ACNe: Attentive Context Normalization for Robust Permutation-Equivariant Learning

Weiwei Sun[1] Wei Jiang[1] Eduard Trulls[2] Andrea Tagliasacchi[3] Kwang Moo Yi[1]
[1]University of Victoria [2]Google Research, Zurich [3]Google Research, Toronto
{weiweisun, jiangwei, kyi}@uvic.ca {trulls, taglia}@google.com

Abstract

Many problems in computer vision require dealing with sparse, unordered data in the form of point clouds. Permutation-equivariant networks have become a popular solution – they operate on individual data points with simple perceptrons and extract contextual information with global pooling. This can be achieved with a simple normalization of the feature maps, a global operation that is unaffected by the order. In this paper, we propose Attentive Context Normalization (ACN), a simple yet effective technique to build permutation-equivariant networks robust to outliers. Specifically, we show how to normalize the feature maps with weights that are estimated within the network, excluding outliers from this normalization. We use this mechanism to leverage two types of attention: local and global – by combining them, our method is able to find the essential data points in high-dimensional space to solve a given task. We demonstrate through extensive experiments that our approach, which we call Attentive Context Networks (ACNe), provides a significant leap in performance compared to the state-of-the-art on camera pose estimation, robust fitting, and point cloud classification under noise and outliers. Source code: https://github.com/vcg-uvic/acne.

1. Introduction

Several problems in computer vision require processing sparse, *unordered* collections of vectors $\mathcal{P} = \{\mathbf{p}_n \in \mathbb{R}^D\}$, commonly called *clouds*. Examples include pixel locations (D=2), point clouds from depth sensors (D=3), and sparse correspondences across a pair of images (D=4). The latter includes wide-baseline stereo, one of the fundamental problems in computer vision. It lies at the core of Structure-from-Motion (SfM), which, in turn, is the building block of applications such as 3D reconstruction [1], image-based rendering [43] and time-lapse smoothing [28].

Wide-baseline stereo has been traditionally solved by extracting small collections of discrete *keypoints* [31] and finding *correspondences* among them with robust estimators [16], a reliable approach used for well over two decades. This has changed over the past few years, with the arrival of deep learning and an abundance of new dense [57, 47, 60] and sparse [55, 12, 39, 59, 27] methods. Here, we focus on sparse methods, which have seen many recent developments made possible by the introduction of *PointNets* [36, 37] – neural networks that rely on multi-layer perceptrons and global pooling to process unordered data in a *permutation-equivariant* manner – something which is not feasible with neither convolutional nor fully-connected layers.

Networks of this type – hereafter referred to as *permutation-equivariant networks* – have pioneered the application of deep learning to point clouds. The original PointNet relied on the concatenation of point-wise (context-agnostic) and global (point-agnostic) features to achieve permutation equivariance. Yi et al. [55] proposed *Context Normalization* (CN) as a simple, yet effective alternative to global feature pooling: all it requires is a non-parametric normalization of the feature maps to zero mean and unit variance. Contrary to other normalization techniques utilized by neural networks [22, 2, 46, 51], whose primary objective is to improve convergence, context normalization is used to generate contextual information while preserving permutation equivariance. Despite its simplicity, it proved more effective than the PointNet approach on wide-baseline stereo, contributing to a relative increase in pose estimation accuracy of $50-100\%$; see [55, Fig. 5].

Note that CN normalizes the feature maps according to first- (mean) and second- (variance) order moments. Interestingly, these two quantities can be expressed as the solution of a least-squares problem:

$$\hat{\boldsymbol{\mu}} = \underset{\boldsymbol{\mu}}{\operatorname{argmin}} \sum_n \|\mathbf{p}_n - \boldsymbol{\mu}\|_2^2 \tag{1}$$

$$\hat{\boldsymbol{\sigma}} = \underset{\boldsymbol{\sigma}}{\operatorname{argmin}} \sum_n \left\| \|\mathbf{p}_n - \hat{\boldsymbol{\mu}}\|_2^2 - \boldsymbol{\sigma}^{\circ 2} \right\|_2^2 \tag{2}$$

However, it is well known that least-squares optimization is *not robust* to outliers [6, Sec. 3], a problem that also afflicts CN. We illustrate this limitation in Fig. 1, where

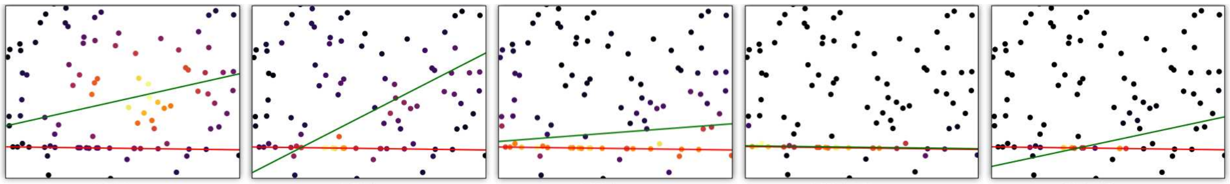

Figure 1. **Robust neural line fitting** – We learn to fit lines with outliers (80%) via our ACNe, as well as CNe [55]. We visualize the **ground truth** and the **network estimates**. We color-code the weights learned by the k-th residual layer of ACNe and used to normalize the feature maps – notice that our method, which mimics Iterative Re-weighted Least Squares (IRLS), learns to *progressively* focus its attention on the inliers. This allows ACNe to find the correct solution where CNe fails.

the toy task is to fit a line to data corrupted by outliers. Note that this is a *critical weakness*, as the application CN was originally devised for, wide-baseline stereo, a problem *plagued with outliers* : outlier ratios above 80% are typical in standard public datasets; see Section 4.3.

To address this issue, we take inspiration[1] from a classical technique used in robust optimization: Iteratively Re-weighted Least Squares (IRLS) [8]. As an example, let us consider the computation of the first-order moment (1). Rather than using the square of the residuals, we can optimize with respect to a *robust* kernel κ that allows for outliers to be ignored:

$$\underset{\boldsymbol{\mu}}{\operatorname{argmin}} \sum_n \kappa(\|\mathbf{p}_n - \boldsymbol{\mu}\|_2), \tag{3}$$

which can then be converted back into an *iterative* least-squares optimization (t indexes iterations):

$$\underset{\boldsymbol{\mu}^t}{\operatorname{argmin}} \sum_n \underbrace{\psi(\|\mathbf{p}_n - \boldsymbol{\mu}^{t-1}\|_2)^{-1}}_{\text{attention } w_n^t} \|\mathbf{p}_n - \boldsymbol{\mu}^t\|_2^2, \tag{4}$$

where $\psi(\cdot)$ is the *penalty function* associated with the kernel $\kappa(\cdot)$; see [33, 17]. Inspired by this, we design a network that learns to *progressively* focus its attention on the inliers, operating analogously to $\psi(.)$ over the IRLS iterations.

