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

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]

Abstract

Estimating relative camera poses between images has been a central problem in computer vision. Methods that find correspondences and solve for the fundamental matrix offer high precision in most cases. Conversely, methods predicting pose directly using neural networks are more robust to limited overlap and can infer absolute translation scale, but at the expense of reduced precision. We show how to combine the best of both methods; our approach yields results that are both precise and robust, while also accurately inferring translation scales. At the heart of our model lies a Transformer that (1) learns to balance between solved and learned pose estimations, and (2) provides a prior to guide a solver. A comprehensive analysis supports our design choices and demonstrates that our method adapts flexibly to various feature extractors and correspondence estimators, showing state-of-the-art performance in 6DoF pose estimation on Matterport3D, InteriorNet, StreetLearn, and Map-free Relocalization. Project page: https://crockwell.github.io/far/

1. Introduction

Relative camera pose estimation is a fundamental problem in computer vision [24], with applications in augmented reality [29, 37, 42], robotics [50, 67, 81], and autonomous driving [9, 23]. One recent line of work learns to estimate correspondences then solve for pose [19, 43, 51, 65, 70], often offering sub-degree errors. Unfortunately, this framework tends to struggle when faced with large view change (Figure 1, left), and additionally cannot recover scale because it produces the Fundamental or Essential matrix. Another line of work learns to estimate pose directly [5, 10, 41, 62, 75], which is not as precise, but can be more robust and produces translation scale (Figure 1, left and right).

The proposed method builds upon both communities to produce a general method that is no worse than either of the options and often better than both. Critically, it leverages learned correspondence predictions as input, and combines learned pose estimation with a solver to estimate 6DoF pose. For this task, we purposefully select the Transformer, which can handle dense features or correspondences as input. Put succinctly, the method is **F**lexible: agnostic to correspondence and feature backbone; **A**ccurate: matches the precision of correspondence-based methods; and **R**obust: builds upon the resilience of learned pose methods.

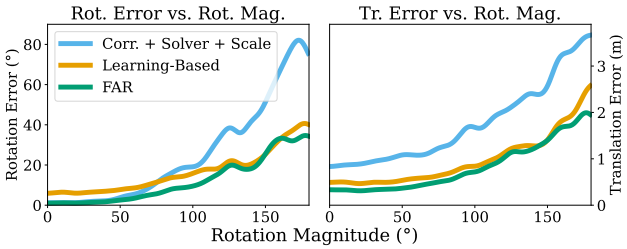

Figure 1. **Precise and Robust 6DoF Pose Estimation**. Correspondence Estimation + Solver methods (here LoFTR [70], RANSAC [22]) produce precise outputs for moderate rotations, but are not robust to large rotations (left), and cannot produce translation scale. Learning-based methods (here LoFTR with 8-Point ViT [62] head) produce scale (right) and are more robust, but lack precision (left). FAR leverages both for precise and robust prediction, including scale.

FAR enables learning-based and solver-based methods to improve each other. Learned predictions are more robust than solver output, and are therefore used as a prior to bias the solver. Improved solver output, which tends to be more precise than learned output when it succeeds, is then combined with Transformer predictions to form final output. Predictions are combined via a weighting predicted by the Transformer, meaning the Transformer can learn to rely more upon either method depending on their effectiveness.

Figure 2 analyzes FAR in practice, measuring error as a function of the number of good input correspondences. With many correspondences, the solver is highly accurate, leaving little room for improvement from the prior. As the number of correspondences drop, solver performance degrades, but this can be alleviated meaningfully using the learned prior. The learned weighting also contributes to robustness, and is plotted on the right: the Transformer primarily uses solver output if there are many correspondences, and more heavily uses the regressor when there

are few correspondences (Figure 2, right). The result is a method that does not sacrifice in the case of many correspondences, but has a large gain given few correspondences.

Experiments analyze FAR in detail across a number of scenarios and datasets. First, we analyze theoretical robustness, beginning from ground truth correspondences and procedurally adding (1) noise and (2) outliers. We next evaluate the proposed method on four challenging datasets, spanning both indoors: Matterport3D and InteriorNet, and outdoors: StreetLearn and Map-free Relocalization. Across settings, the proposed method typically outperforms, or occasionally matches, the state of the art. We additionally analyze the components of FAR in ablations, and apply it to a variety of permutations of correspondence and feature estimation backbones. We also study the impact of dataset size upon model behavior.

