tau_2 = 5\%$, the dashed line in Figure 5). We recommend using these values between ($\tau_1 = 7,5 - 12,5$cm and $\tau_2 = 2,5-7,5\%$) Under a stricter inlier ratio criteria, such as $\tau_2 = 20\%$, D3GAtten significantly outperforms other approaches. In terms of inlier distance error D3GAtten performs slightly worse than CoFiNet [13], this is most likely because to the lower voxel size (2.5cm) utilized in CoFiNet, whereas D3GAtten uses 3cm voxel downsampling.

The sensitivity of the characteristic to the inlier distance threshold τ_1 and the inlier ratio threshold τ_2 is demonstrated in Figure 5. Overall, our techniques outperform other methods in terms of Feature Match Recall across diverse scenes, as well as a wide variety of distance and inlier recall thresholds. For run-time, our method is in the same range as the competitors.

TABLE 2. Feature-matching recall and its standard deviation for the original and rotated data.

Samples	Original FMR (%)	STD	Rotated FMR (%)	STD
Spin [34]	22.7	1.14	22.7	1.21
SHOT [10]	23.8	10.9	23.4	9.5
FPFH [9]	35.9	1.34	36.4	1.36
USC [8]	40.0	1.25	-	-
CGF [40]	58.2	14.2	58.5	14.0
3DMatch [33]	59.6	8.8	1.1	1.2
PPFNet [17]	62.3	10.8	0.3	0.5
PPF-FoldNet [41]	71.8	10.5	73.1	10.4
DirectReg [56]	74.6	0.94	-	-
CapsuleNet [57]	80.7	0.62	80.7	0.6
3DSN [33]	94.7	2.7	94.9	2.4
FCGF [24]	95.2	2.9	95.3	3.3
D3Feat [16]	95.8	2.9	95.6	3.5
PREDATOR [15]	96.7	2.5	96.6	2.7
GeoTransformer [14]	97.9	4.1	97.9	4.5
CoFiNet [13]	98.1	4.1	98.1	4.3
NgeNet [11]	98.2	3.6	**98.2**	3.6
Ours	**98.3**	2.4	**98.2**	2.8

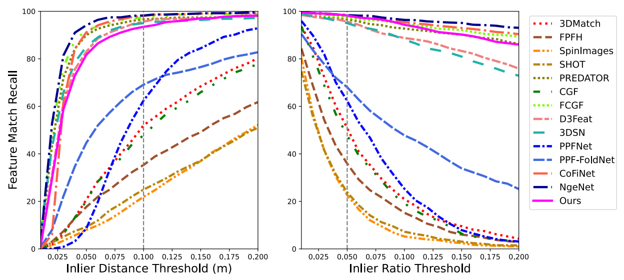


FIGURE 5. Feature matching recall in relation to inlier ratio threshold τ_2 and inlier distance threshold τ_1.

3) EVALUATION UNDER DIFFERENT NUMBERS OF KEYPOINTS

We also provide the results when lowering the sampled point number from 5000 to 2500, 1000, 500, or even 250 to further highlight the advantages of using a self-attention module when the keypoints are selected. As shown in Table 3, our strategy is one of the most effective when employing 5000 points for Feature Matching Recall. When keypoints were lowered, our method produced results that were comparable to those achieved by other methods, but not the best. Because our keypoint selection is based on a self-attention module, this was to be expected. Self-attention modules require an even number of points as input to provide effective results. Outliers from a coarse scale are not explicitly rejected by design. False coarse correspondences can be developed into false point correspondences, which may lead to a lower inlier ratio on a finer scale. Despite these flaws, our module has been able to produce competitive results in terms of the inlier ratio.

4) ROTATION AND TRANSLATION INVARIANCE

Rotation and translation invariance is one of the most significant aspects of excellent geometric features. Experiments show that via low-cost data augmentation, a fully convolutional network can empirically obtain significant rotation invariance. We train the model without rotation augmentation and evaluate it on the 3DMatch dataset to show the impact of basic data augmentation on rotation robustness.

TABLE 3. Evaluation table with different numbers of keypoints for FMR, RR and IR.