Specifically, we propose to train a perceptron that translates the (intermediate) feature maps into their corresponding attention weights, and normalizes them accordingly. We denote this approach as *Attentive Context Normalization* (ACN), and the networks that rely on this mechanism *Attentive Context Networks* (ACNe). We consider two types of attention, one that operates on each data point individually (local), and one that estimates the relative importance of data points (global), and demonstrate that using them together yields the best performance. We also evaluate the effect of supervising this attention mechanism when possible. We verify the effectiveness of our method on (1) robust line fitting, (2) classification of 2D and 3D point clouds, and (3) wide-baseline stereo on real-world datasets (outdoors *and* indoors) showing significant improvements over the state of the art. Our work is, to the best our knowledge, the first to apply attentive mechanisms to the *normalization* of feature maps. One can also apply a more common form of attention by operating *directly* on feature maps [49, 15], but we demonstrate that this does not perform as effectively.

[1]See Section G of the supplementary material. A robust kernel can also be trained with a neural network and enforcing monotonicity; see [11].

2. Related work

We discuss recent works on deep networks operating on point clouds, review various normalization methods for deep networks, and briefly discuss attention mechanisms.

Deep networks for point clouds. Several methods have been proposed to process point cloud data with neural networks. These include graph convolutional networks [13, 26], VoxelNets [61], tangent convolutions [44], and many others. A simpler strategy was introduced by PointNets [36, 37], which has since become a popular solution due to its simplicity. At their core, they are convolutional neural networks with 1×1 kernels and global pooling operations. Enhancements to the PointNet architecture include incorporating locality information with kernel correlation [42], and contextual information with LSTMs [30]. Another relevant work is Deep Sets [56], which derives neural network parameterizations that guarantee permutation-equivariance.

Permutation-equivariant networks for stereo. While PointNets were originally introduced for segmentation and classification of 3D point clouds, Yi et al. [55] demonstrated that they can also be highly effective for robust matching in stereo, showing a drastic leap in performance against hand-crafted methods [16, 45, 5]. The core ingredient of Yi et al. [55] is Context Normalization (CN), an alternative to global feature pooling from PointNets. While similar to other normalization techniques for deep networks [22, 2, 46, 51], CN has a different role – to aggregate point-wise feature maps and generate *contextual* information. Follow-ups to CN include the use of architectures similar to Yi et al. [55] to iteratively estimate fundamental matrices [39], novel loss formulations [12], and the modeling of locality [59]. In OANet [58], order-aware filtering was utilized to incorporate context and spatial correlation. While all of these works rely on "vanilla" CN, we show how to improve its performance by embedding an attention mechanism therein. Our improvements are compatible with *any* of these techniques.

Normalization in deep networks. In addition to CN, different strategies have been proposed to normalize feature maps in a deep network, starting with the seminal work of Batch Normalization [22], which proposed to normalize the feature maps over a mini-batch. Layer Normalization [2] transposed this operation by looking at all channels for a single sample in the batch, whereas Group Normalization [51] applied it over subsets of channels. Further efforts have proposed to normalize the weights instead of the activations [41], or their eigenvalues [34]. The main use of all these normalization techniques is to stabilize the optimization process and speed up convergence. By contrast, Instance Normalization [46] proposed to normalize individual image samples for style transfer, and was improved upon in [21] by aligning the mean and standard deviation of content and style. Regardless of the specifics, all of these normalization techniques operate on the entire sample – in other words, they do not consider the presence of outliers or their statistics. While this is not critical in image-based pipelines, it can be extremely harmful for point clouds; see Fig. 1.

Attentional methods. The core idea behind attention mechanisms is to *focus* on the crucial parts of the input. There are different forms of attention, and they have been applied to a wide range of machine learning problems, from natural language processing to images. Vaswani et al. [48] proposed an attentional model for machine translation eschewing recurrent architectures. Luong et al. [32] blended two forms of attention on sequential inputs, demonstrating performance improvements in text translation. Xu et al. [54] showed how to employ soft and hard attention to gaze on salient objects and generate automated image captions. Local response normalization has been used to find salient responses in feature maps [24, 29], and can be interpreted as a form of lateral inhibition [19]. The use of attention in convolutional deep networks was pioneered by Spatial Transformer Networks [23], which introduced a differentiable sampler that allows for spatial manipulation of the image. In [53], attention is directly applied to the feature map, given by a PointNet-style network operating on point clouds. However, this strategy does not work as well as ours for wide-baseline stereo; see Section B in the supplementary material.

3. Attentive Context Normalization

Given a feature map $\mathbf{f}\in\mathbb{R}^{N\times C}$, where N is the number of features (or data points at layer zero), C is the number of channels, and each row corresponds to a data point, we recall that Context Normalization [55] is a non-parametric operation that can be written as

$$\mathcal{N}_{CN}(\mathbf{f}) = (\mathbf{f} - \mu(\mathbf{f})) \oslash \sigma(\mathbf{f}), \tag{5}$$

where $\mu(\mathbf{f})=\mathbb{E}[\mathbf{f}]$ is the arithmetic mean, $\sigma(\mathbf{f})=\sqrt{\mathbb{E}[(\mathbf{f}-\mathbb{E}[\mathbf{f}])^{\circ 2}]}$ is the standard deviation of the features across N, and $\oslash$ denotes the element-wise division. Here we assume a single cloud, but generalizing to multiple clouds (i.e. batch) is straightforward. Note that to preserve the properties of unstructured clouds, the information in the feature maps needs to be normalized in a *permutation-equivariant* way. We extend CN by introducing a weight vector $\mathbf{w}\in[0,\dots 1]^N$, and indicate with $\mu_{\mathbf{w}}(\cdot)$ and $\sigma_{\mathbf{w}}(\cdot)$ the corresponding weighted mean and standard deviation. In contrast to Context Normalization, we compute the weights $\mathbf{w}$ with a parametric function $\mathcal{W}_{\boldsymbol{\omega}}(\cdot)$ with trainable parameters[2] $\boldsymbol{\omega}$ that takes as input the feature map, and returns a *unit norm* vector of weights:

$$\mathbf{w} = \eta(\mathcal{W}_{\boldsymbol{\omega}}(\mathbf{f}))\,, \quad \eta(\mathbf{x}) = \mathbf{x}/\|\mathbf{x}\|_1. \tag{6}$$

We then define *Attentive Context Normalization* as

$$\mathcal{N}_{ACN}(\mathbf{f};\mathbf{w}) = (\mathbf{f} - \mu_{\mathbf{w}}(\mathbf{f})) \oslash \sigma_{\mathbf{w}}(\mathbf{f}). \tag{7}$$

The purpose of the attention network $\mathcal{W}_{\boldsymbol{\omega}}(\cdot)$ is to compute a weight function that focuses the normalization of the feature maps on a subset of the input features – the inliers. As a result, the network can learn to effectively *cluster* the features, and therefore separate inliers from outliers.

There are multiple attention functions that we can design, and multiple ways to combine them into a single attention vector $\mathbf{w}$. We will now describe those that we found effective for finding correspondences in wide-baseline stereo, and how to combine and supervise them effectively.

