PCAM: Product of Cross-Attention Matrices for Rigid Registration of Point Clouds

Anh-Quan Cao[2,*] Gilles Puy[1] Alexandre Boulch[1] Renaud Marlet[1,3]

[1]Valeo.ai, Paris, France [2]Inria, Paris, France[†]
[3]LIGM, Ecole des Ponts, Univ Gustave Eiffel, CNRS, Marne-la-Vallée, France

Abstract

Rigid registration of point clouds with partial overlaps is a longstanding problem usually solved in two steps: (a) finding correspondences between the point clouds; (b) filtering these correspondences to keep only the most reliable ones to estimate the transformation. Recently, several deep nets have been proposed to solve these steps jointly. We built upon these works and propose PCAM: a neural network whose key element is a pointwise product of cross-attention matrices that permits to mix both low-level geometric and high-level contextual information to find point correspondences. These cross-attention matrices also permits the exchange of context information between the point clouds, at each layer, allowing the network construct better matching features within the overlapping regions. The experiments show that PCAM achieves state-of-the-art results among methods which, like us, solve steps (a) and (b) jointly via deepnets.

1. Introduction

Point cloud registration is the problem of estimating the rigid transformation that aligns two point clouds. It has many applications in various domains such as autonomous driving, motion and pose estimation, 3D reconstruction, simultaneous localisation and mapping (SLAM), and augmented reality. The most famous method to solve this task is ICP [3], for which several improvements have been proposed, modifying the original optimisation process [40, 45] or using geometric feature descriptors [32] to match points.

Recently, end-to-end learning-based methods combining point feature extraction, point matching, and point-pairs filtering, have been developed to solve this task. Deep Closest Point (DCP) [38], improved by PRNet [37], finds point correspondences via an attention matrix and estimate the transformation by solving a least-squares problem. Deep Global Registration (DGR) [7] tackles partial-to-partial point cloud registration by finding corresponding points via deep features, computing confidence scores for these pairs, and solving a weighted least-squares problem to estimate the transformation. IDAM [23] considers all possible pairs of points in a similarity matrix computed using deep or hand-crafted features, and proposes a learned two-step filter to select only relevant pairs for transformation estimation. Finally, [9, 12, 41] use deep networks and the Sinkhorn algorithm to find and filter corresponding points.

Our observation is that, when matching points between point clouds, one would like to find correspondences using both local fine geometric information, to precisely select the best corresponding point, and high-level contextual information, to differentiate between points with similar local geometry but from different parts of the scene. The fine geometric information is naturally extracted at the first layers of a deep convnet, while the context information is found in the deepest layers. The existing deep registration methods estimate the correspondences between point clouds using features extracted at the deepest layers. Therefore, there is no explicit control on the amount of fine geometric and high-level context information encoded in these features.

Instead, we propose to compute point correspondences at every layer of our deep network via cross-attention matrices, and to combine these matrices via a point-wise multiplication. This simple yet very effective solution naturally ensures that both low-level geometric and high-level context information are exploited when matching points. It also permits to remove spurious matches found only at one scale. Furthermore, we also exploit these cross-attention matrices to exchange information between the point clouds at each layer, allowing the network to exploit context information from both point clouds to find the best matching point within the overlapping regions.

The design of our method is inspired by DGR [7], DCP [38] and PRNet [37]. We exploit cross-attention matrices to exchange context information between point clouds, like in

[*]Most of the work was done during an internship at valeo.ai in 2020.
[†]Inria, Mines ParisTech, PSL Research University.

[38, 37, 12], to find the best matches within overlapping region. Our main contribution is to propose to compute such cross-attention matrices at every network layer and to combine them to exploit both fine and high-level information when matching points. Our second contribution is to show that our method achieves state-of-the-art results on two real datasets (indoor, outdoor) and on a synthetic one.

2. Related Work

Optimisation-based methods. Iterative Closest Point (ICP) [6, 3, 44] is the most known algorithm for point cloud registration. It takes in two point clouds and alternate between point matching via nearest-neighbour search and transformation estimation by solving a least-squares problem. Several improvements of ICP's steps have been introduced to improve speed, solve the discretisation problem, become robust to outliers, or incorporate confidence scores on the correspondences [1, 4, 28, 30, 33]. ICP often converges to a local minimum due to the non-convexity of the objective function. Therefore, some works propose solutions to widen the basin of attraction of the global optimum [13, 36], use genetic algorithms [34], or find a good crude alignment to initialise ICP [26]. Another line of works concerns algorithms with theoretical convergence guarantees to a global optimum thanks to, *e.g.*, the combination of ICP and a branch-and-bound technique [40], via convex relaxations [27, 29], or thanks to mixed-integer programming [19]. While theoretically appealing, these approaches usually suffers from a high computational complexity.

Feature-based point matching. Instead of relying solely on point coordinates to find correspondences between points, several approaches have proposed to extract point feature descriptors by analysing the local, and possibly global, 3D geometry. Classical methods use hand-crafted features such as [20, 31, 32, 35] and, more recently, features extracted thanks to deep networks that take as inputs either hand-crafted local features [10, 11, 16, 18, 21, 43], or directly point cloud coordinates [8, 18]. These methods use these point features to establish correspondences and rely on RANSAC to estimate the transformation, possibly using a pre-filtering of the points unlikely to provide good correspondences [18].

End-to-end learning. Another type of approaches consists in training a network that will establish correspondences between the points of both point clouds, filter these correspondences to keep only the reliable pairs, and estimate the transformation based on the best pairs of points. Several methods fall in this category, such as [7, 9, 12, 23, 37, 38, 41] which we described in the introduction. In addition, let us also mention PointNetLK [2] that extracts a global feature vector per point cloud and unroll the Lucas-Kanade algorithm [25] to find the best transformation aligning the point clouds in feature space, DeepGMR [42] that maps each point cloud to a parametric probability distribution via a neural network and estimates the transformation by minimising the KL-divergence between distributions. [22] proposes a probabilistic registration method using deep features and learned attention weights. Finally, [15] proposes a method for multiview registration by jointly training a pairwise registration module and a global refinement module.

3. Network Architecture

3.1. Problem Statement

We consider the problem of partial-to-partial rigid registration between two point clouds $\mathcal{P} = \{\boldsymbol{p}_1, \dots, \boldsymbol{p}_N\}$ and $\mathcal{Q} = \{\boldsymbol{q}_1, \dots, \boldsymbol{q}_M\}$, with $\boldsymbol{p}_i, \boldsymbol{q}_j \in \mathbb{R}^3$. The point clouds $\mathcal{P}$ and $\mathcal{Q}$ represent two views of the same scene with partial overlap. Hence, only a subset of the points in $\mathcal{P}$ can have matching points in $\mathcal{Q}$, and vice versa from $\mathcal{Q}$ to $\mathcal{P}$. Let $\mathcal{P}_{\text{v}}$ (resp. $\mathcal{Q}_{\text{v}}$) be the subset of points in $\mathcal{P}$ (resp. $\mathcal{Q}$) visible in $\mathcal{Q}$ (resp. $\mathcal{P}$). Our goal is to estimate the rotation matrix $\mathsf{R}_{\text{gt}} \in \text{SO}(3)$ and translation vector $\boldsymbol{t}_{\text{gt}} \in \mathbb{R}^3$ that aligns $\mathcal{P}_{\text{v}}$ on $\mathcal{Q}_{\text{v}}$. This transformation can be estimated by solving

$$\min_{\mathsf{R},\mathsf{t}} \sum_{\boldsymbol{p} \in \mathcal{P}_{\text{v}}} \|\mathsf{R}\,\boldsymbol{p} + \boldsymbol{t} - m_{\mathcal{Q}}(\boldsymbol{p})\|_2^2 \,, \tag{1}$$

where $m_{\mathcal{Q}} : \mathcal{P}_{\text{v}} \to \mathcal{Q}_{\text{v}}$ maps any $\boldsymbol{p} \in \mathcal{P}_{\text{v}}$ to its best matching point $\boldsymbol{q} \in \mathcal{Q}_{\text{v}}$. Note that we can also estimate the inverse transformation from $\mathcal{Q}$ to $\mathcal{P}$ with the map $m_{\mathcal{P}} : \mathcal{Q}_{\text{v}} \to \mathcal{P}_{\text{v}}$.