Keypoints	**5000**	**2500**	**1000**	**500**	**250**
Feature Matching Recall (%)					
3DSN	94.7	94.2	92.6	90.1	82.9
FCGF	95.2	95.5	94.6	93.0	89.9
D3Feat	95.8	95.6	94.6	94.3	93.3
PREDATOR	96.6	96.6	96.5	96.3	96.5
GeoTransformer	97.9	97.9	97.9	97.9	97.6
CoFiNet	98.1	98.3	**98.1**	98.2	**98.3**
NgeNet	98.2	**98.4**	**98.1**	**98.6**	98.2
Ours	**98.3**	98.0	97.2	95.6	95.3
Registration Recall (%)					
3DSN	80.3	77.5	73.4	64.8	50.9
D3Feat	82.2	84.4	84.9	82.5	79.3
Ours	84.1	85.5	85.0	81.0	73.8
FCGF	87.3	85.8	85.8	81.0	73.0
PREDATOR	89.0	89.9	90.6	88.5	86.6
CoFiNet	89.3	88.9	88.4	87.4	87.0
GeoTransformer	92.0	91.8	91.8	**91.4**	**91.2**
NgeNet	**92.9**	**92.1**	**92.3**	90.3	88.7
Inlier Ratio (%)					
3DSN	37.7	34.5	28.3	23.0	19.1
D3Feat	40.7	40.6	42.7	44.1	45.0
CoFiNet	49.8	51.2	51.9	52.2	52.2
Ours	52.3	52.2	52.4	51.4	49.5
FCGF	56.9	54.5	49.1	43.3	34.7
PREDATOR	58.0	58.4	57.1	54.1	49.3
NgeNet	63.1	63.5	61.5	57.6	51.1
GeoTransformer	**71.9**	**75.2**	**76.0**	**82.2**	**85.1**

TABLE 4. Recall of feature matches in 3DMatch rotated with and without rotation augmentation for D3GATTEN.

Keypoints	**5000**	**2500**	**1000**	**500**	**250**
Feature Matching Recall (%)					
w\o. rot. aug.	6.1	5.2	4.4	4.3	4.3
w. rot. aug.	96.2	94.5	94.6	90.2	87.9

To test the rotation invariance of our algorithm, we rotate all fragments of the 3DMatch dataset in the same way as [41]. All of the fragments in the 3DMatch test set are rotated along all three axes with a random angle taken from a uniform distribution across $[0 - 2\pi]$. The results are shown in Table 4. We evaluated the model at several keypoints; our model without rotation augmentation is unable to learn rotations from the data, which is why we get such bad results. The last column represents the result obtained after applying the transformation. As one can see, our algorithm registered the two point-clouds without difficulties, being in the top 5 methods with the best FMR for a large number of keypoints.

5) REGISTRATION RESULTS

In order to demonstrate the results obtained in Table 2, we created a visualization to see how our algorithm behaves. As can be seen in Figure 6, on the first two columns we have the point-clouds as input (source and target) on the 3^{rd} column, we have the two inputs superimposed without applying the transformation calculated with our method, and on the last column, we can see the registration results.

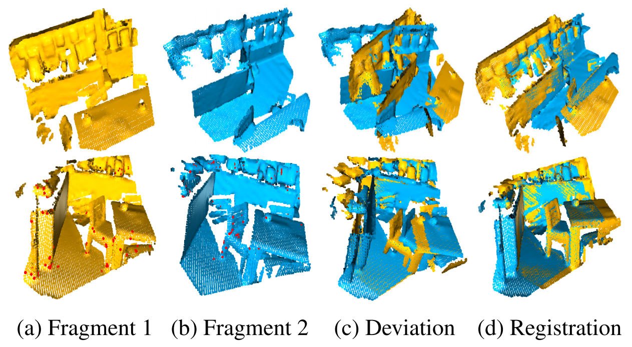


FIGURE 6. Example results on 3DMatch dataset. The point-clouds that will be registered are represented by the first two columns (a) and (b). The standard deviation of the two point-clouds is represented in the third column (c), and the result produced after the transformation is represented in the last column (d).

6) VOTE-RELATED INFLUENCE

A sample of how consistent voting works is shown in Figure 7. Consistent voting gives each point the most discriminative characteristic possibility, which causes massively mismatched relationships to be pruned and the proper ones to be reserved.

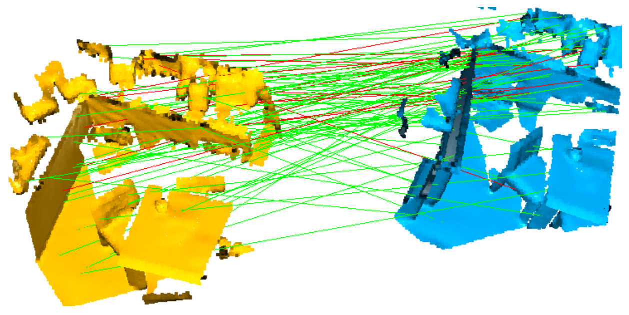


FIGURE 7. Correspondences built using feature matching and reliable voting. 150 correspondences are chosen at random for ease of display. The obtained proper correspondences are highlighted in green here. The erroneous correspondences are shown with red.