Generating attention. We leverage two different types of attention mechanisms, *local* and *global*:

$$\mathbf{w}_i^{\text{local}} = \mathcal{W}_{\boldsymbol{\omega}}^{\text{local}}(\mathbf{f}_i) = \text{sigmoid}(\mathbf{W}\mathbf{f}_i^\top + \mathbf{b}), \tag{8}$$

$$\mathbf{w}_i^{\text{global}} = \mathcal{W}_{\boldsymbol{\omega}}^{\text{global}}(\mathbf{f}_i) = \frac{\exp\left(\mathbf{W}\mathbf{f}_i^\top + \mathbf{b}\right)}{\Sigma_{j=1}^N \exp\left(\mathbf{W}\mathbf{f}_j^\top + \mathbf{b}\right)}, \tag{9}$$

where $\mathbf{W}$ and $\mathbf{b}$ are the parameters of a perceptron, and $\mathbf{f}_k$ denotes the feature vector for data point k – the k-th row of the feature map $\mathbf{f}$. Observe that the *local* attention mechanism (8) acts on each feature vector *independently*, whereas the *global* attention mechanism (9) relates the feature vector for each data point to the *collection* through softmax.

Blending attention. Note that the product does not change the scale of the normalization applied in (7). Therefore, to take into account multiple types of attention simultaneously, we simply merge them together WLremove "toghether" to avoid redundancy through element-wise multiplication. One could use a parametric form of attention blending instead; however, it is non-trivial to combine the weights in a permutation-equivariant way, and we found this simple strategy effective.

[2]For simplicity, we abuse the notation and drop the layer index from all parameters. All the perceptrons in our work operate individually over each data point with shared parameters across each layer.

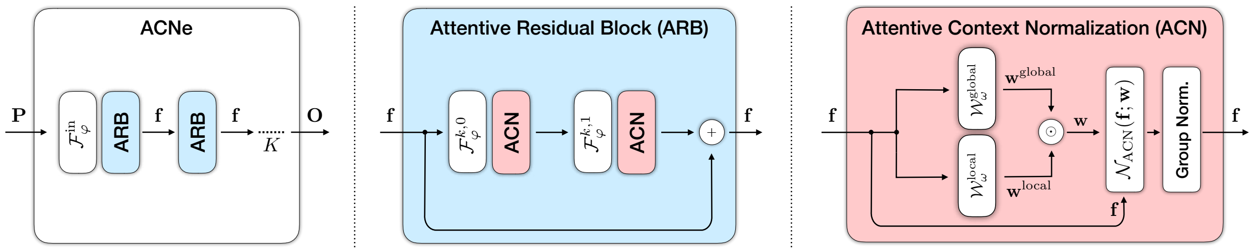

Figure 2. **ACNe architecture** – (Left) Our permutation-equivariant network receives an input tensor $\mathbf{P}$ of size $N\times D$, which is processed by a series of K Attentive Residual Blocks (ARB). The output of the network is a tensor $\mathbf{O}$ size $N\times C$, which is then converted to a representation appropriate for the task at hand. Note that the first perceptron $\mathcal{F}_\varphi^{\text{in}}$ changes the dimensionality from $\mathbf{P}$ of size $N\times D$ (input dimensions) to features $\mathbf{f}$ of size $N\times C$. (Middle) Within each residual path of the ARB, we manipulate the feature map with perceptrons $\mathcal{F}_\varphi$ with parameters φ, followed by Attentive Context Normalization (ACN) – we repeat this structure twice. (Right) An ACN module computes local/global attention with two trainable networks, combines them via element-wise multiplication, and normalizes the feature maps with said weights – the $\mathcal{N}_{\text{ACN}}$ block – followed by Group Normalization. Note that all features are all processed in the same way, *individually*, and the ACN block is the *only* place where they interact with each other – this architecture guarantees permutation-equivariance.

Supervising attention. In some problems, the class for each data point is known *a priori* and explicit supervision can be performed. In this case, adding a supervised loss on the attention signals can be beneficial. For instance, when finding good correspondences for stereo we can apply binary-cross entropy using the epipolar distance to generate labels for each putative correspondence, as in [55]. Our experiments in Section 6.4 show that while this type of supervision can provide a small boost in performance (1-2%), our approach performs nearly as well without this supervision.

4. Network architecture and applications

Our network receives as input $\mathbf{P}\in\mathbb{R}^{N\times D}$, the tensor representation of $\mathcal{P}$, and produces an output tensor $\mathbf{O}\in\mathbb{R}^{N\times C}$. Note that as $\mathcal{P}$ is unstructured, $\mathbf{O}$ must be equivariant with respect to permutations of the N rows of $\mathbf{P}$. This output tensor is then used in different ways according to the task at hand. We model our architecture after [55], which we refer to as *Context Network* (CNe). It features a series of residual blocks [20] with Context Normalization (CN). Our architecture, which we call *Attentive Context Network*, or ACNe, is pictured in Fig. 2. A key distinction is that within each normalization block (Fig. 2; right) we link the individual outputs of each perceptron $\mathcal{F}_\varphi$ to our ACN layer. We also replace the Batch Normalization layers [22] used in [55] with Group Normalization [51], as we found it performs better; see Section 6.4 for ablation tests.

We demonstrate that ACNe can be used to solve multiple applications, ranging from classical problems such as robust line fitting (Section 4.1) and point cloud classification on MNIST and ModelNet40 (Section 4.2), to robust camera pose estimation for wide-baseline stereo (Section 4.3).

4.1. Robust line fitting

We consider the problem of fitting a line to a collection of points $\mathbf{P}\in\mathbb{R}^{N\times 2}$ that is ridden by noise and outliers; see Fig. 1. This problem can be addressed via smooth (IRLS) or combinatorial (RANSAC) optimization – both methods can be interpreted in terms of sparse optimization, such that inliers and outliers are clustered separately; see [7]. Let us parameterize a line as the locus of point $[x, y]$ such that $\boldsymbol{\theta}\cdot[x,y,1]{=}0$. We can then score each row of $\mathbf{P}$ (i.e. each 2D point) by passing the output tensor $\mathbf{O}{=}\text{ACNe}(\mathbf{P})$ to an additional weight network – with local and global components – following (6), yielding weights $\mathbf{w}=\eta(\mathcal{W}_{\boldsymbol{\omega}}(\mathbf{O}))$. Given $\mathbf{w}$, and expressing our points in homogeneous coordinates as $\bar{\mathbf{P}}=[\mathbf{P},1]\in\mathbb{R}^{N\times 3}$, we can compute our covariance matrix as $\mathbf{C}_{\mathbf{w}}(\mathbf{P}){=}\bar{\mathbf{P}}^\top\text{diag}(\mathbf{w})^2\bar{\mathbf{P}}\in\mathbb{R}^{3\times 3}$. Then, denoting $\nu_0[\mathbf{C}]$ as the eigenvector of $\mathbf{C}$ corresponding to its smallest eigenvalue, $\nu_0\left[\mathbf{C}_{\mathbf{w}}(\mathbf{P})\right]$ is the estimated plane equation that we seek to find. We, therefore, minimize the difference between this eigenvector and the ground truth, with additional guidance to $\mathbf{w}^{\text{local}}$ to help convergence:

$$\mathcal{L}(\boldsymbol{\omega}) = \alpha \min_{+/-}\left\{\|\nu_0\left[\mathbf{C}_{\mathbf{w}}(\mathbf{P})\right] \pm \boldsymbol{\theta}\,\|_2^2\right\} + \beta\mathbb{E}\left[H(\mathbf{y},\mathbf{w}^{\text{local}})\right], \tag{10}$$

where $\mathbb{E}\left[H(\mathbf{a},\mathbf{b})\right]$ is the average binary cross entropy between $\mathbf{a}$ and $\mathbf{b}$, $\mathbf{y}$ is the ground-truth inlier label, and hyper-parameters α and β control the influence of these losses. The $\min_{+/-}$ resolves the issue that $-\boldsymbol{\theta}$ and $\boldsymbol{\theta}$ are the same line.