Problem (1) can be solved via an SVD decomposition [3, 17]. The challenging tasks are the estimations of the subset of points $\mathcal{P}_{\text{v}}$ and $\mathcal{Q}_{\text{v}}$, and of the mappings $m_{\mathcal{Q}}(\cdot)$ and $m_{\mathcal{P}}(\cdot)$, given only $\mathcal{P}$ and $\mathcal{Q}$ as inputs.

3.2. Method Overview

Our method is composed of two classical modules: point matching and point-pair filtering (see Figure 1). The point-matching module, denoted by $g(\cdot, \cdot)$, permits us to estimate two maps $\tilde{m}_{\mathcal{Q}} : \mathcal{P} \to \mathcal{Q}$ and $\tilde{m}_{\mathcal{P}} : \mathcal{Q} \to \mathcal{P}$ that provide pairs of corresponding points between $\mathcal{P}$ and $\mathcal{Q}$ (even in non-overlapping regions). The point-pair filtering module, denoted by $h(\cdot, \cdot)$, provides a confidence score to all pairs $(\boldsymbol{p}_i, \tilde{m}_{\mathcal{Q}}(\boldsymbol{p}_i))_{1 \leqslant i \leqslant N}$, $(\boldsymbol{q}_j, \tilde{m}_{\mathcal{P}}(\boldsymbol{q}_j))_{1 \leqslant j \leqslant M}$, hence allowing detection of valid pairs in overlapping regions.

Our main contribution is in g, where we propose the construction of cross-attention matrices at each layer of g and their combination to obtain point correspondences. These successive cross-attention matrices are obtained using point features with an increasing field of view, or scale, thus establishing point correspondences with increasing scene context. We obtain $\tilde{m}_{\mathcal{Q}}$ and $\tilde{m}_{\mathcal{P}}$ by combining all these attention matrices via pointwise matrix multiplications. Within overlapping regions, it permits us to improve the quality of the maps $\tilde{m}_{\mathcal{Q}}$ and $\tilde{m}_{\mathcal{P}}$ by filtering out the matches that are

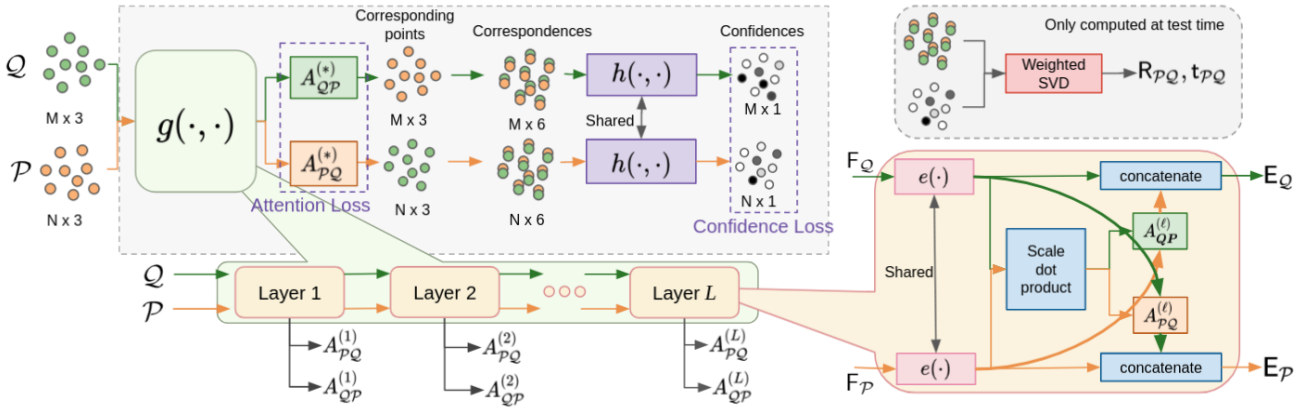

Figure 1. **Overview of our architecture.** $\mathcal{P}$ and $\mathcal{Q}$ enter in the point matching module $g(\cdot)$ that permits the extraction of attention matrices used to compute one matching point for each input point. Each pair of matching points is then given a confidence score via $h(\cdot)$, which permits the estimation of the rigid transformation by solving a weighted least-squares problem via SVD.

inconsistent across scales. We also use these cross-attention matrices to exchange information between $\mathcal{P}$ and $\mathcal{Q}$: each point in $\mathcal{P}$ receives, in addition to its own feature, the feature of the best corresponding point in $\mathcal{Q}$ (and vice-versa) before computing the point feature at the next layer. This process favours the discovery or rejection of correspondences at the next layer depending on the level of similarity between corresponding points at the current layer.

3.3. First Module: Point Matching

Our point matching module, denoted by $g : (\mathbb{R}^{N\times 3}, \mathbb{R}^{M\times 3}) \to (\mathbb{R}^{N\times M}, \mathbb{R}^{N\times M})$, takes two unaligned point clouds $\mathcal{P}$ and $\mathcal{Q}$ as input, and produces two global attention matrices $\mathsf{A}^{(*)}_{\mathcal{PQ}}$ and $\mathsf{A}^{(*)}_{\mathcal{QP}}$ which inform us on the correspondences between point clouds. These global attention matrices are constructed (see Sec. 3.3.2) from L attention matrices, $\mathsf{A}^{(\ell)}_{\mathcal{PQ}}$ and $\mathsf{A}^{(\ell)}_{\mathcal{QP}}$, obtained at layers $\ell = 1, \dots, L$ of our deep network (see Sec. 3.3.1).

3.3.1 Layer-wise Cross-Attention Matrices $\mathsf{A}^{(\ell)}_{\mathcal{PQ}}$, $\mathsf{A}^{(\ell)}_{\mathcal{QP}}$

Let $\mathsf{F}_{\mathcal{P}} \in \mathbb{R}^{N\times c^{(\ell-1)}}$ and $\mathsf{F}_{\mathcal{Q}} \in \mathbb{R}^{M\times c^{(\ell-1)}}$ be the features on $\mathcal{P}$ and $\mathcal{Q}$, respectively, at the input of layer ℓ, where $c^{(\ell-1)}$ is the size of the feature vectors at the output of layer $\ell-1$. Each point is thus described by a $c^{(\ell-1)}$-dimensional feature vector. These features enter the ℓ^{th} point convnet $e : \mathbb{R}^{\cdot\times c^{(\ell-1)}} \to \mathbb{R}^{\cdot\times c^{(\ell)}/2}$ to extract new features of size $c^{(\ell)}/2$ for each input point, to be doubled after concatenation of the best corresponding point feature in the other point cloud (see below). This encoder extracts local geometry information around each point in the point cloud thanks to three residual blocks, each containing two FKAConv [5] sub-layers. The supplementary material details $e(\cdot)$.