7) EXPERIMENTS WITH NOISY DATASET

This section focuses on noise robustness testing methodologies. We compared our technique with the most competitive current methods, including NgeNet [11], CoFiNet [13], and PREDATOR [15]. Often, images obtained from the camera are corrupted by noise. Taking into account this fact, we set out to simulate the real-world environment, so we introduced two types of noise: occlusion noise and Gaussian noise. We also addressed the problem where training is done on data without noise and evaluated on data with noise, respectively, trained on data with noise and evaluated on data with noise.

a: TRAINING FOR THE NOISY DATASET

The following situations were considered: 1) we trained the model on noiseless training data and tested it on noisy data. The goal of this test is to see how a pre-trained model performs on a real dataset that is corrupted by noise (sim2real). 2) We trained the model using noise-corrupted training data and evaluated it using the identical data set as in the first scenario. This would be equivalent to (real2real) approach. For each type of noise used, two types of testing were carried out. The settings that specify the radius of the circle for occlusion noise were 30cm and 50cm. For Gaussian noise, the standard deviation parameter was set at 0.5 in the initial phase and then increased to 0.8. The findings are provided in the following subsections.

b: OCCLUSION NOISE

The occlusion test addresses one of the most common problems in the object identification process, the object occlusion problem. This effectively indicates that the template is only partially visible, i.e. an obscured observation is accessible for recognition. In our experimental setting, we tried changing the occlusion by eliminating a radial zone of an item in a random place. The number of points deleted is expressed as a tuning parameter as a percentage of the template's point-cloud size. The results obtained after applying the algorithm are visible in Figure 8a and Figure 8b.

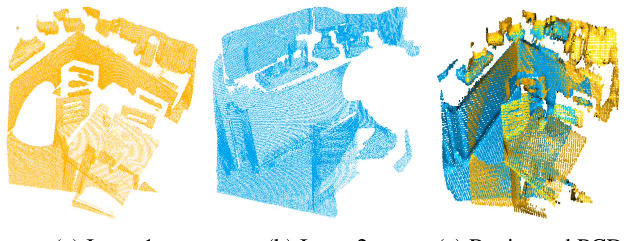


FIGURE 8. **Occlusion noise results with own method. For these experiments, we adjusted the occlusion radius level to 50cm. The point-clouds that will be registered are represented by the first two images (a) and (b). The result produced after the transformation is represented in the last image (c).**

c: GAUSSIAN NOISE

The most typical issue with real-time data is sensor noise, which refers to measurement errors caused by physical sensor limits. The non-systematic element of the noise is often modelled using a Gaussian probability distribution. This assumption was likewise applied in our situation, and we evaluated a system with variable additional Gaussian noise covariance along the XYZ coordinates added separately. The results obtained after applying Gaussian noise, can be seen in Figure 9a and Figure 9b.

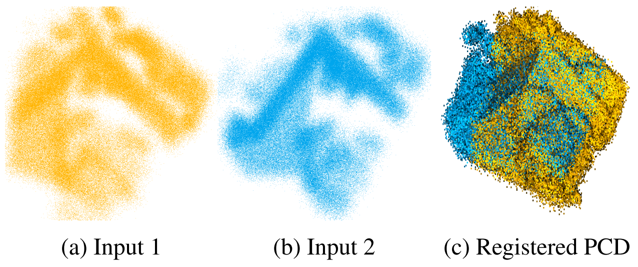


FIGURE 9. **Gaussian noise computed the data with its own method. For these experiments, we adjusted the standard deviation to 0.08. The point-clouds that will be registered are represented by the first two images (a) and (b). The result produced after the transformation is represented in the last image (c).**

d: OVERALL RESULTS ON NOISY DATASET

Table 5 illustrates the outcomes after training on noise-free data and evaluating on noisy data, whereas Table 6 shows the results after training with noisy data. Figure 10 and 11 provide a graphic illustration of the results.