4.2. Point cloud classification

We can also apply ACNe to point cloud *classification* rather than reasoning about individual points. As in the previous application, we consider a set of 2D or 3D locations $\mathbf{P}\in\mathbb{R}^{N\times D}$ as input, where D is the number of dimensions. In order to classify each point set, we transform the output tensor $\mathbf{O}{=}\text{ACNe}(\mathbf{P})$ into a single vector $\mathbf{v}{=}\mu_{\mathbf{w}}(\mathbf{O})$ and associate it with a ground-truth one-hot vector $\mathbf{y}$ through softmax. Additional weight networks to generate $\mathbf{w}$ are trained for this task. We train with the cross entropy loss. Thus, the loss that we optimize is:

$$\mathcal{L}(\boldsymbol{\omega}) = H\left(\mathbf{y},\text{softmax}(\mathbf{v})\right). \tag{11}$$

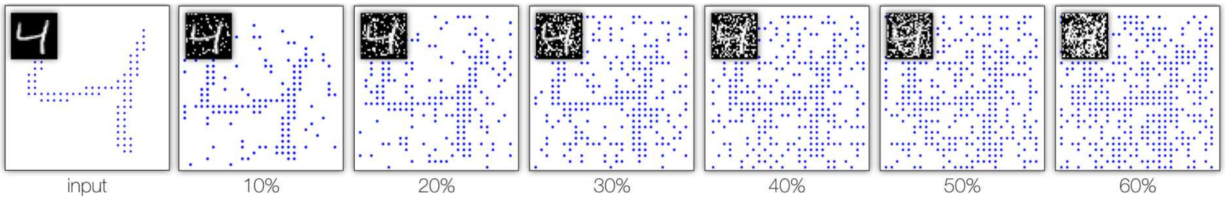

Figure 3. **Classification** – We add salt-and-pepper noise to MNIST images, and then convert the digits to an unstructured point cloud. The % reports the outlier-to-inlier ratio.

4.3. Wide-baseline stereo

In stereo we are given correspondences as input, which is thus $\mathbf{P} \in \mathbb{R}^{N\times 4}$, where N is the number of correspondences and each row contains two pixel locations on different images. In order to remain comparable with traditional methods, we aim to solve for the *Fundamental* matrix, instead of the *Essential* matrix, *i.e.*, without assuming known camera intrinsics. Thus, differently from [55, 12, 59], we simply normalize the image coordinates with the image size instead. This makes our method more broadly applicable, and directly comparable with most robust estimation methods for stereo [16, 45, 9, 10, 4].

We obtain $\mathbf{w}$ from the output tensor $\mathbf{O}$=ACNe($\mathbf{P}$) via (6), as in Section 4.1. The weights $\mathbf{w}$ indicate which correspondences are considered to be inliers and their relative importance. We then apply a *weighted* variant of the 8-point algorithm [18] to retrieve the Fundamental matrix $\hat{\mathbf{F}}$, which parameterizes the relative camera motion between the two cameras. To do so we adopt the differentiable, non-parametric form proposed by [55], and denote this operation as $\hat{\mathbf{F}}=g\left(\mathbf{X},\mathbf{w}\right)$. We then train our network to regress the ground-truth Fundamental matrix, as well as providing auxiliary guidance to $\mathbf{w}^{\text{local}}$ – the *final* local attention used to construct the output of the network – with per-correspondence labels obtained by thresholding over the symmetric epipolar distance [18], as in [55]. In addition, we also perform auxiliary supervision on $\mathbf{w}_k^{\text{local}}$ – the *intermediate* local attentions within the network – as discussed in Section 3. Note that this loss is not necessary, but helps training and provides a small boost in performance; see Section 6.4. We do not supervise global attention and leave it for the network to learn. We therefore write:

$$\mathcal{L}(\boldsymbol{\omega}) = \alpha \min_{+/-}\left\{\left\|\hat{\mathbf{F}} \pm \mathbf{F}^*\right\|_F^2\right\} + \beta\mathbb{E}\left[H(\mathbf{y},\mathbf{w}^{\text{local}})\right] + \gamma\mathbb{E}_k\left[H(\mathbf{y},\mathbf{w}_k^{\text{local}})\right], \tag{12}$$

where $\|\cdot\|_F$ is the Frobenius norm, H is the binary cross entropy, and $\mathbf{y}$ denotes ground truth inlier labels. Again, the hyper-parameters α, β, and γ control the influence of each loss. Similarly to the line-fitting case, the $\min_{+/-}$ resolves the issue that $-\mathbf{F}^*$ and $\mathbf{F}^*$ express the same solution.

5. Implementation details

We employ a K-layer structure (excluding the first linear layer that changes according to the number of channels) for ACNe, with K$\times$ ARB units, and two perceptron layers in each ARB. The number of layers K is set to 3$\times$ for 2D point cloud classification, 6$\times$ for robust line fitting, and 12$\times$ for stereo. For 3D point cloud classification, we add ACN normalization to an existing architecture. We also use 32 groups for Group Normalization, as suggested in [51]. Similarly to [55], we use C=128 channels per perceptron.

Training setup. For all applications we use the ADAM optimizer [25] with default parameters and a learning rate of 10^{-3}. Except for robust line fitting, we use a validation set to perform early stopping. For robust line fitting, the data is purely synthetic and thus infinite, and we train for 50k iterations. For MNIST, we use 70k samples with a 8:1:1 split for training, validation and testing. For stereo, we use the splits from [58]. For the loss term involving eigen-decomposition (terms multiplied by α in (10) and (12)), we use α=0.1, following [55]. All other loss terms have a weight of 1, that is, β=1 and γ=1. For stereo, we follow [55] and enable the term involving the Fundamental matrix – the first term in (12) – after 20k iterations.

Robust estimators for stereo inference. As a special case, we evaluate the possibility of applying standard robust estimators for outlier rejection such as RANSAC after training the model to potentially maximize its performance, as previously done in [55, 12, 58]. To do so, we modify our architecture by changing the final layer to output only the local attention with the ReLU+Tanh activation, as in Yi et al. [55]. We then simply threshold $\mathbf{w}$ with zero, select the data points that survive this process as inliers, and feed them to different RANSAC methods to process them further. We compare these results with those obtained directly from the weighted 8-point formulation.