2. Related Work

Learned Camera Pose Estimation. Learned Camera Pose Estimation has recently made impressive progress. If many views are available, camera pose can be precisely refined during SLAM [15, 72] or Visual Odometry [36, 73, 76]. If fewer views of a scene are available, methods have become increasingly robust to e.g. large rotation [29, 68, 82].

This paper focuses on the wide-baseline two-view setting, which has also strongly progresses [10, 14, 20, 79, 80]. Some of these methods also perform 3D reconstruction [1, 30, 55, 71]. We build off the 8-Point ViT [62], a SOTA method for two-image 6DoF camera pose estimation.

Correspondence Estimation. Correspondence can be learned [13, 18, 19, 28, 35, 51] or attained using classical methods [6, 45, 64], including specialized for wide-baseline stereo [47, 49, 54]. We use recent SOTA methods LoFTR [70] and SuperPoint+SuperGlue [16, 65], but note FAR can readily adapt to alternative estimators.

Camera Pose Estimation from Correspondences. Camera pose estimation from correspondences is a long standing [22] and still active problem [3]. Typically, algorithms use a robust estimator [3, 4, 22] along with the 7-Point [38] or 8-Point [26] algorithm to find the fundamental matrix, if intrinsics are unknown, or 5-Point algorithm [52] to find the essential matrix, if intrinsics are known. F or E can than be decomposed into RT (without translation scale) and estimating direction via triangulation and the chirality check. We assume known intrinsics and use RANSAC with the 5-Point algorithm, but our contributions are orthogonal to estimator.

Recent work incorporates learnable elements into pose estimation from correspondence [57]. Barroso *et al.* [5] learn to select from candidate essential matrices. Roessle and Nießner [63] use a differentiable 8-Point algorithm, while Wei *et al.* [77, 83] use a differentiable robust estimator, to improve F via end-to-end training. DSAC [7] supervises a Scoring CNN to predict consensus and guide RANSAC. FAR is distinct from these works as its final output is a weighted combination of Solver Pose and Learned Pose. This important difference allows FAR to predict scale and improve robustness to poor or limited correspondences (Figure 2). GRelPose [33] predicts pose from correspondences, but does not use a solver, limiting precision.

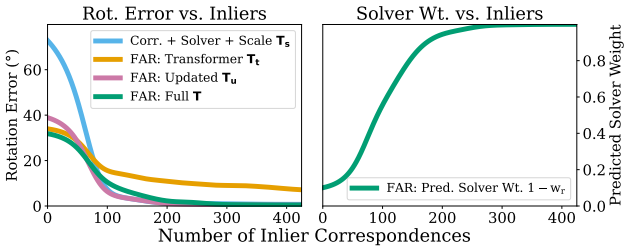

Figure 2. **Combining Classical and Learned**. Left: Solver output is precise given many inliers, but is poor when few are available; Updated solver output via FAR's prior improves robustness significantly. FAR's Transformer is less precise but more robust. The full model fuses prior-guided Solver output and Transformer output for the best of both, giving more weight to the solver when many inliers are available (right).

3. Approach

Our goal is to predict relative camera pose, including translation scale, from two overlapping images. This 6DoF pose can be parameterized as $\mathbf{T} \in \mathrm{SE}(3)$, consisting of $\mathbf{R} \in \mathrm{SO}(3)$ and $\mathbf{t} \in \mathbb{R}^3$. We specifically focus on predicting translation scale, which cannot be solved for from correspondences alone, in order to enable real world applications e.g. 3D reconstruction and neural rendering. The two-view case facilitates these applications on e.g. image collections. We assume known camera intrinsics as they are generally available from modern devices [2].

FAR fuses complimentary strengths from the two lines of pose estimation work: learned correspondence estimation followed by a robust solver, and end-to-end pose estimation. Critically, it produces results that are no worse than either and often better than both. We design the method as flexible to be plugged-in to existing methods with minimal change, showing improved results in a variety of settings and datasets. We outline FAR in Section 3.1, detail the learned network in Section 3.2 and how we apply a prior to the solver in Section 3.3.

