FAR: Flexible, Accurate and Robust 6DoF Relative Camera Pose Estimation: Supplemental Material

Chris Rockwell[1] Nilesh Kulkarni[1] Linyi Jin[1]
Jeong Joon Park[1] Justin Johnson[1] David F. Fouhey[2]
University of Michigan[1] New York University[2]

This Supplement includes additional detail for the method and experiments, as well as additional experiments and explanation too long for the main paper.

1. Network Architecture

We detail the full model along with ablations below as a function of their components; these correspond to predicted pose boxes in Figure 3 of the paper. Architecture is overviewed in Tables 1-2 and detailed in Tables 3-7.

Solver $\mathbf{T}_s$. Pose estimation from a correspondence estimator followed by a Kornia [4] implementation of RANSAC + 5-Point Algorithm, optionally scaled by predicted translation scale. Compare to FAR: Full $\mathbf{T}$, this ablation does not use the Transformer, nor combine Solver and Transformer predictions, nor do a second round of prior-guided solver and combining with Transformer.

We refer to this as "Solver" if it uses perturbed ground truth correspondences as input (Figure 4), meaning no correspondence estimator is used. We refer to this as Corr. + Solver in experiments if correspondence estimator is used (Figure 6 and 8 from paper). We refer to it as Corr. + Solver + Scale if predicted scale is used to evaluate absolute translation error (Figures 1, 2, 7; Table 2); we refer to it as LoFTR + Solver + Scale if LoFTR is used (Table 2).

Predicted scale for Solver $\mathbf{T}_s$ is the output of the Translation Scale Predictor network detailed in Table 7. It takes dense features f as input and outputs a single scalar, which is multiplied by translation angle output from RANSAC + 5-Point to obtain final translation. Early in experiments, we used a transformer-based architecture to predict scale, but found this CNN-based method performed better.

FAR: Transformer $\mathbf{T}_t$. Predicted 6DoF pose from FAR's Transformer. Compare to FAR: Full $\mathbf{T}$, this ablation does not use the Solver, nor combine Solver and Transformer predictions, nor do a second round of prior-guided solver and combining with Transformer.

In the case dense features are available, the 8-Point ViT is used, if only correspondences plus descriptors are available, the Vanilla TF is used. Each is detailed in Table 3 and Table 4, respectively.

FAR: One Round $\mathbf{T}_1$. Predicted 6DoF pose from one round of FAR, which consists of the weighted linear combination in 6D [8] space of $\mathbf{T}_t$ and $\mathbf{T}_s$, weighted by w as described in Equation 1 from the paper. Compare to FAR: Full $\mathbf{T}$, this ablation does not do a second round of prior-guided solver and combining with Transformer.

FAR: Updated $\mathbf{T}_u$. Pose estimation from FAR's prior-guided RANSAC + 5-Point Algorithm, using $\mathbf{T}_1$ as a prior. Compare to FAR: Full $\mathbf{T}$, this ablation does not do a second round of combining with the Transformer. Note: results from FAR: Updated $\mathbf{T}_u$ tend to be less accurate than FAR: One Round $\mathbf{T}_1$. This is expected, as FAR: Updated $\mathbf{T}_u$ is intended to be used in combination with Transformer output $\mathbf{T}_t$ to form final output. In other words, our goal of FAR: Updated $\mathbf{T}_u$ is to improve upon Solver $\mathbf{T}_s$, resulting in better final output after combining with $\mathbf{T}_t$.

FAR: Full $\mathbf{T}$. Final predicted 6DoF pose consisting of the weighted linear combination of $\mathbf{T}_t$ and $\mathbf{T}_u$, weighted by w.

For LoFTR Feature Extractor and Correspondence Estimator, we use $H = 480, W = 640$ and $D = 256, h = 60, w = 80$, except on Map-free Relocalization experiments, where images are size $H = 720, W = 544$, so using the same downsampling, $h = 90, w = 68$. For SuperGlue Correspondence Estimator, we use $H = 480, W = 640$. For 8-Point ViT Feature Extractor (InteriorNet and StreetLearn experiments), we use $H = 224, W = 224$ and $D = 192, h = 24, w = 24$. For 6D Reg Feature Extractor (Map-free Relocalization experiments), we use $H = 360, W = 270$ and $D = 256, h = 12, w = 9$. Map-free Relocalization setup differs slightly from other setups to use 6D Reg features as input rather than LoFTR or 8-Point ViT, and 6D Reg produces a single feature vector for a pair of images rather than two; for details see Section 3.4. In Table 3, we break down the architecture of Transformer $\mathbf{T}_t$. 8-Point ViT output has $d = D/n_h + p_e = 70$, where $n_h = 3$ is the number of heads, and $p_e = 6$ is the size of positional encodings.

Table 1. **Model Architecture: FAR**. High-level first defined, followed by detailed components. N varies depending on the number of correspondences. For LoFTR Feature Extractor and Correspondence Estimator, we use $H = 480, W = 640, D = 256, h = 60, w = 80$. Variables for alternative experiments described in text.

Operation	Overview Inputs	Outputs	Output Shape
Input Image	-	-	$2 \times 3 \times H \times W$
Feature Extractor	Input Image	f_i, f_j	$2 \times D \times h \times w$
Correspondence Estimator	Input Image	$\mathbb{M}$	$N \times 4$
8-Point ViT $\mathbf{T}_t$	f_i, f_j	$\mathbf{T}_t$, w	9, 2
Solver $\mathbf{T}_s$	$\mathbb{M}$	$\mathbf{T}_s$	9
One Round $\mathbf{T}_1$	$\mathbf{T}_s$, $\mathbf{T}_t$, w	$\mathbf{T}_1$	9
Updated $\mathbf{T}_u$	$\mathbb{M}$, $\mathbf{T}_1$	$\mathbf{T}_u$	9
Full $\mathbf{T}$	$\mathbf{T}_u$, $\mathbf{T}_t$, w	$\mathbf{T}$	9

Table 2. **Model Architecture: FAR (Vanilla TF)**.