6. Results

We first consider a toy example on fitting 2D lines with a large ratio of outliers. We then apply our method to point cloud classification, following [36, 37], which includes 2D for digit classification on MNIST and 3D for object classification on ModelNet40 [52]. These three experiments illustrate that our attentional method performs better than

Outlier ratio	60%	70%	80%	85%	90%
CNe [55]	.00019	.0038	.056	.162	.425
ACNe (Ours)	**1e-6**	**.0008**	**.024**	**.130**	**.383**

Table 1. **Robust line fitting** – Line fitting results over the test set in terms of the ℓ_2 distance between ground-truth and the estimates.

Outlier ratio	0%	10%	20%	30%	40%	50%	60%
PointNet [36]	98.1	95.1	93.2	79.5	67.7	70.0	54.8
CNe [55]	98.0	95.8	94.0	91.0	90.1	87.7	87.2
ACNe (Ours)	**98.3**	**97.2**	**96.5**	**95.3**	**94.7**	**94.3**	**93.7**

Table 2. **2D Point cloud classification** – Classification accuracy on MNIST, under different outlier ratios (%). Our method performs best in all cases, and the gap becomes wider with more outliers.

vanilla Context Normalization under the presence of outliers. We then apply our solution to wide-baseline stereo, and demonstrate that this increase in performance holds on challenging real-world applications, and against state-of-the-art methods for robust pose estimation. Finally, we perform an ablation study and evaluate the effect of supervising the weights used for attention in stereo.

6.1. Robust line fitting – Fig. 1 and Table 1

To generate 2D points on a random line, as well as outliers, we first sample 2D points uniformly within the range $[-1, +1]$. We then select two points randomly and fit a line that goes through them. With probability according to the desired inlier ratio, we then project each point onto the line to form inliers. We measure the error in terms of the ℓ_2 distance between the estimated and ground truth values for the line parameters. The results are summarized in Table 1, with qualitative examples in Fig. 1. ACNe consistently outperforms CNe [55]. Both methods break down at a 85-90% outlier ratio, while the performance of ACNe degrades more gracefully. As illustrated in Fig. 1, our method learns to progressively focus on the inliers throughout the different layers of the network and weeds out the outliers.

6.2. Classifying digits – Fig. 3 and Table 2

We evaluate our approach on handwritten digit classification on MNIST, which consists of 28×28 grayscale images. We create a point cloud from these images following the procedure of [36]: we threshold each image at 128 and use the coordinates – normalized to a unit bounding box – of the surviving pixel locations as data samples. We subsample 512 points with replacement, in order to have the same number of points for all training examples. We also add a small Gaussian noise of 0.01 to the pixel coordinates after sampling following [36]. Outliers are generated by sampling from a uniform random distribution. We compare our method against vanilla PointNet [36] and CNe [55]. For PointNet, we re-implemented their method under our framework to have an identical training setup.

Outlier ratio	0%	10%	20%	30%	40%	50%
PointNet	85.8	81.7	81.7	80.1	78.2	78.5
PointNet w/ CN	87.2	84.3	84.5	83.4	81.7	81.6
PointNet w/ ACN	**87.7**	**84.6**	**85.0**	**84.6**	**83.3**	**84.2**

Table 3. **3D Point cloud classification –** We replicate the 3D point classification experiment on ModelNet40 from [36], with *vanilla* PointNet. We then add outliers with Gaussian noise. Our approach performs best with and without outliers.

Table 2 summarizes the results in terms of classification accuracy. Our method performs best, with the gap widening as the outlier ratio increases – while CNe shows some robustness to noise, PointNet quickly breaks down. Note that the results for PointNet are slightly different from the ones reported in [36], as we use a validation split to perform early stopping. In addition, to reduce randomness, we train 10 different models and report the average results.

6.3. Classifying 3D objects – Table 3

We apply our method to the problem of 3D object (point cloud) classification. We use the ModelNet40 dataset [52], and compare with PointNet [37]. Similarly to the MNIST case, we contaminate the dataset with outliers to test the robustness of each method. Specifically, we add a predetermined ratio of outliers to the point clouds, sampled uniformly within the range $[-1, 1]$. We also add small Gaussian perturbations to the locations of the points, with a standard deviation of 0.01. We then sample 1024 points from the point cloud to perform classification. Again, to simply test if ACN can improve existing pipelines, we plug our normalization into the vanilla PointNet architecture. Note that the original PointNet includes an affine estimation step which provides a small performance boost – we omit it from our implementation, in order to isolate the architectural differences between the methods. We report the results in Table 3. Our method performs best, with the gap becoming wider as outliers become prevalent.

6.4. Wide-baseline stereo – Fig. 4 and Table 4

Wide-baseline stereo is an extremely challenging problem, due to the large number of variables to account for – viewpoint, scale, illumination, occlusions, and properties of the imaging device – see Fig. 4 for some examples. We benchmark our approach on a real-world dataset against multiple state-of-the-art baselines, following the data [58] and protocols provided by [55]. Their ground truth camera poses are obtained from Structure-from-Motion with VisualSfM [50], from large image collections of publicly available, challenging photo-tourism data.

We evaluate performance in terms of the *reconstructed poses*. Since the stereo matching problem is defined only up to a scale factor [18], it is not possible to compute absolute (metric) errors for translation. Instead, we follow

Figure 4. **Wide-baseline stereo** – We show the results of different matching algorithms on the dataset of [55]. We draw the *inliers* produced by them, in **green** if the match is below the epipolar distance threshold (in **red** otherwise). Note that this may include some false positives, as epipolar constraints map points to lines – perfect ground truth would require dense pixel-to-pixel correspondences.

the methodology of [55] and measure the error between the ground truth and estimated vectors between both cameras, for both rotation and translation, and combine them by taking the maximum of the two. We then evaluate the accuracy over all image pairs at multiple error thresholds, accumulate it up to a limit (either 10° or 20°), and summarize performance by its mean – which we call the mean Average Precision (mAP); see [55]. This means that methods that perform better at lower error thresholds are rewarded. We use their data mostly as is, using the pre-extracted correspondences and splits from OANet [58], but adapt it to the Fundamental matrix problem. In contrast to previous works [55, 58], which report results on the scene the models were trained on, we focus on unknown scenes, in order to determine each method's *actual* performance.

As we discussed in Section 4.3, both CNe [55] and OANet [58] assume known camera intrinsics and estimate the Essential matrix, instead of the Fundamental matrix – this is a *significantly* easier problem, as the number of free parameters drops from 7 to 5. However, most research papers on this topic focus on estimating the Fundamental matrix [9, 10, 38, 3, 4], which is why we focus on this problem instead. For completeness, we also report results for the Essential matrix in the supplementary appendix, for which we also achieve state-of-the-art results.

In more detail, given an image pair, we extract 2k keypoints for each image with SIFT [31]. Matches are then formed from one image to the other, in both directions. As is typical for image matching, we then filter out non-discriminative correspondences via bi-directional check, enforcing one-to-one matching. For RANSAC variants we found it to be critical to further apply Lowe's ratio test [31] – without it RANSAC variants provide worse results. We apply it with a ratio threshold of 0.8. We do not apply this test for learned methods, as it throws out too many inliers for learned methods to bring any benefit. Also, when training learned methods, we train without bidirectional check to show as many correspondences in the training set as possible to the network.