The new feature vectors $e(\mathsf{F}_{\mathcal{P}})$ and $e(\mathsf{F}_{\mathcal{Q}})$ are used to compute two attention matrices $\mathsf{A}^{(\ell)}_{\mathcal{PQ}}, \mathsf{A}^{(\ell)}_{\mathcal{QP}} \in \mathbb{R}^{N\times M}$ with

$$(\mathsf{A}^{(\ell)}_{\mathcal{PQ}})_{ij} = \frac{\mathrm{e}^{\,a_{ij}/s}}{\sum_{k=1}^{M} \mathrm{e}^{\,a_{ik}/s}}, \; (\mathsf{A}^{(\ell)}_{\mathcal{QP}})_{ij} = \frac{\mathrm{e}^{\,a_{ij}/s}}{\sum_{k=1}^{N} \mathrm{e}^{\,a_{kj}/s}}, \tag{2}$$

where $s > 0$ is the softmax temperature and

$$a_{ij} = \frac{e(\mathsf{F}_{\mathcal{P}})_i \; e(\mathsf{F}_{\mathcal{Q}})_j^{\mathsf{T}}}{\|e(\mathsf{F}_{\mathcal{P}})_i\|_2 \|e(\mathsf{F}_{\mathcal{Q}})_j\|_2}. \tag{3}$$

$\mathsf{A}_{\mathcal{PQ}}, \mathsf{A}_{\mathcal{QP}}$ differ in the normalisation dimension: $\mathsf{A}_{\mathcal{PQ}}$ transfers information from $\mathcal{Q}$ to $\mathcal{P}$, $\mathsf{A}_{\mathcal{QP}}$ from $\mathcal{P}$ to $\mathcal{Q}$.

Finally, we transfer information between point clouds by computing $\mathsf{E}_{\mathcal{P}} = [e(\mathsf{F}_{\mathcal{P}}), \; \mathsf{A}^{(\ell)}_{\mathcal{PQ}}\, e(\mathsf{F}_{\mathcal{Q}})] \in \mathbb{R}^{N\times c^{(\ell)}}$, $\mathsf{E}_{\mathcal{Q}} = [e(\mathsf{F}_{\mathcal{Q}}), \; \mathsf{A}^{(\ell)\,\mathsf{T}}_{\mathcal{QP}}\, e(\mathsf{F}_{\mathcal{P}})] \in \mathbb{R}^{M\times c^{(\ell)}}$. $\mathsf{E}_{\mathcal{P}}, \mathsf{E}_{\mathcal{Q}}$ are the output features of layer ℓ and input features of layer $(\ell+1)$. At the end of the process, we have extracted two sets of L attention matrices: $(\mathsf{A}^{(1)}_{\mathcal{PQ}}, \dots, \mathsf{A}^{(L)}_{\mathcal{PQ}})$ and $(\mathsf{A}^{(1)}_{\mathcal{QP}}, \dots, \mathsf{A}^{(L)}_{\mathcal{QP}})$.

3.3.2 Global Attention Matrices $\mathsf{A}^{(*)}_{\mathcal{PQ}}$, $\mathsf{A}^{(*)}_{\mathcal{QP}}$

We combine the attention matrices of each layer ℓ via a simple pointwise multiplication, denoted by $\odot$, to obtain

$$\mathsf{A}^{(*)}_{\mathcal{PQ}} = A^{(1)}_{\mathcal{PQ}} \odot \dots \odot A^{(L)}_{\mathcal{PQ}}. \tag{4}$$

$\mathsf{A}^{(*)}_{\mathcal{QP}}$ is defined similarly. The motivation for this strategy is that the successive attention matrices are constructed using features with an increasing field of view or scale, and we want to match points only if their features are similar at *all* scales, hence the entry-wise multiplication of the attention coefficients. Another motivation is to permit to backpropagate gradients directly from the loss to each layer ℓ.

3.3.3 Soft and Sparse Maps

There exists two standards ways to match points once the global attention matrices are available: soft and sparse mappings. We explore the performance of both approaches in this work. The soft map $\tilde{m}_{\mathcal{Q}}(\cdot)$ is defined as

$$\tilde{m}_{\mathcal{Q}}(\boldsymbol{p}_i) = \frac{\sum_{j=1}^{M} (\mathsf{A}^{(*)}_{\mathcal{PQ}})_{i,j}\ \boldsymbol{q}_j}{\sum_{k=1}^{M} (\mathsf{A}^{(*)}_{\mathcal{PQ}})_{i,k}} \tag{5}$$

for points $(\boldsymbol{p}_i)_{1 \leqslant i \leqslant N}$. It maps $\mathcal{P}$ to $\mathbb{R}^3$ rather than $\mathcal{Q}$. The sparse map $\tilde{m}_{\mathcal{Q}}(\cdot)$ is defined as

$$\tilde{m}_{\mathcal{Q}}(\boldsymbol{p}_i) = \boldsymbol{q}_{j^*},\ \text{where } j^* = \operatorname*{argmax}_j (\mathsf{A}^{(*)}_{\mathcal{PQ}})_{ij}. \tag{6}$$

It maps $\mathcal{P}$ to $\mathcal{Q}$, but it is not differentiable and it prevents backpropagation from the confidence network to the point-matching network. Yet, the point-matching network will still be trained using a cross-entropy loss on the attention matrices (Sec. 3.6). The mapping $\tilde{m}_{\mathcal{P}}$ from $\mathcal{Q}$ to $\mathcal{P}$ or $\mathbb{R}^3$ is constructed similarly using $\mathsf{A}^{(*)}_{\mathcal{QP}}$.

3.4. Second Module: Confidence Estimation

The module in Sec. 3.3 produces pairs of matching points $(\boldsymbol{p}_i, \tilde{m}_{\mathcal{Q}}(\boldsymbol{p}_i))$ and $(\boldsymbol{q}_j, \tilde{m}_{\mathcal{P}}(\boldsymbol{q}_j))$ for all points in $\mathcal{P}$ and $\mathcal{Q}$. However, as we are tackling partial-to-partial registration, only subsets of $\mathcal{P}$ and $\mathcal{Q}$ match to one another. Hence, there are many incorrect matches which need to be filtered out. We detect these incorrect matches by concatenating each pair in a vector $[\boldsymbol{p}_i^\intercal, \tilde{m}_{\mathcal{Q}}(\boldsymbol{p}_i)^\intercal]^\intercal \in \mathbb{R}^6$, and then pass these N pairs into the confidence estimation module $h : \mathbb{R}^{\cdot \times 6} \to (0,1)^{\cdot \times 1}$ which outputs a confidence score $w_{\boldsymbol{p}_i}$ for each input pair. The same module h is used for pairs $(\boldsymbol{q}_j, \tilde{m}_{\mathcal{P}}(\boldsymbol{q}_j))$ and yields M corresponding scores $w_{\boldsymbol{q}_j}$. The confidence estimator is a point convnet with 9 residual blocks, each containing 2 FKAConv sub-layers. The confidence score are thus estimated using context information provided by the point convolution on $\mathcal{P}$ (or $\mathcal{Q}$). The detailed network architecture is in the supplementary material.

3.5. Transformation Estimation

We estimate the rotation and translation from $\mathcal{P}$ to $\mathcal{Q}$ by solving a weighted least-squares problem, using the weights $w_{\boldsymbol{p}_i}$ given by the confidence estimator:

$$(\mathsf{R}_{\text{est}}, \boldsymbol{t}_{\text{est}}) = \operatorname*{argmin}_{\mathsf{R},\boldsymbol{t}} \sum_{i=1}^{N} \phi(w_{\boldsymbol{p}_i}) \left\| \mathsf{R}\boldsymbol{p}_i + \boldsymbol{t} - \tilde{m}_{\mathcal{Q}}(\boldsymbol{p}_i) \right\|_2^2. \tag{7}$$

The function $\phi : \mathbb{R} \to \mathbb{R}$ applies hard-thresholding on the confidence scores by setting to zero the less confident ones. The above problem can be solved with an SVD [17, 38].