Operation	Overview Inputs	Outputs	Output Shape
Input Image	-	-	$2 \times 3 \times H \times W$
Correspondence Estimator	Input Image	$\mathbb{M}$	$N \times 4$
Vanilla Transformer $\mathbf{T}_t$	$\mathbb{M}$	$\mathbf{T}_t$, w	9, 2
Solver $\mathbf{T}_s$	$\mathbb{M}$	$\mathbf{T}_s$	9
One Round $\mathbf{T}_1$	$\mathbf{T}_s$, $\mathbf{T}_t$, w	$\mathbf{T}_1$	9
Updated $\mathbf{T}_u$	$\mathbb{M}$, $\mathbf{T}_1$	$\mathbf{T}_u$	9
Full $\mathbf{T}$	$\mathbf{T}_u$, $\mathbf{T}_t$, w	$\mathbf{T}$	9

Table 3. **Model Architecture: Transformer $\mathbf{T}_t$ (8-Point ViT)**.

Operation	Overview Inputs	Outputs	Output Shape
Input Features	-	f_i, f_j	$2 \times D \times h \times w$
LoFTR [7] Self-Attn. Block	f_i, f_j	$f_{i,1}, f_{j,1}$	$2 \times D \times h \times w$
LoFTR Cross-Attn. Block	$f_{i,1}, f_{j,1}$	$f_{i,2}, f_{j,2}$	$2 \times D \times h \times w$
8-Point ViT [5] Cross-Attn. Block	$f_{i,2}, f_{j,2}$	f_o	$2 \times D \times d$
Regression MLP	f_o	$\mathbf{T}_t$	9
Gating MLP	f_o	w	2

Table 4. **Model Architecture: Transformer $\mathbf{T}_t$ (Vanilla TF)** Correspondences optionally include descriptors. If do not, skip Linear Layer, use only Positional Encoding as input to Vanilla Transformer.

Operation	Overview Inputs	Outputs	Output Shape
Input Corr.	-	$\mathbb{M}$	$N \times 2 \times 2$
Input Descriptor	-	$\mathbb{M}_d$	$N \times 2 \times 256$
Positional Encoding	$\mathbb{M}$	f_{pos}	$N \times 384$
Linear Layer	$\mathbb{M}_d$	f_{in}	$N \times 128$
Vanilla Transformer	f_{pos}, f_{in}	f_{tmp}	$N \times 512$
Global Avg. Pooling	f_{tmp}	f_o	512
Regression MLP	f_o	$\mathbf{T}_t$	9
Gating MLP	f_o	w	2

Table 5. **Model Architecture: Regression MLP**.

Overview Operation	Inputs	Outputs	Output Shape
Input Features	-	f_o	$shape(f_o)$
Linear	f_o	f_{tmp0}	512
ReLU	f_{tmp0}	f_{tmp1}	512
Linear	f_{tmp1}	f_{tmp2}	512
Linear	f_{tmp2}	f_{tmp3}	512
ReLU	f_{tmp3}	f_{tmp4}	512
Linear	f_{tmp4}	$\mathbf{T}_t$	9

Table 6. **Model Architecture: Gating MLP** Shape of f_o is 512 in the case of Vanilla Transformer and D in the case of 8-Point ViT.

Overview Operation	Inputs	Outputs	Output Shape
Input Features	-	f_o	$shape(f_o)$
Input Transformer Predicted Pose $\mathbf{T}_t$	-	$\mathbf{T}_t$	9
Input Solver Predicted Pose $\mathbf{T}_s$	-	$\mathbf{T}_s$	9
Input Number of Solver Inliers	-	n_i	3
Linear	$f_o, \mathbf{T}_t, \mathbf{T}_s, n_i$	f_{tmp0}	512
ReLU	f_{tmp0}	f_{tmp1}	512
Linear	f_{tmp1}	f_{tmp2}	512
ReLU	f_{tmp2}	f_{tmp3}	512
Linear	f_{tmp3}	w_{tmp}	2
Sigmoid	w_{tmp}	w	2

Table 7. **Model Architecture: Scale Prediction Network**.

Overview Operation	Inputs	Outputs	Output Shape
Feature Extractor Input	-	f_i, f_j	$2 \times D \times h \times w$
MaxPool2D	f_i, f_j	$f_{i,a}, f_{j,a}$	$2 \times D \times h/2 \times w/2$
Conv2D	$f_{i,a}, f_{j,a}$	$f_{i,b}, f_{j,b}$	$2 \times D/2 \times h/2 \times w/2$
ReLU	$f_{i,b}, f_{j,b}$	$f_{i,c}, f_{j,c}$	$2 \times D/2 \times h/2 \times w/2$
MaxPool2D	$f_{i,b}, f_{j,b}$	$f_{i,c}, f_{j,c}$	$2 \times D/2 \times h/4 \times w/4$
Conv2D	$f_{i,c}, f_{j,c}$	$f_{i,d}, f_{j,d}$	$2 \times D/4 \times h/4 \times w/4$
ReLU	$f_{i,d}, f_{j,d}$	$f_{i,e}, f_{j,e}$	$2 \times D/4 \times h/4 \times w/4$
Conv2D	$f_{i,e}, f_{j,e}$	$f_{i,f}, f_{j,f}$	$2 \times D/16 \times h/16 \times w/16$
ReLU	$f_{i,f}, f_{j,f}$	$f_{i,g}, f_{j,g}$	$2 \times D/4 \times h/16 \times w/16$
Linear	$f_{i,g}, f_{j,g}$	f_0	512
ReLU	f_0	f_1	512
Linear	f_1	f_2	512
ReLU	f_2	f_3	512
Linear	f_3	s	1