TABLE 5. **Results of noiseless training and noisy evaluation. Notation "Ours w\o. sa" means "our algorithm without self-attention module" and "Ours w. sa" means "our algorithm with self-attention module".**

Noise type	**Occlusion noise (m)**		**Gaussian noise**	
Noise level	**0.3**	**0.5**	**0.05**	**0.08**
Feature Matching Recall (%)				
PREDATOR	93.63	92.29	42.16	23.18
CoFiNet	94.43	92.21	55.16	18.68
NgeNet	95.50	94.00	31.8	9.8
Ours w\o. sa	94.61	93.28	47.43	29.56
Ours w. sa	**97.79**	**96.39**	**68.22**	**49.81**
Inlier Ratio (%)				
PREDATOR	46.16	43.88	6.9	2.8
CoFiNet	49.28	48.19	6.8	3.8
NgeNet	**54.10**	**49.4**	4.9	1.8
Ours w\o. sa	43.97	41.85	7.08	4.84
Ours w. sa	48.77	45.11	**10.92**	**6.42**

After analyzing the results, we can conclude that our algorithm is robust to noise and outperforms the other approaches in both situations. To demonstrate that adding the attention module improves performance, we tested our algorithm both with and without the newly developed self-attention module. The results show that the developed self-attention module improves performance significantly. Followed by us, is the NgeNet [11] algorithm, which performs well in both scenarios. Additionally, we can say that the 'sim2real' data processing approach works, but 'real2real' method is more relevant in the real life scenarios.

B. POSE ESTIMATION EVALUATION

We utilize the TUM RGB-D [55] benchmark dataset to test the expanded system using our suggested technique.

TABLE 6. Results of noisy training and noisy evaluation.

Noise type	Occlusion noise (m)		Gaussian noise	
Noise level	**0.3**	**0.5**	**0.05**	**0.08**
Feature Matching Recall (%)				
PREDATOR	94.18	93.08	43.89	24.78
CoFiNet	94.89	92.81	59.89	21.19
NgeNet	96.51	95.13	35.83	14.45
Ours w\o. sa	94.87	93.21	50.24	53.42
Ours w. sa	**97.45**	**96.23**	**76.77**	**57.91**
Inlier Ratio (%)				
PREDATOR	49.21	44.47	7.19	3.8
CoFiNet	49.81	47.88	6.94	4.2
NgeNet	**54.81**	**50.09**	7.59	2.84
Ours w\o. sa	48.95	43.88	14.89	11.76
Ours w. sa	49.19	46.04	**21.62**	**16.18**

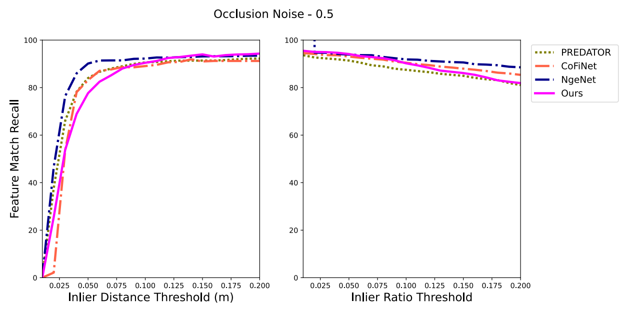


FIGURE 10. Feature matching recall in relation to inlier ratio threshold τ_2 and inlier distance threshold - occlusion noise radius = 0.5(m). Results of noiseless training and noisy evaluation.

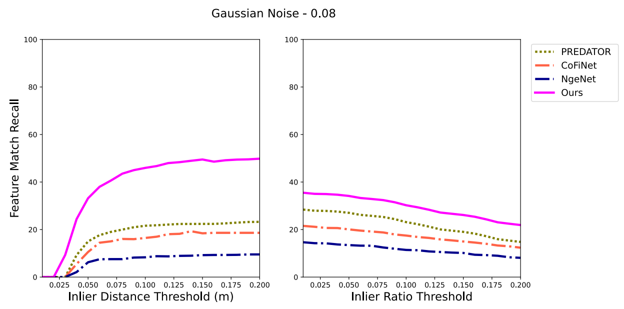


FIGURE 11. Feature matching recall in relation to inlier ratio threshold τ_2 and inlier distance threshold - Gaussian noise = 0.8. Results of noiseless training and noisy evaluation.

The data set contains a number of sequences that include RGB and depth frames captured by an RGB-D sensor, as well as the ground-truth sensor trajectory. We employ *fr1* data sequences recorded in a typical desk in an office environment. The sequences include challenges such as lighting changes, repeated structures, and translational motions along the primary axes.

1) EVALUATION METRICS

To evaluate performance, we use the benchmark's absolute trajectory error (ATE) and relative probability error (RPE) metrics. The ATE is a global consistency metric that calculates the absolute distances between the related poses of the trajectories to calculate the translational drift between the predicted trajectory and the ground truth. The RPE estimates the trajectory's local accuracy over a certain time interval.