3.6. Losses

The networks $g(\cdot,\cdot)$ and $h(\cdot,\cdot)$ and trained under supervision using a first loss, denoted by L^{ca} that applies on the attention matrices $\mathsf{A}^{(*)}_{\mathcal{PQ}}, \mathsf{A}^{(*)}_{\mathcal{QP}}$ and the sum of other losses, L^{cc} and L^{gc}, that apply on the confidence scores $w_{\boldsymbol{p}_i}, w_{\boldsymbol{q}_j}$. The complete training loss satisfies $L^{\text{ca}} + L^{\text{cc}} + L^{\text{gc}}$.

3.6.1 Loss on the Attention Matrices

The role of the attention matrices $\mathsf{A}^{(*)}_{\mathcal{PQ}}, \mathsf{A}^{(*)}_{\mathcal{QP}}$ is to identify mappings between $\mathcal{P}$ and $\mathcal{Q}$ for the final registration task. The pairs of points identified at this stage should include, as a subset, the ideal pairs encoded by $m_{\mathcal{P}}$ and $m_{\mathcal{Q}}$. Identifying this subset is the role of the confidence estimator $h(\cdot,\cdot)$.

The ideal map $m_{\mathcal{P}}$ and $m_{\mathcal{Q}}$ are estimated as follows. For each point $\boldsymbol{p}_i$, we search the closest point $\boldsymbol{q}_{j^*(i)}$ to $\mathsf{R}_{\text{gt}}\boldsymbol{p}_i + \boldsymbol{t}_{\text{gt}}$ in $\mathcal{Q}$ and consider the pair $(\boldsymbol{p}_i, \boldsymbol{q}_{j^*(i)})$ as valid if $\mathsf{R}_{\text{gt}}\boldsymbol{p}_i + \boldsymbol{t}_{\text{gt}}$ is also the closest point to $\boldsymbol{q}_{j^*(i)}$ in $\{\mathsf{R}_{\text{gt}}\boldsymbol{p}_u + \boldsymbol{t}_{\text{gt}}\}_u$. More precisely, let us define $j^*(u) = \operatorname{argmin}_j \left\| \mathsf{R}_{\text{gt}}\boldsymbol{p}_u + \boldsymbol{t}_{\text{gt}} - \boldsymbol{q}_j \right\|_2$ and $i^*(v) = \operatorname{argmin}_v \left\| \mathsf{R}_{\text{gt}}\boldsymbol{p}_i + \boldsymbol{t}_{\text{gt}} - \boldsymbol{q}_v \right\|_2$, The ideal maps satisfy $m_{\mathcal{Q}}(\boldsymbol{p}_u) = \boldsymbol{q}_v,\ m_{\mathcal{P}}(\boldsymbol{q}_v) = \boldsymbol{p}_u,\ \forall (u,v) \in \mathcal{C}$, where $\mathcal{C} = \{(u,v) \mid j^*(u) = v, u = i^*(v)\}$. Only a subset of the points are in $\mathcal{C}$, even in overlapping regions. For convenience, we also define the set of points in $\mathcal{P}$ for which a corresponding point exists in $\mathcal{Q}$ by $\mathcal{C}_{\mathcal{P}} = \{u \mid \exists\, v \in [\![M]\!] \text{ s.t. } (u,v) \in \mathcal{C}\}$. The set $\mathcal{C}_{\mathcal{Q}}$ is defined similarly for the inverse mapping.

We consider a contrastive classification loss, such as used in [7, 23, 9], denoted by $L^{\text{ca}} = L^{\text{ca}}_{\mathcal{PQ}} + L^{\text{ca}}_{\mathcal{QP}}$. $L^{\text{ca}}_{\mathcal{PQ}}$ enforces a good mapping from $\mathcal{P}$ to $\mathcal{Q}$ and satisfies

$$\begin{aligned} L^{\text{ca}}_{\mathcal{PQ}} &= -\frac{1}{N} \sum_{(u,v)\in\mathcal{C}} \log \left[(\mathsf{A}^{(*)}_{\mathcal{PQ}})_{uv} \right] \\ &= -\frac{1}{N} \sum_{\ell=1}^{L} \sum_{(u,v)\in\mathcal{C}} \log \left[\frac{\mathrm{e}^{\,(a^{(\ell)}_{uv}/s)}}{\sum_{j=1}^{M} \mathrm{e}^{\,(a^{(\ell)}_{uj}/s)}} \right]. \end{aligned} \tag{8}$$

$L^{\text{ca}}_{\mathcal{QP}}$ is defined likewise. All points are constrained in (8) thanks to the softmax involved in the attention matrices. In cases where $\mathcal{C}$ contains only few points, rather than augmenting $\mathcal{C}$ with pairs of worse quality, we leave $\mathcal{C}$ untouched to force the point matching network to identify the most reliable pairs (along with bad ones) and let the second module learn how to filter the bad pairs.

3.6.2 Losses on the Confidence Scores

To train the confidence estimator $h(\cdot,\cdot)$, we consider classification and geometric losses.

The classification losses are built by measuring the quality of the maps $\tilde{m}_{\mathcal{Q}}, \tilde{m}_{\mathcal{P}}$ at each input point. If a mapped point $\tilde{m}_{\mathcal{Q}}(\boldsymbol{p}_i)$ is close to the ideal target point $\mathsf{R}_{\text{gt}}\boldsymbol{p}_i + \boldsymbol{t}_{\text{gt}}$, then the pair $(\boldsymbol{p}_i, \tilde{m}_{\mathcal{Q}}(\boldsymbol{p}_i))$ is considered accurate and $w_{\boldsymbol{p}_i}$

should be close to 1, and 0 otherwise. The classification loss thus satisfies $L^{\text{cc}} = L^{\text{cc}}_{\mathcal{PQ}} + L^{\text{cc}}_{\mathcal{QP}}$, where

$$L^{\text{cc}}_{\mathcal{PQ}} = -\left(\frac{1}{N}\sum_i y_{\boldsymbol{p}_i}\log(w_{\boldsymbol{p}_i}) + (1-y_{\boldsymbol{p}_i})\log(1-w_{\boldsymbol{p}_i})\right),$$

$$y_{\boldsymbol{p}_i} = \begin{cases} 1, & \text{if } \|\mathsf{R}_{\text{gt}}\boldsymbol{p_i} + \boldsymbol{t}_{\text{gt}} - \tilde{m}_{\mathcal{Q}}(\boldsymbol{p}_i)\|_2 \leqslant \kappa, \\ 0, & \text{otherwise,} \end{cases} \tag{9}$$

and $\kappa > 0$ is a threshold to decide if a pair of points is accurate or not. The second loss $L^{\text{cc}}_{\mathcal{QP}}$ is defined similarly by considering the inverse mapping.

The threshold $\kappa > 0$ to decide pair accuracy is somewhat arbitrary. To mitigate the effect of this choice, we also consider the geometric loss $L^{\text{gc}} = L^{\text{gc}}_{\mathcal{PQ}} + L^{\text{gc}}_{\mathcal{QP}}$ where

$$L^{\text{gc}}_{\mathcal{PQ}} = \sum_i \frac{w_{\boldsymbol{p}_i}}{N}\,\|\mathsf{R}_{\text{gt}}\boldsymbol{p_i} + \boldsymbol{t}_{\text{gt}} - \tilde{m}_{\mathcal{Q}}(\boldsymbol{p}_i)\|_2. \tag{10}$$

$L^{\text{gc}}_{\mathcal{QP}}$ is defined using $w_{\boldsymbol{q}_j}$ and the inverse transformation. Whenever the distance between a mapped point and its ideal location is large, then $w_{\boldsymbol{p}_i}$ should be small. On the contrary, when this distance is small, then $w_{\boldsymbol{p}_i}$ can be large. Note that this geometric losses can be used only in combination with the classification losses L^{cc} as otherwise $w_{\boldsymbol{p}_i}, w_{\boldsymbol{q}_j} = 0$ is a trivial useless solution.