2. Prior-Guided Robust Pose Estimator

In our implementation of prior guided pose estimation we use RANSAC as the solver to search over the hypothesis space and also score our models with inlier scores. We use the five-point algorithm to estimate the Essential Matrix [3]. Choosing the five-point algorithm is beneficial in the case of known intrinsics (available in all datasets we use) as it only requires 5 correspondences to estimate a minimal model. This increases the chance of sampling a better hypothesis $\mathbf{H}$ over multiple RANSAC iterations. The five-point algorithm recovers the essential matrix corresponding to a minimal set (5) and we convert this to a translation and rotation matrix

(up to scale).

Prior Probability. The $\beta(\mathbf{H}|\mathbf{T}_1)$ measures the log probability of the hypothesized model $\mathbf{H}$ under $\mathbf{T}_1$. The $\mathbf{H}$ is the essential matrix and $\mathbf{T}_1$ is the 6D transformation matrix from round of prediction. Since it is difficult to measure the probability of $\mathbf{H}$ under $\mathbf{T}_1$ we design a proxy formulation. We simplify the formulation with by computing the implied transforms $\{\mathbf{T}_{\{\mathbf{H},k\}}\}_{k=1}^{2}$ corresponding to each of two possible solutions for the rotation matrix.

There are multiple possible ways to measure the probability of the transform $\mathbf{T}_{\mathbf{H},k}$ under $\mathbf{T}$, one possible solution is to independently measure the distribution for rotation and translation component. This approach however requires tuning different weights for each of the components. In our case we measure the difference in the two transformation by computing the implied effect of the transformations on a given point set.

Specifically, for a randomly sampled point set $\mathbf{G} \equiv \{\mathbf{g}\}_{l=1}^{L}$ in $\mathbb{R}^3$ such that $\mathbf{g}_l \in (-3,-3)^3$. We transform these points with $\mathbf{T}_{\{\mathbf{H},k\}}$ and $\mathbf{T}_1$. We then compute the squared residuals, r_i^N, as distance between the transformed point sets. Assuming the distribution of residuals to be proxy for the pose prior, we can now compute the probability of residuals under a standard Gaussian distribution as,

$$\beta'(\mathbf{T}_{\mathbf{H},k}, \mathbf{T}) = \log(\prod_{l=1}^{L} \exp(-r_l^2)/Z), \tag{1}$$

Z is the normalization constant for the probability distribution. We have two hypothesis corresponding to each $\mathbf{H}$ so we choose solution with the highest log likelihood that best fits with prior to recover $\beta(\mathbf{H}, \mathbf{T})$ as,

$$\beta(\mathbf{H}, \mathbf{T}) = \max\Big(\beta'(\mathbf{T}_{\{\mathbf{H},1\}}, \mathbf{T}), \beta'(\mathbf{T}_{\{\mathbf{H},2\}}, \mathbf{T})\Big) \tag{2}$$

Scoring Function. Using the prior score above now we can combine this with our existing RANSAC inliers scoring function by combining the log likelihood for the hypothesis $\mathbf{H}$ under $\mathbf{T}$ and the likelihood of correspondence set M under the hypothesis $\mathbf{H}$ as,

$$\text{score}(\mathbf{H}) = \alpha\beta(\mathbf{H}, \mathbf{T}) + \sum_{(\mathbf{p},\mathbf{q})\in\mathbb{M}} \mathbb{1}\Big(\mathbf{E}(\mathbf{p}, \mathbf{q}|\mathbf{H}) < \sigma\Big), \tag{3}$$

here σ denotes the inlier threshold and $\mathbf{E}(\mathbf{p}, \mathbf{q}|\mathbf{H})$ measures the Sampson error for correspondences $\mathbf{p}, \mathbf{q}$ under the essential matrix $\mathbf{H}$

3. Additional Experimental Details

Our typical training procedure is to train the Correspondence Estimator, followed by FAR: Transformer $\mathbf{T}_t$ jointly with the backbone, followed by FAR: One Round $\mathbf{T}_1$, followed by FAR: Full $\mathbf{T}$. At each step, we train until validation mean rotation error plateaus, and reload the existing components for the next round of training. In some cases different steps are not applicable e.g. we build upon learned pose backbone in Map-free Relocalization and cannot train the prior on StreetLearn or InteriorNet. We use OneCycleLR [6] scheduler, except if using 6DReg backbone; here we follow prior work [1] in using a constant learning rate. FAR's Kornia-based solver is slower than OpenCV, so for speed we use OpenCV solver to compute $\mathbf{T}_s$ in our final model. For fair comparison in ablations, we compute $\mathbf{T}_s$ using Kornia.

3.1. Ground Truth Robustness Study

In this experiment, we are given correspondences as input, so we proceed directly to training FAR: Transformer $\mathbf{T}_t$ and remaining steps. Training upon perturbed ground truth correspondences typically plateaus after 90 epochs for FAR: Transformer $\mathbf{T}_t$ and FAR: One Round $\mathbf{T}_1$; we find 10 epochs of additional training is sufficient for FAR: Full $\mathbf{T}$. We report $\mathbf{T}_t$ output after FAR: Transformer $\mathbf{T}_t$ training rather than after training with the other components in Far: Full $\mathbf{T}$. This is because after full training, $\mathbf{T}_t$ is inaccurate standalone, since it is trained to be effective in conjunction with $\mathbf{T}_s$. We report $\mathbf{T}_u$ and $\mathbf{T}$ after full training of $\mathbf{T}$. We noticed a mistake in $\mathbf{T}_u$ computation in the ground truth experiment in the original submission; fixing this improves $\mathbf{T}_u$, particularly in the case of more outliers. This is more consistent with our claims the prior improves Solver robustness; intuitively most to outliers. FAR still matches or outperforms $\mathbf{T}_u$, most significantly in the case of noise. We use ground truth correspondence computed via LoFTR's correspondence algorithm from true pose and depth, which consists of a mutual nearest neighbor check.