2) EVALUATION FOR TUM RGB-D BENCHMARK

We performed two types of tests. In the first, we used the TUM RGB-D dataset without any noise addition, and in the second, we added Gaussian noise with a standard deviation of 0.05. Table 8 shows the results of the test from the perspective of RPE and ATE for noiseless data, while Table 9 shows the results for noisy data generated in a similar way as for previous noise test cases. Figures 12 provide a visual representation of the acquired results for our method.

3) RESULTS

In terms of results analysis, the first test scenario used noiseless data. In this case, NgeNet [11] outperforms other methods followed by PREDATOR [15]. Our method is ranked third, while the CofiNet [13] algorithm is ranked last. The situation is slightly different when Gaussian noise is added to the TUM RBG-D dataset. With this scenario, we want to simulate a'sim2real' environment. In this case, our method outperforms the other existing algorithms and shows that D3GATTEN is a robust method against noise for pose estimation. We achieved the best results in the translation error estimation, while for the rotation error our method proved to provide the best mean estimation.

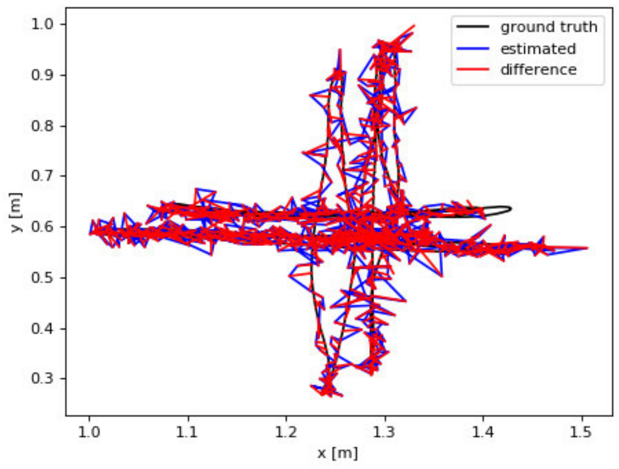


FIGURE 12. Relative pose error for D3GATTEN.

C. RUNTIME

In Table 7, we compare the runtime of our method on 3DMatch dataset with PREDATOR [15], CoFiNet [13], NgeNet [11], and D3Feat [16] to demonstrate its efficiency. Voxel size 2.5 cm and batch size 1 were set for each of the three approaches. The test run is on 4 x Nvidia A100 with Intel(R) Xeon(R) Gold 6226R @ 2.90GHZ x 16, 750GB RAM. The data load time was also taken into account

TABLE 7. Average run-time (ms) and VRAM (Kb) and per fragment pair in the 1623 number of 3DMatch test pairings.

Method	**Model VRAM (Kb)**	**Time (ms)**
PREDATOR	294160	**236**
CoFiNet	**159302**	243
NgeNet	224974	346
D3Feat	188942	238
Ours	190080	241

TABLE 8. Results for the TUM RGB-D benchmark in terms of RPE and ATE for noiseless data. Notation: TE (translation error) and RE (rotation error).

Relative Pose Error (RPE)				
Metrics	**Ours**	**CoFiNet**	**NgeNet**	**PREDATOR**
TE.RMSE (m)	0.033	0.038	**0.024**	0.027
TE.MEAN (m)	0.029	0.031	**0.019**	0.023
TE.STD (m)	0.014	0.018	**0.009**	0.011
RE.RMSE (deg)	1.430	2.182	**1.218**	1.349
RE.MEAN (deg)	1.296	1.846	**1.108**	1.152
RE.STD (deg)	0.660	0.951	**0.486**	0.597
Absolute Trajectory Error (ATE)				
RMSE (m)	0.022	0.029	**0.018**	0.020
MEAN (m)	0.020	0.027	**0.013**	0.016
MEDIAN (m)	0.018	0.023	**0.012**	0.014
STD (m)	0.010	0.018	**0.008**	0.009

TABLE 9. Results for the TUM RGB-D benchmark in terms of RPE and ATE for noisy data. Notation: TE (translation error) and RE (rotation error).

Relative Pose Error (RPE)				
Metrics	**Ours**	**CoFiNet**	**NgeNet**	**PREDATOR**
TE.RMSE (m)	**0.041**	0.052	0.044	0.048
TE.MEAN (m)	**0.038**	0.048	0.040	0.042
TE.STD (m)	0.027	0.038	**0.026**	0.031
RE.RMSE (deg)	**1.895**	2.857	1.954	2.148
RE.MEAN (deg)	**1.756**	2.406	1.795	2.017
RE.STD (deg)	**0.985**	1.325	1.003	1.129
Absolute Trajectory Error (ATE)				
RMSE (m)	**0.031**	0.041	0.037	0.039
MEAN (m)	0.027	0.034	**0.025**	0.016
MEDIAN (m)	**0.024**	0.031	0.025	0.027
STD (m)	**0.018**	0.024	0.021	0.024

while calculating the run-time. The best runtime is provided by PREDATOR, according to our analysis. With the help of this table, we want to show that the addition of the self-attention module did not result in a significantly longer runtime.