3.1. Approach Outline

Figure 3 shows an overview of the proposed approach. At the heart of the approach is the Pose Transformer (Sec 3.2) which takes in dense features and outputs (1) an estimate of 6DoF pose $\mathbf{T}_t$ and (2) a relative weight $\mathbf{w}$ of this prediction. $\mathbf{T}_t$ is then combined with solver-estimated pose $\mathbf{T}_s$ using weight $\mathbf{w}$ to obtain pose estimate $\mathbf{T}_1$. $\mathbf{T}_1$ is used as a prior

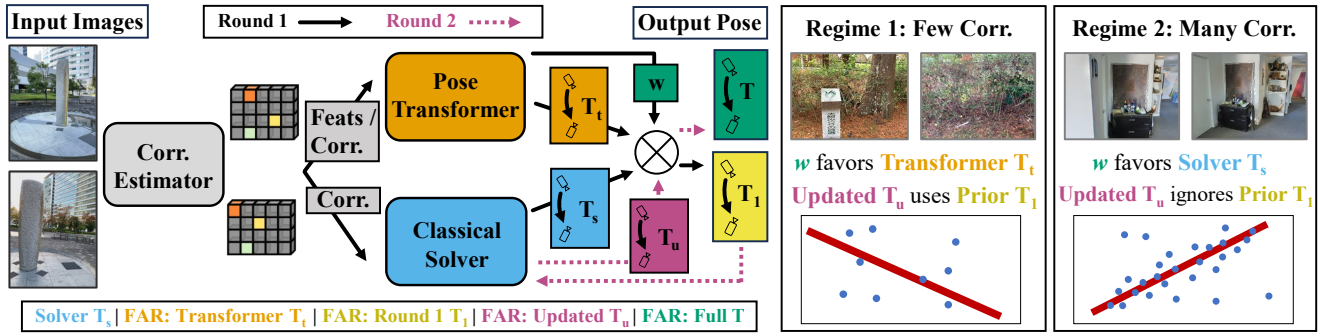

Figure 3. **Overview**. Given dense features and correspondences, FAR's Transformer produces camera poses (in square boxes) through a transformer (round box) and classical solver (round box). In the first round, the solver produces a pose $\mathbf{T}_s$. FAR's pose transformer averages this with its own prediction $\mathbf{T}_t$ via weight $\mathbf{w}$, to yield the round 1 pose $\mathbf{T}_1$. $\mathbf{T}_1$ pose serves as a prior for the classic solver, which produces an updated pose $\mathbf{T}_u$. This is combined with an additional estimate of $\mathbf{T}_t$ and weight $\mathbf{w}$ to produce the final result $\mathbf{T}$. With few correspondences, $\mathbf{T}_1$ helps solver output, while the network learns to weigh Transformer predictions more heavily; with many correspondences, solver output is often good, so the network relies mostly on solver output.

for the solver, resulting in updated solver output $\mathbf{T}_u$, which is combined with $\mathbf{T}_t$ via $\mathbf{w}$ to get the final output $\mathbf{T}$.

This architecture enables the network to learn to behave differently depending on the data regime. In the case of many, high-quality correspondences, classical solvers are typically precise, so the prior has little impact on the solver, while the network learns to heavily rely upon solver output via a low $\mathbf{w}$. In the case of few, low-quality correspondences, solvers degrade, so the prior is designed to have a strong influence on solver output, while the network relies more heavily on the transformer predictions (high $\mathbf{w}$).

The approach is agnostic to input features and correspondences. In experiments (Sec 4), we show success with features from three feature estimation methods [2, 62, 70] and correspondences from two correspondence estimators [65, 70]. We use a ViT [17] to handle spatial features, and losses can backpropagate through the backbone. We also explore using only correspondences and descriptors as input.

3.2. Pose Transformer

The goal of the Transformer is to estimate (1) 6DoF relative camera pose $\mathbf{T}_t$ between two wide-baseline images and (2) weight $\mathbf{w} \in [0, 1]$ of its estimate vs. solver estimates from a set of 2D correspondence matches $\mathbb{M} = \{(\mathbf{p}, \mathbf{q})\} \mid \mathbf{p}, \mathbf{q} \in \mathbb{R}^2$ and optionally dense 2D image-wise features f_i, f_j. Given predicted camera pose and weight, the final output is the linear combination of Transformer pose $\mathbf{T}_t$ and solver pose $\mathbf{T}_s$ weighted by $\mathbf{w}$. We use separate weights for translation w_t and rotation w_r, allowing the Transformer to have different confidences for two subtly different problems.