3.2. Wide-Baseline Pose Estimation on Matterport3D

On the full dataset, we found LoFTR reached best performance after 30 epochs, FAR: Transformer $\mathbf{T}_t$ reached best performance after 39 epochs, FAR: One Round $\mathbf{T}_1$ reached best performance after 32 epochs, FAR: Full $\mathbf{T}$ plateaued after 14 epochs. If using the Vanilla Transformer, FAR: Transformer $\mathbf{T}_t$ reached best performance after 89 epochs, FAR: One Round $\mathbf{T}_1$ reached best performance after 69 epochs, FAR: Full $\mathbf{T}$ plateaued after 12 epochs. We report $\mathbf{T}_t$ after its training for the reasons detailed in 3.1. In addition, we report $\mathbf{T}_s$ output after Correspondence Estimator training for consistency with the Correspondence + Solver baseline throughout the paper. This has little impact upon results compared to reporting after full training of $\mathbf{T}$.

In paper submission, we reported $\mathbf{T}_u$ using scale from LoFTR + Solver + Scale for like-for-like comparison with

the unbiased solver. For final release, we report $\mathbf{T}_u$ using scale from the final FAR model, as this is more consistent with the method figure. For reference, applying the prior to LoFTR + Solver + Scale is reported below. It provides a large improvement to rotation accuracy while translation accuracy still struggles due to poor predicted scale.

	Translation (m)			Rotation (°)		
Transformer: 8-Point ViT	Med.↓	Avg.↓	≤1m↑	Med.↓	Avg.↓	≤30↑
LoFTR + Solver + Scale $\mathbf{T}_s$	0.85	1.21	56.3	**0.26**	9.66	91.2
LoFTR + Updated Solver + Scale $\mathbf{T}_u$	**0.83**	**1.17**	**57.5**	**0.26**	**6.62**	**93.7**
Transformer: Vanilla TF						
LoFTR + Solver + Scale $\mathbf{T}_s$	0.85	1.21	56.3	**0.26**	9.66	91.2
LoFTR + Updated Solver + Scale $\mathbf{T}_u$	**0.83**	**1.17**	**57.2**	**0.26**	**7.30**	**93.2**

3.3. Approach Flexibility

Flexibility to Features and Correspondences. 8-Point ViT features refers to spatial features after all self-attention layers in the 8-Point ViT backbone, since the cross-attention layer in 8-Point ViT produces only a flattened array of features. Given this input, FAR: Transformer $\mathbf{T}_t$ uses the 8-Point ViT variant. This normally consists of a LoFTR layer followed by 8-Point ViT cross-attention. However, in this special case of 8-Point ViT input, we drop the LoFTR layer to make FAR: Transformer $\mathbf{T}_t$ equivalent to full 8-Point ViT output. This allows for closer comparison to the original 8-Point ViT work, while using a specialized architecture, in which inserting a LoFTR layer would not likely be helpful. FAR: Full $\mathbf{T}$ can then use FAR: Transformer $\mathbf{T}_t$ combined with FAR: Updated, with solver output coming from either LoFTR or SuperGlue.

We follow 8-Point ViT training procedure for the model, training for 120k iterations with batch size 60, or about 225 epochs. We then repeat this procedure for FAR: One Round $\mathbf{T}_1$ given correspondences from LoFTR or SuperGlue. Finally, we train for 20k iterations for FAR: Full $\mathbf{T}$.

Dataset Size. On the 40% sized dataset, we found LoFTR reached best performance after 86 epochs, FAR: Transformer $\mathbf{T}_t$ reached best performance after 94 epochs, FAR: One Round $\mathbf{T}_1$ reached best performance after 43 epochs, FAR: Full $\mathbf{T}$ plateaued after 27 epochs.

3.4. Wide-Baseline Pose Estimation on Additional Datasets

InteriorNet and StreetLearn. We use 8-Point ViT as our feature extractor on InteriorNet and StreetLearn as it is SOTA and correspondence-based methods such as LoFTR cannot be trained on the data as it does not contain depth. We follow the training setup of Section 3.3, training FAR: Transformer $\mathbf{T}_T$ and FAR: One Round $\mathbf{T}_1$ sequentially, reloading at each new phase of training, and following 8-Point ViT training schedule for each phase. We cannot use the prior since it requires translation prediction, which cannot be supervised, since the dataset does not contain translation information. Therefore, FAR: Full $\mathbf{T}$ is the output from FAR: One Round $\mathbf{T}_1$. However, we find results are strong even without prior. On InteriorNet, we use LoFTR pretrained on Matterport3D for correspondences. On StreetLearn, correspondences come from LoFTR pretrained on MegaDepth.