VI. CONCLUSION

In this work, we propose a novel self-attention-based keypoint feature extraction method for point clouds. With the adaptation of an efficient self-attention module to the D3Feat methods, we achieved a run-time-efficient and keypoint feature extraction method. The performance of the proposed method was tested on public datasets both for registration and pose estimation tasks. For the registration, the best performances were measured against the existing methods on a larger number of keypoints using FMR metrics and sets a new state of the art for the 3DMatch benchmark with 98.3% feature matching recall. For the pose estimation, the method obtained the best result in terms of translation errors.

With a run-time close to the best from the state-of-the-art, we aim in the future to validate the method in the SLAM context on embedded devices.

ACKNOWLEDGMENT

The authors thank Analog Devices for support, and NVidia for the DGX grade server offered as support for this work.

REFERENCES

[1] A. Geiger, J. Ziegler, and C. Stiller, "StereoScan: Dense 3D reconstruction in real-time," in *Proc. IEEE Intell. Vehicles Symp. (IV)*, Jun. 2011, pp. 963–968.

[2] R. F. Salas-Moreno, R. A. Newcombe, H. Strasdat, P. H. J. Kelly, and A. J. Davison, "SLAM++: Simultaneous localisation and mapping at the level of objects," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2013, pp. 1352–1359.

[3] Y. Cui, Z. Fang, J. Shan, Z. Gu, and S. Zhou, "3D object tracking with transformer," 2021, *arXiv:2110.14921*.

[4] X. Gu, Y. Wang, C. Wu, Y. J. Lee, and P. Wang, "HPLFlowNet: Hierarchical permutohedral lattice FlowNet for scene flow estimation on large-scale point clouds," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2019, pp. 3254–3263.

[5] A. Blaga, C. Militaru, A.-D. Mezei, and L. Tamas, "Augmented reality integration into MES for connected workers," *Robot. Comput.-Integr. Manuf.*, vol. 68, Apr. 2021, Art. no. 102057.

[6] R. Frohlich, L. Tamas, and Z. Kato, "Absolute pose estimation of central cameras using planar regions," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 43, no. 2, pp. 377–391, Feb. 2019.

[7] A. E. Johnson and M. Hebert, "Using spin images for efficient object recognition in cluttered 3D scenes," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 21, no. 5, pp. 433–449, May 1999.

[8] F. Tombari, S. Salti, and L. D. Stefano, "Unique signatures of histograms for local surface description," in *Proc. Eur. Conf. Comput. Vis.* Berlin, Germany: Springer, 2010, pp. 356–369.

[9] R. B. Rusu, N. Blodow, and M. Beetz, "Fast point feature histograms (FPFH) for 3D registration," in *Proc. IEEE Int. Conf. Robot. Autom.*, May 2009, pp. 3212–3217.

[10] S. Salti, F. Tombari, and L. Di Stefano, "SHOT: Unique signatures of histograms for surface and texture description," *Comput. Vis. Image Understand.*, vol. 125, pp. 251–264, Aug. 2014.

[11] L. Zhu, H. Guan, C. Lin, and R. Han, "Neighborhood-aware geometric encoding network for point cloud registration," 2022, *arXiv:2201.12094*.

[12] Y. Yang, W. Chen, M. Wang, D. Zhong, and S. Du, "Color point cloud registration based on supervoxel correspondence," *IEEE Access*, vol. 8, pp. 7362–7372, 2020.

[13] H. Yu, F. Li, M. Saleh, B. Busam, and S. Ilic, "CoFiNet: Reliable coarse-to-fine correspondences for robust point cloud registration," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 34, 2021, pp. 1–13.

[14] Z. Qin, H. Yu, C. Wang, Y. Guo, Y. Peng, and K. Xu, "Geometric transformer for fast and robust point cloud registration," 2022, *arXiv:2202.06688*.

[15] S. Huang, Z. Gojcic, M. Usvyatsov, A. Wieser, and K. Schindler, "PREDATOR: Registration of 3D point clouds with low overlap," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2021, pp. 4267–4276.

[16] X. Bai, Z. Luo, L. Zhou, H. Fu, L. Quan, and C.-L. Tai, "D3Feat: Joint learning of dense detection and description of 3D local features," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Jun. 2020, pp. 6359–6367.