4. Experiments

4.1. Datasets and Training Parameters

We evaluate our method on real (indoor, outdoor) and synthetic datasets. The indoor dataset is 3DMatch [43]. We use the standard train/test splits and the procedure of [7, 10, 11, 8] to generate pairs of scans with at least 30% of overlap for training and testing. During training, as in [7], we apply data augmentation using random rotations in $[0°, 360°)$ around a random axis, and random scalings in $[0.8, 1.2]$. For the experiments on outdoor data, we use the KITTI odometry dataset [14] and follow the same protocol as [7]: GPS-IMU is used to create pairs of scans that are at least 10m apart; the ground-truth transformation is computed using GPS followed by ICP. Unlike in [7], we do not use data augmentation during training on KITTI. For synthetic data, we use ModelNet40 [39] and follow the setup of [37] to simulate partial registration problems.

All models are trained using AdamW [24], with a weight decay of 0.001, a batch size of 1, and a learning rate of 0.001. On 3Dmatch and KITTI, the models are trained for 100 epochs with a learning rate divided by 10 after 60 and 80 epochs. On ModelNet40, it is sufficient to train the models for 10 epochs (with a learning rate divided by 10 after 6 and 8 epochs) to observe convergence. All results are reported using the models obtained at the last epoch. The temperature s in (2) is set to $s = 0.03$.

	ϕ	Opt.	Saf.	TE_{all}	RE_{all}	Recall	TE	RE
FGR [45]						42.7	0.11	4.08
RANSAC [31]						74.9	0.09	2.92
DCP [38]						3.2	0.21	8.42
PointNetLK [2]						1.6	0.21	8.04
DGR [7]				0.47	17.4	73.9	0.09	2.97
DGR [7]	✓			0.38	13.4	81.5	0.08	2.56
DGR [7]	✓	✓		0.33	11.8	86.6	**0.07**	2.34
DGR [7]	✓	✓	✓	0.25	9.5	91.2	**0.07**	2.42
PCAM-Sparse				0.44	16.3	74.9	0.08	2.98
PCAM-Sparse	✓			0.35	13.0	83.8	**0.07**	2.43
PCAM-Sparse	✓	✓		0.32	11.8	87.0	**0.07**	**2.11**
PCAM-Sparse	✓	✓	✓	**0.23**	**8.9**	**92.4**	**0.07**	2.16
PCAM-Soft				0.46	17.5	74.7	0.09	3.01
PCAM-Soft	✓			0.37	13.1	81.4	0.08	2.50
PCAM-Soft	✓	✓		0.33	11.9	85.6	**0.07**	2.12
PCAM-Soft	✓	✓	✓	0.24	9.8	91.3	**0.07**	2.25

Table 1. **Results on 3DMatch test split**. The recall is computed using 0.3 m and $15°$ as thresholds. The scores of all variants of DGR are obtained using the official implementation of DGR [7], and those of the other concurrent methods are reported from [7]. The columns 'ϕ', 'Opt.', and 'Saf.' indicate respectively whether hard-thresholding on the confidence scores, the pose refinement of [7], and the safeguard registration of [7], are used or not.

4.2. Metrics

The performance on 3DMatch and KITTI is measured using the metrics of [7]: the translation error (TE) and rotation error (RE) are defined as $\text{TE}(\boldsymbol{t}_{\text{est}}) = ||\boldsymbol{t}_{\text{est}} - \boldsymbol{t}_{\text{gt}}||_2$ and $\text{RE}(\mathsf{R}_{\text{est}}) = \text{acos}\left[(\text{Tr}(\mathsf{R}^{\intercal}_{\text{gt}}\mathsf{R}_{\text{est}}) - 1)/2\right]$, respectively. We also computed the metric coined 'recall' in [7], which is the percentage of registrations whose rotation and translation errors are both smaller than predefined thresholds, i.e., the proportion of successful registrations. We report the average rotation error RE_{all} and translation error TE_{all} on *all* pairs of scans, as well as average rotation error RE and translation error TE on the subset of *successful* registrations. Performance on ModelNet40 is computed using the metrics of [37].

4.3. Comparison with Existing Methods

We report in this section the performance of PCAM on 3DMatch and KITTI. The results obtained on ModelNet40 are available in the supplementary material and show that PCAM outperforms the concurrent methods on this dataset. In what follows, PCAM-Soft and PCAM-Sparse refer to our method with soft (5) and sparse maps (6), respectively.

	ϕ	Opt.	ICP	TE_{all}	RE_{all}	Recall	TE	RE
FGR [45]						0.2	0.41	1.02
RANSAC [31]						34.2	0.26	1.39
FCGF [8]						**98.2**	0.1	0.33
DGR [7]				0.77	2.32	63.1	0.28	0.56
DGR [7]	✓			0.76	2.32	63.4	0.28	0.56
DGR [7]	✓	✓		0.34	1.62	96.6	0.21	0.33
DGR [7]	✓	✓	✓	0.16	1.43	**98.2**	**0.03**	**0.14**
PCAM - Soft				0.18	1.00	97.2	0.08	0.33
PCAM - Sparse				0.22	1.17	96.5	0.08	0.31
PCAM - Soft			✓	**0.12**	**0.79**	98.0	**0.03**	**0.14**
PCAM - Sparse			✓	0.17	1.04	97.4	**0.03**	**0.14**

Table 2. **Results on KITTI test split**. The recall is computed using 0.6 m and 5° as thresholds. The scores of all variants of DGR are obtained using the official implementation of DGR [7], and those of the other concurrent methods are reported from [7]. The columns 'ϕ', 'Opt.', and 'ICP' indicate respectively whether hard-thresholding on the confidence scores, the pose refinement of [7], and post-processing by ICP, are used or not.

4.3.1 Indoor Dataset: 3DMatch

We train PCAM on 3DMatch using $L = 6$ layers, with point clouds obtained by sampling $N = M = 4096$ points at random from the voxelised point clouds (voxel size of 5 cm). The parameter κ in (9) is set at 12 cm.

For a thorough comparison with DGR [7], we study the performance of PCAM after applying each DGR's post-processing steps: filtering the confidence weights with ϕ, refining the pose estimation using the optimisation proposed in [7], and using the same safeguard as in [7]. The threshold applied in ϕ is tuned on 3DMatch's validation set. The safeguard is activated when the ℓ_1-norm of the confidence weights is below a threshold, in which case the transformation is estimated using RANSAC (see details in [7]). For a fair comparison with DGR, we compute this second threshold such that DGR and PCAM use the safeguard on the same proportion of scans.

We report the performance of PCAM and concurrent methods in Table 1. PCAM achieves the best results and this is confirmed in the curves of Fig. 2. We provide illustrations of the quality of matched points and registrations in the supplementary material. We did not compare PCAM with RLL, which is not performing as well as DGR on 3DMatch [22].

4.3.2 Outdoor Dataset: KITTI

We train PCAM on 3DMatch using $L = 6$ layers with point clouds of (at most) $N = M = 2048$ points drawn at random from the voxelised point clouds (voxel size of 30 cm). The parameter κ in (9) is set at 60 cm.