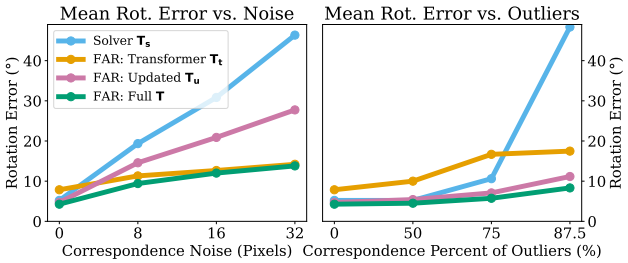

Figure 1. **Ground Truth Robustness Study on Matterport3D: Mean Rotation Error**. Using true correspondence, the solver has low mean error, which is nonzero because of some image pairs having limited ground truth correspondences, leading to small mean error. As with median error, adding noise or outliers causes it to quickly degrade, while prior-guided Updated solver is robust to outliers and Transformer is robust to noise. FAR matches or beats all methods across settings.

Map-free Relocalization. We use 6D Reg as our feature extractor for similar reasons to InteriorNet and StreetLearn: 6D Reg has most competitive rotation and translation errors of existing methods, and correspondence-estimation methods such as LoFTR or SuperGlue cannot be trained on the dataset since it does not contain depth.

6D Reg architecture is different from 8-Point ViT and LoFTR in that it warps features to a common frame before estimating pose. This setup is distinct from the canonical setting of having two dense feature matrices as input, but FAR can nevertheless be adapted. FAR's Transformer $\mathbf{T}_t$ takes features after the penultimate ResNet layer of 6D Reg, which yields feature size of $12 \times 9 \times 256$. The Transformer is a Vanilla Transformer consisting of 6 Transformer Encoder layers with feature size 256 and 8 heads. We choose the penultimate layer as input to the Transformer as these late features are instructional for predicting pose, and are of feasible resolution and feature size for a Transformer. The Vanilla Transformer is lightweight, allowing this to be added to a light 6D Reg architecture without significantly impacting run-time or batch size.

We begin from a 6D Reg backbone pretrained for 50 epochs, train FAR: Transformer $\mathbf{T}_t$ for 20 epochs, train FAR: One Round $\mathbf{T}_1$ for 30 epochs (50 if using SuperGlue correspondences; which runs faster), followed by another 3 for FAR: Full $\mathbf{T}$. Correspondences come from either LoFTR or SuperGlue, both of which are pretrained on outdoor environments. SuperGlue is faster than LoFTR, leading to faster network speed during training and more iterations in the same amount of time. FAR: Full $\mathbf{T}$ training is slower given Kornia solver, so we train for only 3 epochs.

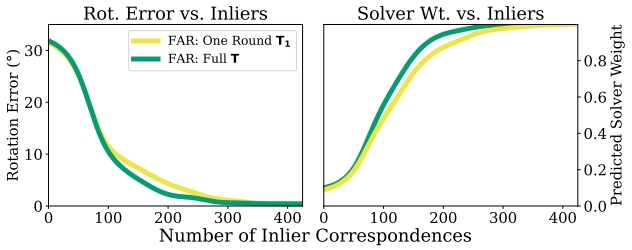

Figure 2. **FAR: Full $\mathbf{T}$ vs. FAR: One Round $\mathbf{T}_1$.** After one round, FAR produces high-quality results, making further improvement difficult. However, a second forward pass through the solver injected with a prior (FAR: Full $\mathbf{T}$) improves solver estimates. Correspondingly, the Transformer learns to give more weight to the solver (right), and there is a nontrivial improvement in rotation error in difficult cases (left, 100-250 inliers).

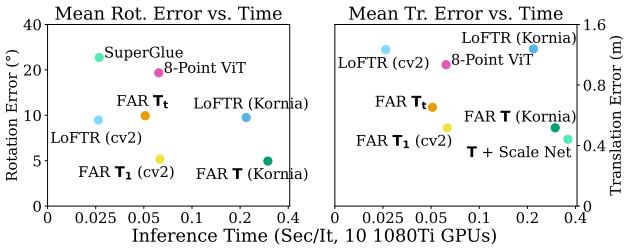

Figure 3. **Efficiency on Matterport (Log Scale vs. Log Scale).** The efficient frontier includes LoFTR+Solver, $\mathbf{T}_1$ and $\mathbf{T}$. FAR $\mathbf{T}_t$ is strictly better than 8-Point ViT. FAR $\mathbf{T}_1$ cuts error almost in half for little time cost. $\mathbf{T}$ improves further, but is slower due to Prior-Guided RANSAC Kornia implementation. Kornia similarly slows down LoFTR+Solver. $\mathbf{T}_1$ with Kornia (not pictured) is also worse than $\mathbf{T}$, while being slower. $\mathbf{T}$+Scale Net further reduces speed and improves Translation error.

We nevertheless find this training beneficial.

4. Additional Results

4.1. Ground-Truth Robustness Experiment

Figure 1 presents mean rotation errors of methods confronted with ground truth correspondences, with varying amounts of noise and outliers. It corresponds to Figure 4 in the paper, except mean rotation error is reported here rather than median rotation error in the paper. The results correspond to those in the paper: the solver is strong faced with little noise or few outliers, but degrades severely. Prior-guided Updated solver is robust to outliers, while Transformer is more robust to noise, at the expense of precision. FAR produces the best of both results in either low perturbation or high perturbation. Note solver median errors are 0, but mean errors are nonzero due to image pairs occasionally having very few ground truth correspondences, producing high errors accordingly. However, this does not impact the experimental conclusion.

4.2. FAR: Full vs. FAR: One Round

Figure 2 displays an analysis of FAR: Full $\mathbf{T}$ vs. FAR: One Round $\mathbf{T}_1$. The distinction between these baselines is highlighted in Figure 3 in the paper, which is that FAR: Full $\mathbf{T}$ adds an additional forward pass to the solver, this time injected with the prior. Like FAR: One Round $\mathbf{T}_1$, this is followed by combination with Transformer predictions.

