ACNe: Attentive Context Normalization for Robust Permutation-Equivariant Learning

(Supplementary Material)

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

A. Visualizing attention – Fig. 5

We visualize the weights for wide-baseline stereo, along with the "ground truth" labels on the matches. Since the labels are obtained by thresholding the epipolar distance, computed from the ground truth poses, they contain a few false positives. This figure shows that ACN learns to focus on inliers by emulating a robust iterative optimization. As our system is trained by Fundamental matrix supervision, it was also able to learn to ignore these false positives.

Methods	Attn. on feature map			Our method		
	L	G	L+G	L	G	L+G
Weighted-8pt	.410	.260	.408	.531	.593	**.597**
Weighted-8pt (ReLU+Tanh)	.427	.347	.369	—	—	—

Table 7. **Applying attention to the feature maps –** We compare our method (right) to applying attention directly to the feature maps (left). We report mAP at a 20° error threshold on our validation set – the *Saint Peter's Square* scene. Applying attention on the normalization performs significantly better than applying attention to the feature maps.

B. Attention on feature maps – Table 7

We show that applying attention to the *normalization* of the feature maps (our method) outperforms the more commonly used strategy of applying attention directly to the feature maps [49, 15]. Table 7 extends our ablation study from Table 5, demonstrating that our method outperforms this alternative approach by 29% relative.

It is important to note that introducing global attention to the feature maps resulted in *unstable* training. To avoid gradient explosion we reduced the learning rate to one-tenth of the value we typically used. Nonetheless, gradients exploded after 224k iterations. We suspect that attention on feature maps causes the feature maps to become artificially small, resulting in numerical instability. In all cases, the performance is slightly worse than what CNe [55] gives in Table 5 (.414) showing that attention on feature maps is actually harmful.

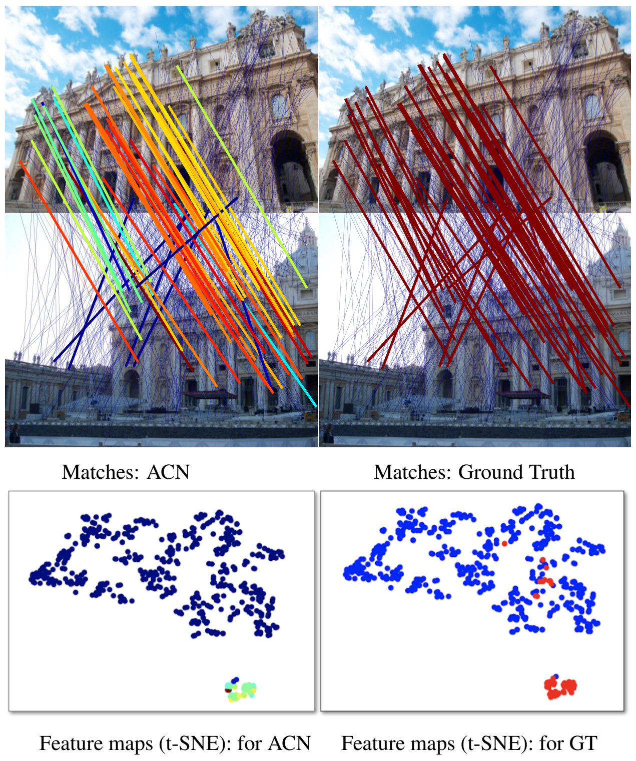

Figure 5. **Attention in wide-baseline stereo.** An illustration of ACNe on an example image pair. The top row shows the *matches*, and the bottom row shows a representation of the *feature maps* obtained via t-SNE. The left column displays ACN weights, color-coded by magnitude (highest in **red**, lowest in **blue**), and the right column the ground truth match labels (inliers in **red**, outliers in **blue**), computed by thresholding the symmetric epipolar distance (Sec. 4.3). We draw matches with negative labels with thinner lines.

We also tried modifying the output of the network – *i.e.*, the weights used by the eight-point algorithm – to use the ReLU+Tanh configuration from [55], which we report in the bottom row of Table 7. This variant trained in a stable way, but provided sub-par results that are always lower than using local attention only on the normalization. Note that with ReLU+Tanh and local attention only, attention on the feature maps does help – by 1% relative – but the increase in performance is very small compared to what our method can achieve.

C. Essential matrix estimation – Table 8

Several learned methods [55, 58] focus on estimating the Essential matrix instead of the Fundamental matrix. The latter is more broadly applicable as it does not need a-priori knowledge about the intrinsics of the camera – hence it is closer to Computer Vision "in the wild".

We now demonstrate that our approach outperforms these methods for Essential matrix estimation as well. We report the results in Table 8, using the authors' original implementations for this comparison. We also report the performance of robust estimators such as RANSAC and MAGSAC. For RANSAC, we rely on `findEssentialMat` from OpenCV. We found that it is beneficial to use both local and global attention when applying *SAC to the Essential matrix problem unlike the Fundamental matrix problem, and we simply threshold $\mathbf{w}$ with an optimal threshold (i.e., 10^{-7}) found on the validation set, and feed the surviving correspondences to *SAC. We observe that RANSAC improves the performance for ACNe in the outdoor experiments. This is due to the reduced complexity of the problem, which assumes known camera intrinsics. For MAGSAC, we carefully implement the 5-point algorithm into their framework for estimating the Essential matrix. While it achieves competitive results, MAGSAC is still much worse than RANSAC because it is originally geared for Fundamental matrix estimation.