[17] H. Deng, T. Birdal, and S. Ilic, "PPFNet: Global context aware local features for robust 3D point matching," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Jun. 2018, pp. 195–205.

[18] B. L. Zagar, E. Yurtsever, A. Peters, and A. C. Knoll, "Point cloud registration with object-centric alignment," *IEEE Access*, vol. 10, pp. 76586–76595, 2022.

[19] Y. Wang and J. Solomon, "Deep closest point: Learning representations for point cloud registration," in *Proc. IEEE/CVF Int. Conf. Comput. Vis. (ICCV)*, Oct. 2019, pp. 3523–3532.

[20] Y. Wang and J. M. Solomon, "PRNet: Self-supervised learning for partial-to-partial registration," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 32, 2019, pp. 1–13.

[21] Z. J. Yew and G. H. Lee, "RPM-Net: Robust point matching using learned features," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2020, pp. 11824–11833.

[22] K. Fu, S. Liu, X. Luo, and M. Wang, "Robust point cloud registration framework based on deep graph matching," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2021, pp. 8893–8902.

[23] P. J. Besl and N. D. McKay, "Sensor fusion IV: Control paradigms and data structures," in *Proc. SPIE*, vol. 1611, 1992, pp. 586–607.

[24] C. Choy, J. Park, and V. Koltun, "Fully convolutional geometric features," in *Proc. IEEE/CVF Int. Conf. Comput. Vis. (ICCV)*, Oct. 2019, pp. 8958–8966.

[25] Z. Gojcic, C. Zhou, J. D. Wegner, and A. Wieser, "The perfect match: 3D point cloud matching with smoothed densities," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2019, pp. 5545–5554.

[26] M. A. Fischler and R. Bolles, "Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography," *Commun. ACM*, vol. 24, no. 6, pp. 381–395, 1981.

[27] S. Ao, Q. Hu, B. Yang, A. Markham, and Y. Guo, "SpinNet: Learning a general surface descriptor for 3D point cloud registration," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2021, pp. 11753–11762.

[28] L. Tamas and B. Jensen, "Robustness analysis of 3D feature descriptors for object recognition using a time-of-flight camera," in *Proc. 22nd Medit. Conf. Control Autom.*, Jun. 2014, pp. 1020–1025.

[29] L. Tamas and L. Cosmin Goron, "3D semantic interpretation for robot perception inside office environments," *Eng. Appl. Artif. Intell.*, vol. 32, pp. 76–87, Jun. 2014.

[30] Y. Aoki, H. Goforth, R. A. Srivatsan, and S. Lucey, "PointNetLK: Robust & efficient point cloud registration using PointNet," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2019, pp. 7163–7172.

[31] H. Wang, Y. Liu, Z. Dong, and W. Wang, "You only hypothesize once: Point cloud registration with rotation-equivariant descriptors," 2021, *arXiv:2109.00182*.

[32] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 30, 2017, pp. 1–11.

[33] A. Zeng, S. Song, M. Niessner, M. Fisher, J. Xiao, and T. Funkhouser, "3DMatch: Learning local geometric descriptors from RGB-D reconstructions," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jul. 2017, pp. 199–208.

[34] A. Johnson, "Spin-images: A representation for 3-D surface matching," Ph.D. thesis, Robot. Inst., Carnegie Mellon Univ., Pittsburgh, PA, USA, Aug. 1997.

[35] A. Frome, D. Huber, R. Kolluri, T. Bülow, and J. Malik, "Recognizing objects in range data using regional point descriptors," in *Proc. Eur. Conf. Comput. Vis.* Berlin, Germany: Springer, pp. 224–237, 2004.

[36] Y. Guo, M. Bennamoun, F. Sohel, M. Lu, J. Wan, and N. M. Kwok, "A comprehensive performance evaluation of 3D local feature descriptors," *Int. J. Comput. Vis.*, vol. 116, no. 1, pp. 66–89, 2016.

[37] Z. Gojcic, C. Zhou, J. D. Wegner, L. J. Guibas, and T. Birdal, "Learning multiview 3D point cloud registration," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2020, pp. 1759–1769.

[38] H. Yang, J. Shi, and L. Carlone, "Teaser: Fast and certifiable point cloud registration," *IEEE Trans. Robot.*, vol. 37, no. 2, pp. 314–333, Apr. 2020.

[39] J. Serafin and G. Grisetti, "NICP: Dense normal based point cloud registration," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS)*, Sep. 2015, pp. 742–749.

[40] M. Khoury, Q.-Y. Zhou, and V. Koltun, "Learning compact geometric features," in *Proc. IEEE Int. Conf. Comput. Vis. (ICCV)*, Oct. 2017, pp. 153–161.