Despite the two variants of the method being similar, and results of FAR: One Round $\mathbf{T}_1$ being highly competitive, FAR: Full $\mathbf{T}$ yields improvement. This is apparent in the case of 100-250 inliers, where the prior improves solver output, causing the Transformer to rely on it more (Figure 2, right) and the full model to improve (Figure 2, left).

Note FAR: Full $\mathbf{T}$ has different weightings w than FAR: One Round $\mathbf{T}_1$. This is because FAR: Full $\mathbf{T}$ is trained to predict final output given prior-guided solver output. Since prior-guided solver output is more accurate than vanilla solver output, the network learns to rely upon it more heavily. For fair comparison with FAR: Full $\mathbf{T}$, we finetune FAR: One Round $\mathbf{T}_1$ using a Kornia solver for an equal number of epochs used to finetune FAR: Full $\mathbf{T}$; before using the Kornia solver during inference. This is necessary because, as detailed in 3, FAR: Full $\mathbf{T}$ uses cv2's unbiased solver during the first round of computation for efficiency.

4.3. Inference Time Efficiency Comparison

We plot error vs. time in Figure 3. FAR is on the efficient frontier (down and left), though it is slower than LoFTR+Solver using OpenCV (cv2). We note a Prior-Guided RANSAC implementation in cv2 could speed FAR up towards 15fps (e.g. T1).

We also report the performance of using a separate network to predict scale as suggested by R1. To do so, we retrain FAR to predict translation angle only, then train a separate network using the FAR Transformer architecture to predict only scale, such that its final MLP also conditions on predicted translation angle from the first network. It appears only on the right plot, as it impacts Translation and not Rotation error. Indeed, it improves mean Translation error at a small time cost. It produces median Translation error of 0.19m, better than FAR's of 0.25m; and yields 90.9% of predictions within 1m, better than FAR's 89.2%.

4.4. Map-free Precision Analysis

We thank R2 for pointing attention to lack of Map-free precision in the submission. After analyzing, we found an issue with how FAR incorporated Solver predictions on Map-free. Note this issue was specific to Map-free and did not impact experiments on other datasets. After fixing this, the precision of FAR improved to better than LoFTR or SuperGlue on Map-free (paper Table 5), and Solver weight increased to towards 1 on easier cases (Figure 4).

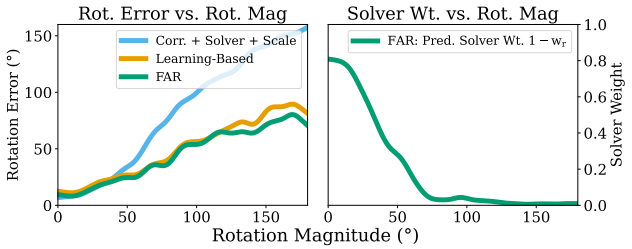

Figure 4. **Solver Weight on Map-free**. Solver Weight on Map-free has similar behavior to Matterport3D, with high weight in easy cases and low weight in difficult cases. The result is high precision and accuracy (paper Table 5).

4.5. Random Results

Results on random examples are presented in Figures 5- 8.

The comparisons are to the same baselines as in Figure 8 in the paper. C+S is an abbreviation for "Correspondence Estimation + Solver", specifically LoFTR with a solver, and learned scale if necessary. L-B is an abbreviation for "Learning-Based", in practice we use LoFTR with an 8-Point ViT head for Matterport3D (specifically, this is equivalent to FAR: Transformer $\mathbf{T}_t$), we use 8-Point ViT for InteriorNet and StreetLearn, and use 6D Reg for Map-free Relocalization. We choose these learning-based and correspondence-based comparisons as they are the state of the art and we build upon them for our method: on all datasets, we use LoFTR to extract correspondence; on Matterport3D, we use LoFTR for features, on InteriorNet and StreetLearn we use 8-Point ViT for features, and on Map-free Relocalization we use 6D Reg for features.

Random results give visual grounding to quantitative results from the paper and are consistent with conclusions that FAR outperforms both C+S and L-B. Only 14 results are presented on each dataset, meaning the sample size is small, and conclusions should not be drawn from aggregate numbers. Rather, these examples are intended to be indicative of performance on a sample-by-sample basis.

For instance, on Matterport3D, FAR is best 10 out of 14 times in rotation and 7 out of 14 times in translation error. In addition, when it is not best, it typically is better than one of C+S or L-B and is typically not significantly worse than the best method. The two qualities that it is often best and rarely worst is in line with significantly better performance than prior work averaged over a full test set.

Random Map-free Relocalization results also agree with conclusions from the paper that FAR is strong. FAR has best rotation and translation 7 out of 14 times; while rival L-B wins 2 times in rotation and 3 times in translation; C-S wins 5 times in rotation and 7 times in translation. Despite strong performance some of the time, recall from the paper C-S error is significantly higher on average than FAR. This is showcased in the random results: when C-S fails, it does so spectacularly, for instance with rotation error of at least 120 degrees on 5 occasions, compared to 0 for FAR. To summarize, in line with quantitative results from the paper, in random samples FAR tends to be significantly more robust than C-S, while producing best results frequently. L-B is also more robust than C-S, but rarely produces best results.

Random results on InteriorNet also are consistent with the paper's findings. FAR has lowest error in 7 of 14 occasions, vs. 5 for L-B and 6 for C+S. However, the highest error for FAR is 4.2°, while L-B hits 4.9° and C-S has 14.5°. On StreetLearn, FAR has a maximum error of 4.4°, while L-B has errors up to 8.2° and C+S has errors of 124° and 177°. FAR has lowest error in 8 instances, vs. 3 for C+S and 5 for L-B. When FAR beats L-B, it is often better by multiple degrees (up to 6, Figure 8, bottom left), while when L-B bests FAR, it is typically by less than three degrees. To summarize, random results elucidate FAR is both precise and robust.