We consider the following methods: LMedS [40], RANSAC [16, 9], MLESAC [45], DegenSAC [10], MAGSAC [4] CNe [55], DFE [39], OANet [58] and ACNe (ours). We consider the pose estimated with the weighted 8-point algorithm directly, as well as those combined with a robust outlier rejection method as outlined in Section 5.

Quantitative results. We report quantitative results in Table 4, for two different error thresholds (10°/20°). We make three fundamental observations:

(1) Our method consistently outperforms all of the baselines, including CNe and OANet. The difference in performance between ACNe and its closest competitor, OANet, is of $14.1/9.8\%$ relative for Outdoors, and $8.6/9.2\%$ relative for Indoors when used without any additional post processing. The gap for Outdoors is reduced to 1% when they are combined with MAGSAC, but ACNe still outperforms OANet. For Indoors, we observe a drop in performance for both OANet and ACNe when combining them with RANSAC or MAGSAC. The margin between learned and traditional methods is significant, with ACNe performing $30.1/39.6\%$ better relative on Outdoors and $45.5/47.1\%$ better relative on Indoors, compared to the best performing traditional baseline – including a very recent method, MAGSAC.

(2) Different from the findings of [55], we observe that RANSAC variants may harm performance, particularly with ACNe. This is because through its global attention – $\mathbf{w}^{\text{global}}$ – ACNe can infer the relative importance of each correspondence, which is not easily taken into account when passing samples to a robust estimator. In this manner, ACNe goes *beyond* simple outlier rejection. The best performance is typically achieved by using ACNe at its pure form, directly feeding its weights to the weighted 8-point algorithm. Given that all our experiments are on unseen sequences, this further shows that ACNe generalizes very well, even without being followed by an additional robust estimator.

(3) Contrary to the results of Yi *et al.* [55] and Zhang *et al.* [58], we find that traditional baselines perform better than reported on either work. This is because their exper-

	Method	Outdoors	Indoors
Traditional	LMedS	.296/.383	.142/.235
	RANSAC	.356/.437	.172/.272
	MLESAC	.148/.216	.135/.230
	DegenSAC	.328/.394	.191/.291
	MAGSAC	.385/.457	.185/.282
Learned	CNe (weighted-8pt)	.323/.469	.189/.331
	CNe+RANSAC	.449/.554	.201/.315
	CNe+MAGSAC	.500/.598	.213/.326
	DFE (weighted-8pt)[3]	.319/.470	.167/.294
	DFE+RANSAC	.414/.508	.193/.303
	DFE+MAGSAC	.452/.541	.211/.320
	OANet (weighted-8pt)	.439/.581	<u>.256</u>/<u>.392</u>
	OANet+RANSAC	.482/.592	.211/.331
	OANet+MAGSAC	<u>.514</u>/.615	.230/.346
Ours	ACNe (weighted-8pt)	.501/**.638**	**.278**/**.428**
	ACNe+RANSAC	.478/.590	.209/.329
	ACNe+MAGSAC	**.518**/<u>.621</u>	.226/.343

Table 4. **Pose estimation accuracy** – mAP at 10°/20° error threshold. Similarly to [55], we consider multiple baselines, as well as pairing different methods with state-of-the-art RANSAC variants. Our method consistently outperforms all others by a significant margin, even without an additional robust estimator, in some cases.

imental setup did not consider Lowe's ratio test, nor the bidirectional check. Without these, the performance of traditional baselines drops drastically – RANSAC and MAGSAC drop $79.2/73.0\%$ and $92.0/85.1\%$ relative performance, respectively, for Outdoors, and $66.9/59.2\%$ and $82.2/74.1\%$ for Indoors.

Ablation study – Table 5. We perform an ablation study to evaluate the effect of the different types of attention, as well as the supervision on the local component of the attentive mechanism. We also compare with CNe, as its architecture is the most similar to ours. We use the train and validation splits for the *Saint Peter's Square* sequence for this study, as it is the primary sequence used for training in [55] and has many images within the set. (1) We confirm that CNe [55] performs better with Batch Normalization (BN) [22] than with Group Normalization (GN) [51] – we use GN for ACNe, as it seems to perform marginally better with our attention mechanism. (2) We observe that our attentive mechanisms allow ACNe to outperform CNe, and that their combination outperforms their separate use. (3) Applying supervision on the weights further boosts performance.

With learned features – Table 6. Finally, we report that our method also works well with two state-of-the-art, learned local feature methods – SuperPoint [14] and LF-Net [35]. They are learned end-to-end – their characteristics are thus different from those of SIFT keypoints. We test again on *Saint Peter's Square*, as our primary focus is to show that it is possible to use other feature types. In Table 6 we report that

[3]DFE, when tested with bidirectional check, gives poor results – .236 for Outdoors (20°). We hypothesize this may be due to the fact that DFE uses additional, "side" information, whose statistics may vary drastically with the bidirectional check. We therefore do not use it for this baseline. This problem disappears when additional robust estimators are used.

Methods	CNe [55]		ACNe (Ours)			
	w/ BN	w/ GN	L	G	L+G	L+G+S
Weighted-8pt	.435	.414	.531	.593	.597	**.602**

Table 5. **Ablation study** – mAP at 20° with different CNe [55] and ACNe (ours) variants on stereo. The labels indicate: *L* – local attention; *G* – global attention; *S* – local attention supervision.

	SIFT	SuperPoint	LF-Net
MAGSAC (w/o ratio test)	.146	.205	.134
MAGSAC	.264	.230	.157
OANet (weighted 8pt)	<u>.488</u>	<u>.547</u>	<u>.543</u>
OANet+MAGSAC	.479	.442	.452
ACNe (weighted 8pt)	**.602**	**.637**	**.619**

Table 6. **With learned local features –** mAP at 20° with learned local features and different methods. Our method outperforms other methods and performs best with SuperPoint [14].

both methods improve performance over SIFT with OANet and ACNe, but with ratio test and MAGSAC they perform worse. It is interesting how SuperPoint, without the ratio test, performs better than SIFT with MAGSAC, but the order is reversed when the ratio test is introduced, highlighting its importance. Regardless of feature type, we demonstrate that our approach provides improved performance over other methods, and that it pairs best with SuperPoint.

7. Conclusion

We have proposed *Attentive Context Normalization* (ACN), and used it to build *Attentive Context Networks* (ACNe) to solve problems on permutation-equivariant data. Our solution is inspired by IRLS, where one iteratively re-weighs the importance of each sample, via a soft inlier/outlier assignment. We demonstrated that by learning both local and global attention we are able to outperform state-of-the-art solutions on line fitting, classification of point clouds in 2D (digits) and 3D (objects), and challenging wide-baseline stereo problems. Notably, our method *thrives* under large outlier ratios. For future research directions, we consider incorporating ACN into general normalization techniques for deep learning. We believe that this is an interesting direction to pursue, as all existing techniques make use of statistical moments.