[41] H. Deng, T. Birdal, and S. Ilic, "PPF-FOLDNet: Unsupervised learning of rotation invariant 3D local descriptors," in *Proc. Eur. Conf. Comput. Vis. (ECCV)*, 2018, pp. 602–618.

[42] R. Q. Charles, H. Su, M. Kaichun, and L. J. Guibas, "PointNet: Deep learning on point sets for 3D classification and segmentation," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jul. 2017, pp. 652–660.

[43] Y. Yang, C. Feng, Y. Shen, and D. Tian, "FoldingNet: Point cloud auto-encoder via deep grid deformation," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Jun. 2018, pp. 206–215.

[44] H. Thomas, C. R. Qi, J.-E. Deschaud, B. Marcotegui, F. Goulette, and L. Guibas, "KPConv: Flexible and deformable convolution for point clouds," in *Proc. IEEE/CVF Int. Conf. Comput. Vis. (ICCV)*, Oct. 2019, pp. 6411–6420.

[45] W. Lu, G. Wan, Y. Zhou, X. Fu, P. Yuan, and S. Song, "DeepIcp: An end-to-end deep neural network for 3D point cloud registration," in *Proc. ICCV*, 2019, pp. 12–21.

[46] C. R. Qi, L. Yi, H. Su, and L. J. Guibas, "PointNet++: Deep hierarchical feature learning on point sets in a metric space," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 30, 2017, pp. 1–10.

[47] K. Cho, B. V. Merrienboer, D. Bahdanau, and Y. Bengio, "On the properties of neural machine translation: Encoder–decoder approaches," 2014, *arXiv:1409.1259*.

[48] N. Parmar, A. Vaswani, J. Uszkoreit, L. Kaiser, N. Shazeer, A. Ku, and D. Tran, "Image transformer," in *Proc. Int. Conf. Mach. Learn.*, 2018, pp. 4055–4064.

[49] L. Zhou, Y. Zhou, J. J. Corso, R. Socher, and C. Xiong, "End-to-end dense video captioning with masked transformer," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Jun. 2018, pp. 8739–8748.

[50] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," 2015, *arXiv:1512.03385*.

[51] M. Dusmanu, I. Rocco, T. Pajdla, M. Pollefeys, J. Sivic, A. Torii, and T. Sattler, "D2-net: A trainable CNN for joint detection and description of local features," 2019, *arXiv:1905.03561*.

[52] H. Zhang, I. Goodfellow, D. Metaxas, and A. Odena, "Self-attention generative adversarial networks," 2018, *arXiv:1805.08318*.

[53] I. Igbinedion, D. Phillips, and D. Lévy, "Fast softmax sampling for deep neural networks," Stanford Univ., Stanford, CA, USA, Tech. Rep. 130, 2017.

[54] N. Ketkar and J. Moolayil, "Introduction to Pytorch," in *Deep Learning With Python*. Berkeley, CA, USA: Springer, 2021, pp. 27–91.

[55] J. Sturm, N. Engelhard, F. Endres, W. Burgard, and D. Cremers, "A benchmark for the evaluation of RGB-D SLAM systems," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, Oct. 2012, pp. 573–580.

[56] H. Deng, T. Birdal, and S. Ilic, "3D local features for direct pairwise registration," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2019, pp. 3244–3253.

[57] Y. Zhao, T. Birdal, H. Deng, and F. Tombari, "3D point capsule networks," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2019, pp. 1009–1018.

BENJAMIN KELENYI received the M.Sc. degree in computer science from the Technical University of Cluj-Napoca, Romania, in 2020, where he is currently pursuing the Ph.D. degree in artificial intelligence. He proved to be a highly skilled Software Engineer, realizing efficient implementations on different edge GPU devices, including Jetson family products. During his studies, he has already shown great potential in robotics and AI. He is engaged in several national and international projects. His research interest includes optimization of deep learning algorithms for edge GPU devices in the context of robotic perception.

LEVENTE TAMAS (Member, IEEE) received the M.Sc. (valedictorian) and Ph.D. degrees in electrical engineering from the Technical University of Cluj-Napoca, Romania, in 2005 and 2010, respectively. He took part in several postdoctoral programs dealing with 3D perception and robotics, the most recent one spent with the Bern University of Applied Sciences, Switzerland. He is currently with the Robotics Research Group, Department of Automation, Technical University of Cluj-Napoca. His current research interests include 3D perception and planning for autonomous systems. He is a member of the IEEE Robotics and Automation Society.