Note results on Map-free Relocalization here, as well as Figure 8 in the paper, are on the validation set, since the test set ground truth is private – test results are available from submitting predictions through the Map-free Relocalization website (https://research.nianticlabs.com/mapfree-reloc-benchmark/submit). Otherwise we use test sets for random results.

4.6. Failure Cases

We can consider some failures in the random examples from Figures 5-8. For instance, some examples in Map-free are beyond the capacity of all the tested models: row 1 column 2 has all models with error above 60°. This is a case of a large rotation around a symmetric and unusually-shaped object, so much so it might be initially challenging to a human. This is a case where recent work focused on visual disambiguation [2] could be of assistance.

Occasionally, FAR also produces the worst results compared to C+S and L-B, for instance in Map-free results row 2 column 4. Ideally, it would perform at least as well as the best of C+S and L-B on any instance, but this is evidence it is not always better than one alternative.

References

[1] Eduardo Arnold, Jamie Wynn, Sara Vicente, Guillermo Garcia-Hernando, Aron Monszpart, Victor Prisacariu, Daniyar Turmukhambetov, and Eric Brachmann. Map-free visual relocalization: Metric pose relative to a single image. In *ECCV*, 2022. 4, 10

[2] Ruojin Cai, Joseph Tung, Qianqian Wang, Hadar Averbuch-Elor, Bharath Hariharan, and Noah Snavely. Doppelgangers: Learning to disambiguate images of similar structures. In *ICCV*, 2023. 7

[3] Richard I Hartley. Estimation of relative camera positions for uncalibrated cameras. In *ECCV*. Springer, 1992. 3

[4] E. Riba, D. Mishkin, D. Ponsa, E. Rublee, and G. Bradski. Kornia: an open source differentiable computer vision library for PyTorch. In *WACV*, 2020. 1

[5] Chris Rockwell, Justin Johnson, and David F Fouhey. The 8-point algorithm as an inductive bias for relative pose prediction by ViTs. In *3DV*, 2022. 2, 9

[6] Leslie N Smith and Nicholay Topin. Super-Convergence: Very fast training of neural networks using large learning rates. In *Artificial intelligence and machine learning for multi-domain operations applications*, 2019. 4

[7] Jiaming Sun, Zehong Shen, Yuang Wang, Hujun Bao, and Xiaowei Zhou. LoFTR: Detector-free local feature matching with transformers. In *CVPR*, 2021. 2, 9

[8] Yi Zhou, Connelly Barnes, Jingwan Lu, Jimei Yang, and Hao Li. On the continuity of rotation representations in neural networks. In *CVPR*, 2019. 1

Random Results: Matterport3D

Rot Mag: 32.5º C+S: 0.3º L-B: 0.3º **FAR**: **0.2º** **Tr Mag**: 2.0m C+S: **0.3m** L-B: 2.8m FAR: 0.5m	**Rot Mag**: 37.0º C+S: 0.3º L-B: 2.8º **FAR**: **0.0º** **Tr Mag**: 2.1m C+S: 0.7m L-B: 0.3m **FAR**: **0.2m**	**Rot Mag**: 41.3º C+S: 0.3º L-B: 0.3º **FAR**: **0.1º** **Tr Mag**: 0.8m C+S: 0.6m L-B: **0.1m** FAR: 0.2m	**Rot Mag**: 12.8º C+S: **0.1º** L-B: 2.3º FAR: 1.3º **Tr Mag**: 2.6m C+S: 1.1m L-B: 0.7m **FAR**: **0.6m**	**Rot Mag**: 78.6º C+S: 0.5º L-B: 2.2º **FAR**: **0.3º** **Tr Mag**: 1.9m C+S: 0.4m L-B: **0.2m** FAR: 0.3m	**Rot Mag**: 82.7º C+S: 73.4º L-B: **2.6º** FAR: 3.2º **Tr Mag**: 0.4m C+S: 1.0m L-B: **0.3m** **FAR**: **0.3m**	**Rot Mag**: 79.3º C+S: 0.6º L-B: 8.2º **FAR**: **0.1º** **Tr Mag**: 3.8m C+S: 2.5m L-B: 1.0m **FAR**: **0.8m**
Rot Mag: 68.9º C+S: **0.3º** L-B: 8.9º FAR: 0.7º **Tr Mag**: 0.5m C+S: 0.9m L-B: 0.1m **FAR**: **0.0m**	**Rot Mag**: 76.7º C+S: 0.5º L-B: 2.9º **FAR**: **0.2º** **Tr Mag**: 2.1m C+S: 0.5m L-B: **0.1m** FAR: 0.2m	**Rot Mag**: 99.2º C+S: 17.0º L-B: 5.3º **FAR**: **1.5º** **Tr Mag**: 1.9m C+S: **0.2m** L-B: 0.3m FAR: 0.3m	**Rot Mag**: 48.6º C+S: **0.1º** L-B: 12.2º **FAR**: **0.1º** **Tr Mag**: 1.5m C+S: **0.1m** L-B: 0.9m FAR: 0.3m	**Rot Mag**: 31.2º C+S: 0.5º L-B: 4.2º **FAR**: **0.1º** **Tr Mag**: 1.1m C+S: 0.6m L-B: 0.4m **FAR**: **0.2m**	**Rot Mag**: 1.5º C+S: **0.1º** L-B: 8.0º FAR: 0.4º **Tr Mag**: 2.4m C+S: 1.7m L-B: **0.4m** FAR: 0.6m	**Rot Mag**: 4.9º C+S: 1.8º L-B: 7.2º **FAR**: **1.0º** **Tr Mag**: 2.3m C+S: 1.8m L-B: **0.3m** **FAR**: **0.3m**

Figure 5. **Random results on Matterport3D**. C+S: LoFTR [7] + Solver. L-B: LoFTR + 8-Point ViT [5]. FAR: uses LoFTR features and correspondences. FAR is typically best. When not best, it is usually better than one of C+S or L-B.