Method	Outdoors	Indoors
RANSAC	.671	.365
MAGSAC	.415	.204
CNe (weighted-8pt)	.515	.332
CNe+RANSAC	.750	.404
CNE+MAGSAC	.514	.259
DFE (weighted-8pt)	.573	.352
DFE+RANSAC	.721	.384
DFE+MAGSAC	.532	.265
OANet (weighted-8pt)	.648	.401
OANet+RANSAC	<u>.776</u>	<u>.419</u>
OANet+MAGSAC	.547	.271
ACNe (weighted-8pt)	.706	**.429**
ACNe+RANSAC	**.780**	.418
ACNe+MAGSAC	.415	.187

Table 8. **Essential matrix estimation –** mAP at a 20° error threshold when we train the models to estimate the Essential matrix instead of the Fundamental matrix, for the indoors and outdoors experiments.

D. Number of ACNs within ARB – Table 9

We perform an ablation study to evaluate the impact of the number of ACN blocks within each ARB. Due to the increasing computation overhead and GPU memory limitations, we only report the results of ACNe up to three ACN blocks for each ARB; see Table 9. We expect that more ACN blocks would further improve the accuracy, at the cost of an increase in memory/computation. We use 2 blocks, as it provides a good compromise between computational requirements and performance, and also the additional advantage that this makes our results directly comparable to CNe [55].

#ACN per ARB	1	2	3
Weighted-8pt	.527	.602	.621

Table 9. **Ablation on #ACN** – Performance as we vary the number of ACNs within each ARB – mAP at 20° on our validation set – *Saint Peter's Square*.

E. Number of parameters – Table 10

Even though ACNe significantly outperforms CNe, the number of parameters added to CNe is only 6K, which is only $\approx$1.5% more. The advantages are more prominent when we compare against OANet, which introduces a *significant* increase in the number of parameters in the network, while providing worse results.

Methods	OANet	CNe	ACNe
# of Parameters	2347K	394K	400K

Table 10. **Number of parameters –** Our method introduces a small overhead compared to CNe, and is much smaller than OANet.

F. Timing of *SAC Methods – Table 11

We observed that ACNe and CNe share a similar runtime, and are both more efficient than OANet, which uses a deep permutation-equivariant network that performs an iterative refinement of an initial guess. Additionally, due to the GPU efficiency and low computational complexity, the runtime of learned methods is negligible compared with traditional robust estimators (*SAC). Furthermore, learning-based methods are capable of facilitating the task of a robust estimator by proactively rejecting outliers. For instance, we found that ACNe makes RANSAC approximately 12$\times$ times and MAGSAC approximately 5$\times$ faster, while also significantly improving overall performance; see Table 4.

Methods	Network	Robust estimator
RANSAC	—	194
MAGSAC	—	2752
CNe	15	—
CNe + RANSAC	15	19
CNe + MAGSAC	15	523
OANet	18	—
OANet + RANSAC	18	13
OANet + MAGSAC	18	546
ACNe	14	—
ACNe + RANSAC	14	16
ACNe + MAGSAC	14	594

Table 11. **Average elapsed time –** Runtime, in milliseconds, for individual steps of each method. We execute the forward pass of our networks on a GTX 1080 Ti GPU and the robust estimator on a Intel(R) Core(TM) i7-8700 CPU. We disable multi-threading for the CPU timings since not all robust estimator implementations support multi-threading. Networks are implemented with TensorFlow 1.8.0, except for OANet, which uses PyTorch 1.2.0.

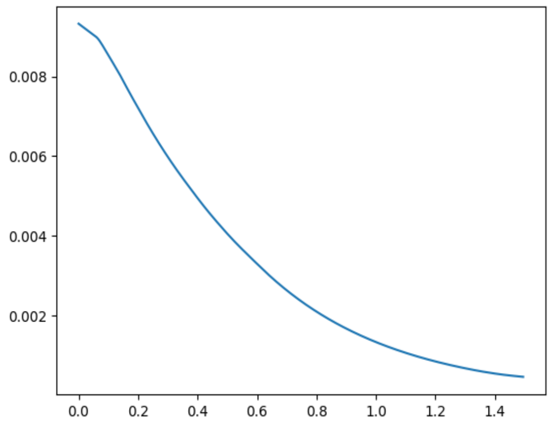

Figure 6. **Output weight computed by the 2-layer MLP for a given residual input –** Computed for the *residual-to-weight* variant of ACNe.

Method	CNe	IRLS	ACNe
ℓ_2 error	.0038	.0024	**.0008**

Table 12. **Performance of learnt *IRLS* –** Comparison with CNe and ACNe on the line fitting problem, under an outlier ratio of 0.7.

G. Learnt IRLS variant – Fig. 6, Table 12

Our method is *inspired* by iteratively re-weighted least squares (IRLS), and not a direct translation. We learn a representations-to-weights mapping, and not a residuals-to-weights mapping as in traditional IRLS; see (4). To ensure the validity of our approach, we compare to a variant of our method that is more faithful to IRLS. In each (unrolled) iteration, we compute residuals by solving for the final objective (*e.g*. the optimal line parameters) with the current weights. Then, we represent $\psi(.)^{-1}$ in (4) with a 2-layer MLP that is *shared* between iterations. In other words, the MLP serves as a *learnt kernel*, which is traditionally hand-picked in the IRLS literature. We evaluate performance on the line fitting example due to its simplicity.

As shown in Table 12, the *residuals-to-weights* variant performs significantly worse than ACNe. This is not surprising, given that the IRLS variant is more restricted in what in can do, compared to ACNe. However, it is interesting to note that it still performs better than classical CNe. One very interesting aspect is that the learnt $\psi(.)^{-1}$ has the typical monotonically decreasing property of typical (M-Estimator) kernel functions; see Fig. 6.