Two challenges to this setup are that linear combinations of rotations are often not rotation matrices, and solver translation does not have scale. To address the former, we represent pose in the 6D coordinate system of Zhou *et al.* [84], which enables us to combine in 6D space before computing a rotation matrix using Gram–Schmidt orthogonalization. To address scale-less solver output, we scale translation $\mathbf{t}_s$ by the Transformer predicted translation magnitude $||\mathbf{t}_t||$, before linearly combining. We find this stabilizes training compared to first averaging predicted the angles of $\mathbf{t}_s$ and $\mathbf{t}_{tf}$ and then applying scale to normalized predictions. Our final formula is:

$$\begin{aligned} \hat{\mathbf{R}} &= w_r \mathbf{R}_t + (1 - w_r)\mathbf{R}_s \\ \hat{\mathbf{t}} &= w_t \mathbf{t}_t + (1 - w_t)||\mathbf{t}_t||\mathbf{t}_s \end{aligned} \tag{1}$$

Transformer Backbone. We use two distinct architectures to span possible inputs: if features are available, we use a modified ViT. If only correspondences are available, we use a Vanilla Transformer. This means the method can be used with correspondence or regression-based methods producing dense features, while accommodating methods only outputting correspondence. In each case, the Transformer produces features f_o used as input to two MLP heads.

8-Point ViT. This network takes as input pairwise dense features f_i, f_j and produces a feature vector f_o. It consists first of one LoFTR [70] self-attention and cross-attention layer followed by an 8-Point ViT cross-attention layer [17, 46, 62]; both networks are aimed at producing good features for pose estimation. For detailed architectures of each, see the original works.

Vanilla Transformer. This network takes as input a set of correspondences $\mathbb{M} = \{(\mathbf{p}, \mathbf{q})\} \mid \mathbf{p}, \mathbf{q} \in \mathbb{R}^2$ including associated descriptors, if available, and produces a set of features f_o. We use a vanilla Transformer encoder with N layers, and map correspondences and descriptors as input. We encode correspondences in a sinusoidal manner with K bands, followed by a linear mapping to a size of c input to the Transformer. If descriptive features of each correspondence point are available, of dimension $d < c$, a Linear layer maps them to $\frac{c}{4}$ and they are concatenated to correspondence locations which are linearly mapped to $\frac{3c}{4}$.

The Vanilla Transformer can also be used with dense features f as input. If networks produce a joint feature encoding for two images, the Transformer can be applied directly to low-resolution features, without positional encoding. This occurs in the Regression model of Arnold *et al.* [2], which we build upon in Table 5.

Regression MLP. This MLP maps Transformer features f_0 to $\mathbf{R} \in \mathbb{R}^6$ and $\mathbf{t} \in \mathbb{R}^3$ using two hidden layers.

Gating MLP. This takes as input Transformer features concatenated with Regression MLP predictions and predictions from the classical solver, along with the number of inlier correspondences in the solver output, using several thresholds. Predictions and number of inliers are normalized then input as scalar features. As we analyzed in Figure 2, the number of inlier correspondences is highly correlated with the performance of solver pose estimate $\mathbf{T}_s$. The Gating MLP has two hidden layers and ends with a Sigmoid, producing $w_t, w_r \in (0, 1)$.

3.3. Prior-Guided Robust Pose Estimator

Now, having shown how the solver can help learning, how can the learning based methods help the solver? The performance of search-based solver methods like RANSAC [22] is driven by *searching* over a model space by sampling valid hypothesis and then ranking them based on a *scoring function*. The scoring function serves as the measure of probability of data under a hypothesis [74]. It's typical to use such solvers when estimating pose from a set of correspondences, but direct application of these methods can not be robust when correspondence estimation is done with a scarce set. Our key idea is to use the predicted pose estimate, $\mathbf{T}_1$, to influence both the *search* and the *scoring* function to help in data scarce scenarios.