Figure 6. **Random results on Map-free Relocalization**. C+S: LoFTR + Solver. L-B: 6D Reg [1]. FAR: uses 6D Reg features and LoFTR correspondences. FAR is often best, having minimum rotation error 7 instances vs. 5 for C+S and 2 for L-B, and minimum translation error 7 times vs. 7 for C+S and 3 for L-B. C-S has far worse errors in failure cases than FAR (e.g. row 1 column 7).

Figure 7. **Random results on InteriorNet**. C+S: LoFTR + Solver. L-B: 8-Point ViT. FAR: uses 8-Point ViT features and LoFTR correspondences. FAR has the lowest error more frequently than alternatives, and has the lowest maximum error: 4.2° vs. 4.9° for L-B and 14.5° for C-S.

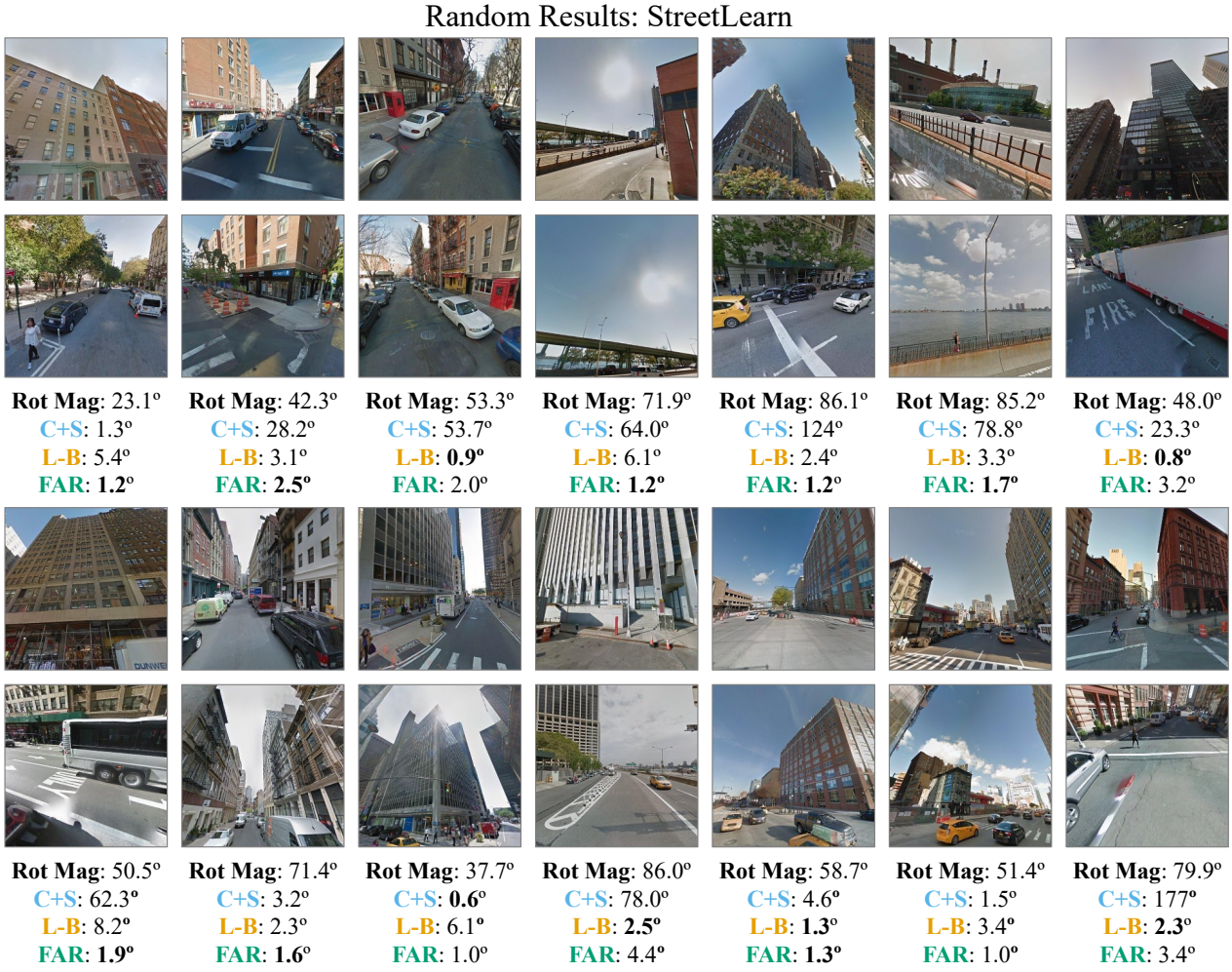

Figure 8. **Random results on StreetLearn**. C+S: LoFTR + Solver. L-B: 8-Point ViT. FAR: uses 8-Point ViT features and LoFTR correspondences. FAR often has the lowest error – here 8 times vs. 1 for C+S and 5 for L-B; and is more robust than alternatives: FAR has maximum error of 4.4°, L-B has maximum error of 8.2°, C+S has errors of 124° and 177°. When FAR beats L-B, it is often better by multiple degrees (up to 6), while when L-B bests FAR, it is typically by less than three degrees.