The performance of our method is compared to others in Table 2. As scores after refinement of DGR registration by ICP are reported on this dataset in [7], we also include the scores of PCAM after the same refinement in this table. PCAM outperforms DGR on all metrics with both sparse (6) and soft (5) maps, except on the recall when using sparse maps, where it is just 0.1 point behind DGR. When combined with ICP, PCAM also achieves better results than DGR except on the recall but where it is just 0.2 point behind with soft maps and 0.8 with sparse maps. Finally, we notice in Fig. 2 a clear advantage of PCAM over DGR for the estimation of the translation. Note that PCAM achieves better performance than DGR without the need to use ϕ, nor used the pose refinement of [7]; we did not notice any significant improvement when using them on KITTI. We compare PCAM to RLL on the version of KITTI used in [22]: PCAM outperforms RLL with a recall of 84.7% for sparse maps and 86.5% for soft maps, vs 76.9% for RLL.

4.4. Ablation Study

In this section, we conduct two ablation studies: the first justifies our choice of the training loss; the second demonstrates the benefit of the product of attention matrices and of the exchange of contextual information between point clouds. The performance of the trained models are evaluated on the validation set of the considered datasets. On 3DMatch and KITTI, the input point clouds are obtained by drawing at random (at most) $N = M = 2048$ points from the voxelised point clouds. Because of these random draws, different evaluations lead to different scores. To take into account these variations, we report the average scores and the standard deviations obtained using three evaluations on the validation set for each trained model. On ModelNet40, we do not subsample the point clouds. The rotation R_{est} and translation $\boldsymbol{t}_{\text{est}}$ are computed using the method described in Sec. 3.5 *without* using any hard-thresholding on the confidence scores. We conduct experiments with $L = 2$ or $L = 6$ layers. For the experiments at $L = 2$, two models are trained from different initialisation and the reported average scores and the standard deviations are computed using both models. One model is trained for the experiments at $L = 6$.

4.4.1 Training Loss

The experiments in this section are conducted with $L = 2$ layers and using the product of attention matrices (4).

We recall that the loss L^{cc} cannot be removed as there exists a trivial solution where all the confidence scores are equal to zero in absence of this loss. Hence, the only loss that can be removed is L^{gc}. The results in Table 3 show that removing L^{gc} yields worse performance both with soft and sparse mappings. We explain this result by the fact that R_{est} and $\boldsymbol{t}_{\text{est}}$ are obtained using a slightly modified version of L^{gc} at test time (see Sec. 3.5), hence the presence of L^{gc} at

	L^{ca}	L^{ga}	L^{gc}	$\odot_\ell \mathsf{A}^{(\ell)}$	$\mathsf{A}^{(L)}$	A	L	3DMatch Rec. (%)	3DMatch RE_{all}	3DMatch TE_{all}	KITTI Rec. (%)	KITTI RE_{all}	KITTI TE_{all}	ModelNet40 RMSE (R) ($\times 10^{-3}$)	ModelNet40 RMSE (t) ($\times 10^{-3}$)
Soft map	✓	✓	✓	✓			2	53.7 (1.8)	36.4 (1.9)	0.49 (0.03)	93.4 (0.6)	3.0 (0.2)	0.45 (0.02)	18 (1.2)	0.13 (0.002)
	✓		✓	✓			2	52.3 (0.6)	36.3 (1.0)	0.49 (0.01)	94.3 (0.5)	2.5 (0.1)	0.42 (0.02)	15 (2.7)	0.09 (0.014)
	✓			✓			2	45.9 (1.5)	41.9 (2.5)	0.55 (0.03)	93.7 (0.3)	3.0 (0.1)	0.44 (0.01)	18 (1.1)	0.12 (0.015)
		✓	✓	✓			2	19.4 (2.7)	67.4 (1.5)	0.85 (0.02)	90.9 (1.1)	3.1 (0.1)	0.52 (0.04)	494 (218)	1.78 (0.879)
	✓		✓		✓		2	37.1 (2.9)	47.9 (2.2)	0.65 (0.02)	90.4 (0.9)	3.5 (0.1)	0.61 (0.08)	38 (3.9)	0.28 (0.007)
	✓		✓			✓	2	37.3 (1.4)	47.2 (0.6)	0.65 (0.01)	92.7 (0.4)	2.8 (0.2)	0.53 (0.01)	66 (10.5)	0.39 (0.010)
	✓		✓	✓			6	82.4 (0.3)	13.5 (0.3)	0.21 (0.01)	95.4 (0.3)	2.2 (0.1)	0.33 (0.01)	16	0.11
	✓		✓		✓		6	76.6 (1.9)	18.2 (0.8)	0.27 (0.01)	93.5 (0.8)	2.9 (0.03)	0.46 (0.03)	22	0.16
	✓		✓			✓	6	64.4 (1.1)	27.8 (0.6)	0.38 (0.01)	93.9 (0.0)	2.0 (0.1)	0.33 (0.01)	28	0.16
Sparse map	✓	N/A	✓	✓			2	52.4 (2.1)	38.6 (2.0)	0.51 (0.02)	94.6 (0.5)	2.8 (0.5)	0.39 (0.02)	19 (1.9)	0.10 (0.016)
	✓	N/A		✓			2	48.5 (1.0)	39.4 (1.5)	0.53 (0.02)	94.4 (0.4)	2.6 (0.4)	0.36 (0.03)	24 (6.6)	0.14 (0.031)
	✓	N/A	✓		✓		2	54.1 (3.8)	36.0 (2.7)	0.49 (0.03)	94.2 (0.5)	2.6 (0.2)	0.43 (0.05)	15 (0.4)	0.09 (0.002)
	✓	N/A	✓			✓	2	50.9 (1.3)	38.9 (1.5)	0.52 (0.02)	94.6 (0.8)	2.7 (0.3)	0.43 (0.03)	17 (1.3)	0.09 (0.004)
	✓	N/A	✓	✓			6	81.6 (1.2)	16.3 (1.2)	0.24 (0.01)	95.9 (0.3)	2.3 (0.05)	0.33 (0.01)	36	0.20
	✓	N/A	✓		✓		6	78.4 (1.1)	17.5 (1.3)	0.24 (0.01)	96.5 (0.3)	2.4 (0.1)	0.46 (0.003)	27	0.15
	✓	N/A	✓			✓	6	68.0 (1.1)	25.9 (0.5)	0.34 (0.01)	95.7 (0.3)	1.9 (0.1)	0.33 (0.01)	18	0.10

Table 3. **Ablation study on the validation set of 3DMatch, KITTI, and ModelNet40**. The losses L^{ca}, L^{gc} L^{ga} are defined in the main paper. The global attention matrix $\mathsf{A}^{(*)}$ is either equal to the pointwise multiplication of all the attention matrices ($\odot_\ell \mathsf{A}^{(\ell)}$), or to the last attention matrix ($\mathsf{A}^{(L)}$), or to the attention matrix (A) computed at the last layer of our network in a version where we have removed all the intermediate attention matrices $\ell = 1, \dots, L$ (see main paper for details).

training time probably improves the quality of confidence scores for the estimation of the transformation.

With soft maps (5), we can use another loss to train g in replacement or in complement to L^{ca}. The motivation for this loss is that L^{ca} penalises all points that are not mapped correctly to the ideal target point in the same way, whether the mapped point is close to the ideal location or far away. Instead, we can consider the possibility to use a geometric loss, $L^{\text{ga}} = L^{\text{ga}}_{\mathcal{PQ}} + L^{\text{ga}}_{\mathcal{QP}}$, that takes into account the distance between the estimated and ideal corresponding points:

$$L^{\text{ga}}_{\mathcal{PQ}} = \frac{1}{|\mathcal{C}_{\mathcal{P}}|} \sum_{u \in \mathcal{C}_{\mathcal{P}}} \|\tilde{m}_{\mathcal{Q}}(\boldsymbol{p}_u) - m_{\mathcal{Q}}(\boldsymbol{p}_u)\|_2 \,. \qquad (11)$$

$L^{\text{ga}}_{\mathcal{QP}}$ is defined likewise using $\mathcal{C}_{\mathcal{Q}}$. Note that it can be used to train g only in the case of soft maps (5) as $\tilde{m}_{\mathcal{Q}}$ is not differentiable when using sparse maps (6). The results in Table 3 show that the performance drops significantly when using L^{ga} alone. Note that a similar observation was made in [46]. Finally, using $L^{\text{ga}} + L^{\text{ca}}$ yields mildly better performance only on 3DMatch compared to using L^{ca} alone.