We take inspiration from existing lines of work in using learning to better inform sampling and selection in RANSAC-like algorithms [3, 4, 7, 56, 74]. We show that we can recycle estimates from a learning-based model and plug these estimates in simplistically. Specifically, an initial estimated pose, $\mathbf{T}_1$, to modify the search function so as to sample more hypothesis close to $\mathbf{T}_1$. Secondly, we modify the scoring function to consider the distance to the $\mathbf{T}_1$ along with inlier count.

RANSAC Preliminaries. The typical approach to pose estimation from correspondences applies random sample consensus (and variants) e.g. RANSAC, USAC [56] or MAGSAC [3, 4] to model fitting e.g. 5, 7, or 8-Point algorithms [26, 39, 44, 52]. These methods use a notion of epipolar distance such as Sampson Error [24] for inlier thresholding (soft and hard). More concretely, given a set of 2D correspondence matches $\mathbb{M} = \{(\mathbf{p}, \mathbf{q})\} \mid \mathbf{p}, \mathbf{q} \in \mathbb{R}^2$, a minimal subset of points is randomly sampled to fit a model $\mathbf{H}$ via an n-point algorithm. The scoring function counts number of inliers that have Sampson Error less than a fixed threshold σ. Given, hypothesis $\mathbf{H}$ and set $\mathbf{M}$ of correspondences, $\mathbf{E}(\mathbf{p}, \mathbf{q}|\mathbf{H})$, is the Sampson Error between points $\mathbf{p}$ and $\mathbf{q}$ under $\mathbf{H}$. The scoring function is defined as $\text{score}(\mathbf{H}) = \sum_{\{\mathbf{p},\mathbf{q}\}\in\mathbb{M}} \mathbb{1}(\mathbf{E}(\mathbf{p}, \mathbf{q}|\mathbf{H}) < \sigma)$. Sampling repeats up to N times or until stopping heuristics are met for efficiency [56], and the highest scoring model, e.g. the one with the most inliers, is selected. Works like MAGSAC [3], MAGSAC++ [4] have shown that improving scoring functions to show better performance. For simplicity, we continue the exposition with thresholding based function that are popular with classic RANSAC [22].

Limitations in Few-Correspondence Case. The heuristic score of counting inliers typically is not effective especially in the low-correspondence case [5]. When the number of correspondences is only a small multiple of the number of points needed to minimally define a model the algorithm becomes particularly unreliable. Consider the extreme case of doing pose recovery with calibrated cameras from nine points, of which five are inliers. The minimal subset for pose estimation is five points, and so while the true model will have five inliers, so will any other sampled hypothesis. Accordingly, the result will be random hypothesis.

Prior-Guided Estimator. We propose to incorporate a learning based predictions to aid the solver in the case of few or poor correspondences. We operationalize this by incorporating a prior model that estimates the likelihood of hypothesis under the network's prediction using a function $\beta(\cdot|\mathbf{T}_1)$. The $\beta(\mathbf{H}|\mathbf{T}_1)$ measures the log probability of the hypothesized model $\mathbf{H}$ under $\mathbf{T}_1$. We found it difficult to measure probabilities in rotation and translation and weigh them, so as a proxy, we compare how the models transform a fixed set of grid points. In particular, we measure the negative of average squared distance between a fixed set of grid points transformed by $\mathbf{T}_1$ and the same fixed grid transformed by $\mathbf{H}$. See Supplemental for details.

Now, we show how this β function can alter the scoring. The modified scoring function measures the likelihood of the hypothesis $\mathbf{H}$ under $\mathbf{T}_1$ along with measuring the likelihood of data [74] under $\mathbf{H}$. It is defined as,

$$\text{score}(\mathbf{H}) = \alpha\beta(\mathbf{H}|\mathbf{T}_1) + \sum_{(\mathbf{p},\mathbf{q})\in\mathbb{M}} \mathbb{1}\Big(\mathbf{E}(\mathbf{p}, \mathbf{q}|\mathbf{H}) < \sigma\Big) \quad (2)$$

which is the (log) product of probability of the hypothesis given our β prior function and the probability of the the data, $\mathbb{M}$, under $\mathbf{H}$. We weigh the prior with a scalar, $\alpha \in \mathbb{R}$. In this setting, the prior tie-breaks ambiguous cases where two hypotheses have similar numbers of inliers, but has diminishing influence as $|\mathbb{M}|$ gets bigger. As $|\mathbb{M}| \to \infty$, the prior's impact is washed out entirely. This formulation has the desired impact of significant effect when correspondences are few and unbiased hypotheses are poor, and little impact when correspondences are many.