Acknowledgements

This work was supported by the Natural Sciences and Engineering Research Council of Canada (NSERC) Discovery Grant, NSERC Collaborative Research and Development Grant (Google), and by Compute Canada.

References

[1] Sameer Agarwal, Noah Snavely, Ian Simon, Steven M. Seitz, and Richard Szeliski. Building Rome in One Day. In *Int. Conf. on Comput. Vis.*, 2009. 1

[2] Jimmy L. Ba, Jamie R. Kiros, and Geoffrey E. Hinton. Layer Normalization. *arXiv Preprint*, 2016. 1, 2, 3

[3] Daniel Barath and Jiří Matas. Graph-cut RANSAC. In *Conf. on Comput. Vis. Pattern Recognit.*, 2018. 7

[4] Daniel Barath, Jana Noskova, and Jiří Matas. MAGSAC: Marginalizing Sample Consensus. In *Conf. on Comput. Vis. Pattern Recognit.*, 2019. 5, 7

[5] JiaWang Bian, Wen-Yan Lin, Yasuyuki Matsushita, and Sai-Kit Yeungand Tan-Dat Nguyena nd Ming-Ming Cheng. GMS: Grid-Based Motion Statistics for Fast, Ultra-Robust Feature Correspondence. In *Conf. on Comput. Vis. Pattern Recognit.*, 2017. 2

[6] Sofien Bouaziz, Andrea Tagliasacchi, Hao Li, and Mark Pauly. Modern Techniques and Applications for Real-Time Non-rigid Registration. In *SIGGRAPH Asia (Technical Course Notes)*, 2016. 1

[7] Sofien Bouaziz, Andrea Tagliasacchi, and Mark Pauly. Sparse Iterative Closest Point. In *Comput. Graphics Forum*, 2013. 4

[8] Rick Chartrand and Wotao Yin. Iteratively Reweighted Algorithms for Compressive Sensing. In *Int. Conf. on Acoustics, Speech, Signal Process.*, 2008. 2

[9] Ondřej Chum, Jiří Matas, and Josef Kittler. Locally Optimized RANSAC. In *Joint Pattern Recognition Symposium*, 2003. 5, 7

[10] Ondrej Chum, Tomas Werner, and Jiri Matas. Two-view Geometry Estimation Unaffected by a Dominant Plane. In *Conf. on Comput. Vis. Pattern Recognit.*, 2005. 5, 7

[11] Andrew Cotter, Maya Gupta, Heinrich Jiang, Erez Louidor, James Muller, Tamann Narayan, Serena Wang, and Tao Zhu. Shape constraints for set functions. In *Int. Conf. on Mach. Learn.*, pages 1388–1396, 2019. 2

[12] Zheng Dang, Kwang Moo Yi, Yinlin Hu, Fei Wang, Pascal Fua, and Mathieu Salzmann. Eigendecomposition-Free Training of Deep Networks with Zero Eigenvalue-Based Losses. In *European Conf. on Comput. Vis.*, 2018. 1, 2, 5

[13] Michaël Defferrard, Xavier Bresson, and Pierre Vandergheynst. Convolutional Neural Networks on Graphs with Fast Localized Spectral Filtering. In *Adv. Neural Inf. Process. Syst.*, 2016. 2

[14] Daniel Detone, Tomasz Malisiewicz, and Andrew Rabinovich. Superpoint: Self-Supervised Interest Point Detection and Description. *CVPR Workshop on Deep Learning for Visual SLAM*, 2018. 8

[15] Mihai Dusmanu, Ignacio Rocco, Tomas Pajdla, Marc Pollefeys, Josef Sivic, Akihiko Torii, and Torsten Sattler. D2-Net: A Trainable CNN for Joint Detection and Description of Local Features. *Conf. on Comput. Vis. Pattern Recognit.*, 2019. 2, 11

[16] Martin A. Fischler and Robert C. Bolles. Random Sample Consensus: A Paradigm for Model Fitting with Applications to Image Analysis and Automated Cartography. *Communications ACM*, 24(6):381–395, 1981. 1, 2, 5, 7

[17] John Fox. *An R and S-Plus Companion to Applied Regression*. Sage, 2002. 2

[18] Richard Hartley and Andrew Zisserman. *Multiple View Geometry in Computer Vision*. Cambridge University Press, 2000. 5, 6

[19] H K. Hartline, Henry G. Wagner, and Floyd Ratliff. Inhibition in the Eye of Limulus. *Journal of General Psychology*, 1956. 3

[20] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep Residual Learning for Image Recognition. In *Conf. on Comput. Vis. Pattern Recognit.*, 2016. 4

[21] Xun Huang and Serge Belongie. Arbitrary Style Transfer in Real-time with Adaptive Instance Normalization. In *Int. Conf. on Comput. Vis.*, 2017. 3

[22] Sergey Ioffe and Christian Szegedy. Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift. In *Int. Conf. on Mach. Learn.*, 2015. 1, 2, 3, 4, 8

[23] Max Jaderberg, Karen Simonyan, Andrew Zisserman, and Koray Kavukcuoglu. Spatial Transformer Networks. In *Adv. Neural Inf. Process. Syst.*, 2015. 3

[24] Kevin Jarrett, Koray Kavukcuoglu, Marc A. Ranzato, and Yann LeCun. What is the Best Multi-Stage Architecture for Object Recognition? In *Int. Conf. on Comput. Vis.*, 2009. 3

[25] Diederik P. Kingma and Jimmy Ba. Adam: A Method for Stochastic Optimisation. In *Int. Conf. on Learn. Representations*, 2015. 5

[26] Thomas N. Kipf and Max Welling. Semi-Supervised Classification with Graph Convolutional Networks. *Int. Conf. on Learn. Representations*, 2017. 2

[27] Florian Kluger, Eric Brachmann, Hanno Ackermann, Carsten Rother, Michael Ying Yang, and Bodo Rosenhahn. CONSAC: Robust Multi-Model Fitting by Conditional Sample Consensus. *CVPR*, 2020. 1

[28] Johannes Kopf, Michael F Cohen, and Richard Szeliski. First-Person Hyper-Lapse Videos. *ACM Trans. on Graphics*, 33(4):78, 2014. 1

[29] Alex Krizhevsky, Ilya Sutskever, and Geoffery E. Hinton. ImageNet Classification with Deep Convolutional Neural Networks. In *Adv. Neural Inf. Process. Syst.*, 2012. 3

[30] Xinhai Liu, Zhizhong Han, Yu-Shen Liu, and Matthias Zwicker. Point2Sequence: Learning the Shape Representation of 3D Point Clouds with an Attention-Based Sequence to Sequence Network. *Amer. Assoc. for Artif. Intell. Conf.*, 2019. 2

[31] David G. Lowe. Distinctive Image Features from Scale-Invariant Keypoints. *Int. J. Comput. Vis.*, 20(2):91–110, 2004. 1, 7

[32] Minh-Thang Luong, Hieu Pham, and Christopher D. Manning. Effective Approaches to Attention-based Neural Machine Translation. *Empirical Methods in Nat. Language Process.*, 2015. 3