In view of these results, we chose to train PCAM using $L^{\text{ca}} + L^{\text{cc}} + L^{\text{gc}}$, both for soft and sparse maps.

4.4.2 Role of the Attention Matrices

Product of attention matrices: $\odot_\ell \mathsf{A}^{(\ell)}$ vs $\mathsf{A}^{(L)}$. To show the benefit of mixing low-level geometric information and high-level context information via the proposed product of attention matrices (4), we compare the performance of PCAM when replacing (4) by $\mathsf{A}^{(*)} = \mathsf{A}^{(L)}$, *i.e.*, using only the cross-attention matrix at the deepest layer. Note that the matrices $\mathsf{A}^{(1)}, \dots, \mathsf{A}^{(L-1)}$ are still present in g.

We expect the interest of the mix of information (4) to appear at large L (so that there is enough information to mix) and on challenging datasets where small overlaps and large transformations yield many ambiguities to be resolved for accurate point matching. In that respect, the most challenging dataset used in this work is 3DMatch. One can indeed verify in Table 3 that all models yields quite similar and very good results on KITTI and ModelNet40, leaving little room to clearly observe the impact of different network architectures. On ModelNet40, we highlight that all scores are close to zero (the scores are multiplied by 10^3 in Table 3) which makes it difficult to rank the different strategies as the difference between them could be due to training noise. In contrast, the impact of different choices of architecture is much more visible on 3DMatch.

We clearly observe the benefit of increasing L from 2

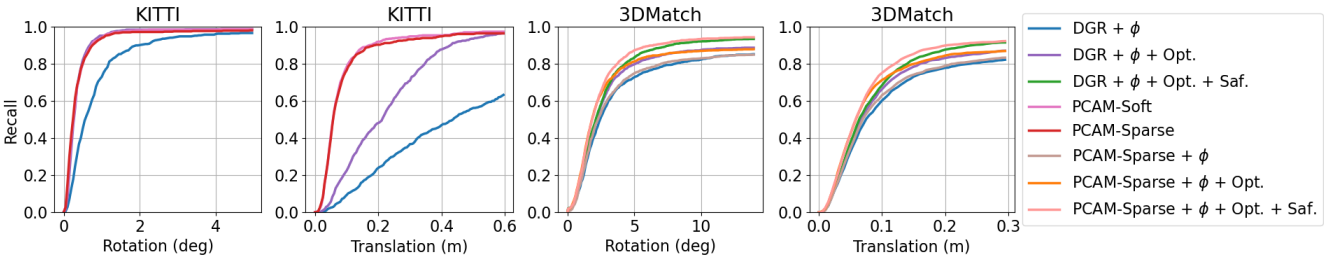

Figure 2. Registration recall vs rotation and translation error thresholds for KITTI (left) and 3DMatch (right).

to 6, and of mixing low-level geometric information and high-level context via the product of cross-attention matrices at $L = 6$ on 3DMatch. At $L = 2$, we observe a benefit of the product of attention matrices when using soft maps on 3DMatch, while this advantage is not visible when using sparse maps. We explain this fact because the product can make the soft maps more tightly concentrated around the found corresponding points, yielding better quality maps. This effect is invisible with sparse maps at $L = 2$ because, by construction, these maps are tightly concentrated around the corresponding points that are found.

Sharing contextual information: $\mathsf{A}^{(L)}$ vs A. To show the benefit of exchanging information between point clouds via the intermediate cross-attention matrices $\mathsf{A}^{(1)}, \ldots, \mathsf{A}^{(L-1)}$, we conduct experiments where we remove all intermediate matrices but the last one, which we denote by A. As before, we expect the benefit of this strategy to be more visible at large L and on challenging datasets such as 3DMatch. The results in Table 3 confirm this result where the benefit of exchanging contextual information is particularly noticeable at $L = 6$ on 3DMatch.

5. Discussion

In this section we differentiate our method from the most closely related works on three main aspects: exchange of contextual information between point clouds, computation of the pairs of corresponding points, and point-pair filtering.

Exchange of contextual information. Multiple cross-attention layers in PCAM allows the network to progressively exchange information between point clouds at *every* layer to find the best matches in overlapping regions. DCP [38], PRNet [37], and OPRNet [9] use transformer layers that also exchange information between point clouds, but only in the deepest layers of their convnet. PREDATOR [18] uses one cross-attention layer in the middle of their U-Net, hence only exchanging high-level contextual information. The newly published method [12] uses several cross-attention layers but, unlike us, does not merge them for point matching. There is no exchange of information between point clouds in [7, 23, 41].

Point matching. PCAM uses *multiple* cross-attention matrices that allows the network to exploit both fine-grained geometric information and high-level context to find pairs of corresponding points. In contrast, DCP [38] and PRNet [37] use only one attention layer at the deepest layer of their network, which might limit their performance due to the lack of this fine-grained geometric information. RPMNet[41], OPRNet [9] use a single Sinkhorn layer to establish the correspondences. IDAM [23] replaces the dot product operator by a learned convolution module that computes the similarity score between point features. The point-matching module in [15] is equivalent to ours in the non-optimal setup of one attention A.

Point filtering. PCAM first constructs pairs of corresponding points and assign to each of them a confidence score using a point convolution network. A similar strategy is used in [15]. DCP [38] does not have any filtering step, hence cannot handle partial-to-partial registration. Before starting to match points, PRNet [37] first selects a subset of points to be matched in each point cloud using the ℓ_2-norm of point features. Similarly to PRNet [37], PREDATOR [18] also start with a selection of points to be matched in each point cloud, using the learned overlap probability and matchability scores. DGR [7] uses a similar process to PCAM but uses 6D convolutions to predict the confidence score, whereas we use 3D convolutions with 6D point-features (the concatenation of the coordinates of the point itself and its match point). IDAM [23] proposes a two-stage point elimination technique where points are individually filtered first; pairs of points are then discarded. Finally, [9, 12, 41] use their Sinkhorn layer to filter outliers.

6. Conclusion

In this work, we have proposed a novel network architecture where we transfer information between point clouds to find the best match within the overlapping regions, and mix low-level geometric information and high-level contextual knowledge to find correspondences between point clouds. Our experiments show that this architecture achieves state-of-the-art results on several datasets.

Acknowledgements. This work was partly performed using HPC resources from GENCI–IDRIS (Grant 2020-AD011012040). We thank Raoul de Charette for his constructive feedbacks on an earlier version of this paper.

References

[1] G. Agamennoni, S. Fontana, R. Y. Siegwart, and D. G. Sorrenti. Point Clouds Registration with Probabilistic Data Association. In *International Conference on Intelligent Robots and Systems (IROS)*, pages 4092–4098, 2016.

[2] Y. Aoki, H. Goforth, R. A. Srivatsan, and S. Lucey. PointNetLK: Robust & Efficient Point Cloud Registration Using PointNet. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 7156–7165, 2019.

[3] P. J. Besl and N. D. McKay. A method for registration of 3-D shapes. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 14(2):239–256, 1992.

[4] S. Bouaziz, A. Tagliasacchi, and M. Pauly. Sparse Iterative Closest Point. *Computer Graphics Forum*, 32(5):113–123, 2013.