[33] Muhammad J Mirza and Kim L Boyer. Performance Evaluation of a Class of M-estimators for Surface Parameter Estimation in Noisy Range Data. *IEEE Transactions on Robotics and Automation*, 9(1):75–85, 1993. 2

[34] Takeru Miyato, Toshiki Kataoka, Masanori Koyama, and Yuichi Yoshida. Spectral Normalization for Generative Adversarial Networks. In *Int. Conf. on Learn. Representations*, 2018. 3

[35] Yuki Ono, Eduard Trulls, Pascal Fua, and Kwang Moo Yi. Lf-Net: Learning Local Features from Images. In *Adv. Neural Inf. Process. Syst.*, 2018. 8

[36] Charles R. Qi, Hao Su, Kaichun Mo, and Leonidas J. Guibas. Pointnet: Deep Learning on Point Sets for 3D Classification and Segmentation. In *Conf. on Comput. Vis. Pattern Recognit.*, 2017. 1, 2, 5, 6

[37] Charles R. Qi, Li Yi, Hao Su, and Leonidas J. Guibas. Pointnet++: Deep Hierarchical Feature Learning on Point Sets in a Metric Space. In *Adv. Neural Inf. Process. Syst.*, 2017. 1, 2, 5, 6

[38] Rahul Raguram, Ondrej Chum, Marc Pollefeys, Jiri Matas, and Jan-Michael Frahm. USAC: a Universal Framework for Random Sample Consensus. *IEEE Trans. on Pattern Anal. Mach. Intell.*, 35(8):2022–2038, 2012. 7

[39] René Ranftl and Vladlen Koltun. Deep Fundamental Matrix Estimation. In *European Conf. on Comput. Vis.*, 2018. 1, 2, 7

[40] Peter J. Rousseeuw. Least Median of Squares Regression. *Journal of the American Statistical Association*, 1984. 7

[41] Tim Salimans and Diederik P. Kingma. Weight Normalization: A Simple Reparameterization to Accelerate Training of Deep Neural Networks. In *Adv. Neural Inf. Process. Syst.*, 2016. 3

[42] Yiru Shen, Chen Feng, Yaoqing Yang, and Dong Tian. Mining Point Cloud Local Structures by Kernel Correlation and Graph Pooling. In *Conf. on Comput. Vis. Pattern Recognit.*, pages 4548–4557, 2018. 2

[43] Noah Snavely, Rahul Garg, Steven M Seitz, and Richard Szeliski. Finding Paths Through the World's Photos. *ACM Trans. on Graphics*, 2008. 1

[44] Maxim Tatarchenko, Jaesik Park, Vladlen Koltun, and Qian-Yi Zhou. Tangent Convolutions for Dense Prediction in 3D. In *Conf. on Comput. Vis. Pattern Recognit.*, 2018. 2

[45] Philip H.S. Torr and Andrew Zisserman. MLESAC: A New Robust Estimator with Application to Estimating Image Geometry. *Comput. Vis. Image Understanding*, 78:138–156, 2000. 2, 5, 7

[46] Dmitry Ulyanov, Andrea Vedaldi, and Victor Lempitsky. Instance Normalization: The Missing Ingredient for Fast Stylization. *arXiv Preprint*, 2016. 1, 2, 3

[47] Benjamin Ummenhofer, Huizhong Zhou, Jonas Uhrig, Nikolaus Mayer, Eddy Ilg, Alexey Dosovitskiy, and Thomas Brox. DeMoN: Depth and Motion Network for Learning Monocular Stereo. In *Conf. on Comput. Vis. Pattern Recognit.*, 2017. 1

[48] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention Is All You Need. In *Adv. Neural Inf. Process. Syst.*, 2017. 3

[49] Fei Wang, Mengqing Jiang, Chen Qian, Shuo Yang, Cheng Li, Honggang Zhang, Xiaogang Wang, and Xiaoou Tang. Residual Attention Network for Image Classification. In *Conf. on Comput. Vis. Pattern Recognit.*, 2017. 2, 11

[50] Changchang Wu. Towards Linear-Time Incremental Structure from Motion. In *3DV*, 2013. 6

[51] Yuxin Wu and Kaiming He. Group Normalization. In *European Conf. on Comput. Vis.*, 2018. 1, 2, 3, 4, 5, 8

[52] Zhirong Wu, Shuran Song, Aditya Khosla, Fisher Yu, Linguang Zhang, Xiaoou Tang, and Jianxiong Xiao. 3D ShapeNets: A Deep Representation for Volumetric Shapes. In *Conf. on Comput. Vis. Pattern Recognit.*, 2015. 5, 6

[53] Saining Xie, Sainan Liu, Zeyu Chen, and Zhuowen Tu. Attentional ShapeContextNet for Point Cloud Recognition. In *Conf. on Comput. Vis. Pattern Recognit.*, 2018. 3

[54] Kelvin Xu, Jimmy Ba, Ryan Kiros, Kyunghyun Cho, Aaron Courville, Ruslan Salakhutdinov, Richard Zemel, and Yoshua Bengio. Show, Attend and Tell: Neural Image Caption Generation with Visual Attention. *Int. Conf. on Mach. Learn.*, 2015. 3

[55] Kwang Moo Yi, Eduard Trulls, Yuki Ono, Vincent Lepetit, Mathieu Salzmann, and Pascal Fua. Learning to Find Good Correspondences. In *Conf. on Comput. Vis. Pattern Recognit.*, 2018. 1, 2, 3, 4, 5, 6, 7, 8, 11, 12

[56] Manzil Zaheer, Satwik Kottur, Siamak Ravanbakhsh, Barnabas Poczos, Ruslan R Salakhutdinov, and Alexander J Smola. Deep Sets. In *Adv. Neural Inf. Process. Syst.*, 2017. 2

[57] Amir R. Zamir, Tilman Wekel, Pulkit Argrawal, Colin Weil, Jitendra Malik, and Silvio Savarese. Generic 3D Representation via Pose Estimation and Matching. In *European Conf. on Comput. Vis.*, 2016. 1

[58] Jiahui Zhang, Dawei Sun, Zixin Luo, Anbang Yao, Lei Zhou, Tianwei Shen, Yurong Chen, Long Quan, and Hongen Liao. Learning Two-View Correspondences and GeometryUsing Order-Aware Network. In *Int. Conf. on Comput. Vis.*, 2019. 2, 5, 6, 7, 12

[59] Chen Zhao, Zhiguo Cao, Chi Li, Xin Li, and Jiaqi Yang. NM-Net: Mining Reliable Neighbors for Robust Feature Correspondences. *Conf. on Comput. Vis. Pattern Recognit.*, 2019. 1, 2, 5

[60] Tinghui Zhou, Matthew Brown, Noah Snavely, and David Lowe. Unsupervised Learning of Depth and Ego-Motion from Video. In *Conf. on Comput. Vis. Pattern Recognit.*, 2017. 1

[61] Yin Zhou and Oncel Tuzel. VoxelNet: End-to-End Learning for Point Cloud Based 3D Object Detection. In *Conf. on Comput. Vis. Pattern Recognit.*, 2018. 2