[5] A. Boulch, G. Puy, and R. Marlet. FKAConv: Feature-Kernel Alignment for Point Cloud Convolution. In *Asian Conference on Computer Vision (ACCV)*, 2020.

[6] Y. Chen and G. Medioni. Object modeling by registration of multiple range images. In *International Conference on Robotics and Automation (ICRA)*, volume 3, pages 2724—-2729, 1991.

[7] C. Choy, W. Dong, and V. Koltun. Deep Global Registration. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2511–2520, 2020.

[8] C. Choy, J. Park, and V. Koltun. Fully convolutional geometric features. In *International Conference on Computer Vision (ICCV)*, pages 8957–8965, 2019.

[9] Z. Dang, F. Wang, and M. Salzmann. Learning 3D-3D Correspondences for One-Shot Partial-to-partial Registration. *arXiv:2006.04523*, 2020.

[10] H. Deng, T. Birdal, and S. Ilic. PPF-FoldNet: Unsupervised Learning of Rotation Invariant 3D Local Descriptors. In *European Conference on Computer Vision (ECCV)*, 2018.

[11] H. Deng, T. Birdal, and S. Ilic. PPFNet: Global Context Aware Local Features for Robust 3D Point Matching. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 195–205, 2018.

[12] K. Fischer, M. Simon, F. Olsner, S. Milz, H.-M. Gross, and P. Mader. StickyPillars: Robust and Efficient Feature Matching on Point Clouds Using Graph Neural Networks. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 313–323, 2021.

[13] A. Fitzgibbon. Robust Registration of 2D and 3D Point Sets. *Image and Vision Computing*, 21:1145–1153, 2002.

[14] A. Geiger, P. Lenz, and R. Urtasun. Are we ready for Autonomous Driving? The KITTI Vision Benchmark Suite. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2012.

[15] Z. Gojcic, C. Zhou, J. Wegner, L. Guibas, and T. Birdal. Learning Multiview 3D Point Cloud Registration. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020.

[16] Z. Gojcic, C. Zhou, J. Wegner, and A. Wieser. The Perfect Match: 3D Point Cloud Matching With Smoothed Densities. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5540–5549, 2019.

[17] J. Gower. Generalized procrustes analysis. *Psychometrika*, 40:33–51, 1975.

[18] S. Huang, Z. Gojcic, M. Usvyatsov, A. Wieser, and K. Schindler. PREDATOR: Registration of 3D Point Clouds with Low Overlap. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4267–4276, 2021.

[19] G. Izatt, H. Dai, and R. Tedrake. Globally Optimal Object Pose Estimation in Point Clouds with Mixed-Integer Programming. In *International Symposium on Robotics Research*, 2017.

[20] A. E. Johnson and M. Hebert. Using spin images for efficient object recognition in cluttered 3d scenes. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 21(5):433–449, 1999.

[21] M. Khoury, Q.-Y. Zhou, and V. Koltun. Learning Compact Geometric Features. In *International Conference on Computer Vision (ICCV)*, pages 153–161, 2017.

[22] F. J. Lawin and P.-E. Forssén. Registration loss learning for deep probabilistic point set registration. In *International Conference on 3D Vision (3DV)*, 2020.

[23] J. Li, C. Zhang, Z. Xu, H. Zhou, and C. Zhang. Iterative Distance-Aware Similarity Matrix Convolution with Mutual-Supervised Point Elimination for Efficient Point Cloud Registration. In *European Conference on Computer Vision (ECCV)*, 2020.

[24] I. Loshchilov and F. Hutter. Decoupled weight decay regularization. In *International Conference on Learning Representations (ICLR)*, 2019.

[25] B. Lucas and T. Kanade. An Iterative Image Registration Technique with an Application to Stereo Vision. In *International joint conference on Artificial intelligence*, volume 2, pages 674—-679, 1981.

[26] A. Makadia, A. Patterson, and K. Daniilidis. Fully automatic registration of 3d point clouds. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, volume 1, pages 1297–1304, 2006.

[27] H. Maron, N. Dym, I. Kezurer, S. Kovalsky, and Y. Lipman. Point Registration via Efficient Convex Relaxation. *ACM Transactions on Graphics*, 35(4):1–12, 2016.

[28] F. Pomerleau, F. Colas, and R. Siegwart. *A Review of Point Cloud Registration Algorithms for Mobile Robotics*. 2015.

[29] D.M. Rosen, L. Carlone, A.S. Bandeira, and J.J. Leonard. A Certifiably Correct Algorithm for Synchronization over the Special Euclidean Group. In *International Workshop on the Algorithmic Foundations of Robotics (WAFR)*, volume 13, 2020.

[30] S. Rusinkiewicz and M. Levoy. Efficient variants of the ICP algorithm. In *International Conference on 3-D Digital Imaging and Modeling*, pages 145–152, 2001.

[31] R. B. Rusu, N. Blodow, and M. Beetz. Fast Point Feature Histograms (FPFH) for 3D registration. In *International Conference on Robotics and Automation (ICRA)*, pages 3212–3217, 2009.

[32] S. Salti, F. Tombari, and L. Di Stefano. SHOT: Unique Signatures of Histograms for Surface and Texture Description. *Computer Vision and Image Understanding*, 125, 2014.

[33] A. Segal, D. Hähnel, and S. Thrun. Generalized-ICP. In *Robotics: Science and Systems*, 2009.

[34] L. Silva, O.R.P. Bellon, and K. L. Boyer. Precision range image registration using a robust surface interpenetration measure and enhanced genetic algorithms. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 27(5), 2005.

[35] F. Tombari, S. Salti, and L. Di Stefano. Unique shape context for 3d data description. In *ACM workshop on 3D object retrieval*, 2010.

[36] Y. Tsin and T. Kanade. A correlation-based approach to robust point set registration. In *European Conference on Computer Vision (ECCV)*, pages 558—-569, 2004.

[37] Y. Wang and J. Solomon. PRNet: Self-Supervised Learning for Partial-to-Partial Registration. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 32, pages 8814–8826, 2019.

[38] Y. Wang and J. M. Solomon. Deep Closest Point: Learning Representations for Point Cloud Registration. In *International Conference on Computer Vision (ICCV)*, pages 3522–3531, 2019.

[39] Z. Wu, S. Song, A. Khosla, F. Yu, L. Zhang, X. Tang, and J. Xiao. 3D ShapeNets: A deep representation for volumetric shapes. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1912–1920, 2015.

[40] J. Yang, H. Li, D. Campbell, and Y. Jia. Go-ICP: A Globally Optimal Solution to 3D ICP Point-Set Registration. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 38(11):2241–2254, 2016.

[41] Z. J. Yew and G. H. Lee. Rpm-net: Robust point matching using learned features. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020.

[42] W. Yuan, B. Eckart, K. Kim, V. Jampani, D. Fox, and J. Kautz. Deepgmr: Learning latent gaussian mixture models for registration. In *European Conference on Computer Vision (ECCV)*, pages 733–750, 2020.

[43] A. Zeng, S. Song, M. Nießner, M. Fisher, J. Xiao, and T. Funkhouser. 3DMatch: Learning Local Geometric Descriptors from RGB-D Reconstructions. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[44] Z. Zhang. Iterative point matching for registration of freeform curves and surfaces. *International Journal on Computer Vision*, 13(2):119–152, 1994.

[45] Q.-Y. Zhou, J. Park, and V. Koltun. Fast Global Registration. In *European Conference on Computer Vision (ECCV)*, pages 766–782, 2016.

[46] T. Zodage, R. Chakwate, V. Sarode, R. A. Srivatsan, and H. Choset. Correspondence matrices are underrated. In *International Conference on 3D Vision (3DV)*, 2020.