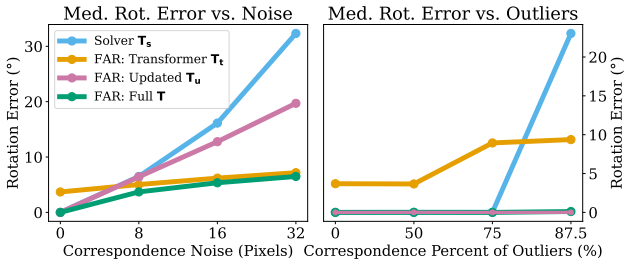

Figure 4. **Ground Truth Robustness Study on Matterport3D**. Using true correspondence, the solver is nearly perfect. Adding noise or outliers, it quickly degrades, while prior-guided Updated solver is robust to outliers and the Transformer is robust to noise. FAR matches or beats all methods across settings.

Sampling Good Hypotheses. Randomly sampling points and estimating $\mathbf{H}$ is unlikely to lead to hypothesis consistent with the model $\mathbf{T}_1$. To increase the chance to sampling consistent hypothesis we want to sample a minimal subset that best agrees with the model. We achieve this by weighing the correspondences by their agreement with the model in turn influencing the as $w(\mathbf{p}, \mathbf{q}) = \exp(-\text{Sampson}(\mathbf{p}, \mathbf{q}|\mathbf{T}_1)/\tau)$.

In practice, we sample half the hypothesis using biased sampling use uniform sampling for the other half. This improves sample diversity in the case of many correspondences, in which case unbiased sampling is very effective.

3.4. Implementation and Training Details

Across experiments, we use the Adam [34] optimizer. The 8-Pt ViT trains for about 300k iterations or 7 days on 10 GTX 1080Ti; Vanilla TF trains for about 600k iterations or 3 days. We select the checkpoint with the lowest validation mean rotation error, which tends to be marginally more stable than translation error. We represent rotation in 6D coordinates [84] and use L1 loss. Models are trained stage-wise: first we train the Transformer to estimate pose, then to estimate pose jointly with a vanilla solver, then to estimate jointly with the prior-based solver. We find this progressive training improves final performance. We implement in PyTorch [53] Lightning [21], using TIMM [78] for the ViT.

8-Point ViT. We train 8-Point ViT end-to-end with the feature extraction backbone. We found including a self and cross-attention LoFTR layer significantly improved learning capacity, while additional layers did not help further.

Vanilla Transformer. We use a 6-layer encoder with 8 heads and 512 feature size followed by global average pooling. We use $K = 42$ bands for positional encoding, and linearly map them to size 384, concatenating descriptive features linearly mapped to size 128. We found random dropout on correspondences with $p = 0.1$ helps performance. We cache correspondences for fast training. Training speed is ≈12 iterations per second on a GTX 1080Ti.

Table 1. **Camera Pose Estimation on Matterport3D.** Correspondence-based methods have low median but high mean error, and do not produce translation scale. Regression-based methods are less precise but produce scale. FAR builds upon both, resulting in low median and mean error, with translation scale.

	Translation (m)			Rotation (°)		
Method	Med.↓	Avg.↓	≤1m↑	Med.↓	Avg.↓	≤ 30↑
[60] + [58]	3.34	4.00	8.3	50.98	57.92	29.9
Assoc.3D [55]	2.17	2.50	14.8	42.09	52.97	38.1
Sparse Planes [30]	0.63	1.25	66.6	7.33	22.78	83.4
PlaneFormers [1]	0.66	1.19	66.8	5.96	22.20	83.8
8-Point ViT [62]	0.64	1.01	67.4	8.01	19.13	85.4
NOPE-SAC-Reg [71]	0.52	0.94	73.2	2.77	14.37	89.0
SuperGlue [65]	-	-	-	3.88	24.17	77.8
LoFTR [70]	-	-	-	0.23	9.49	91.4
LoFTR+Reg. Scale	0.85	1.21	56.3	0.26	9.66	91.2
FAR (Vanilla TF)	0.37	0.67	81.9	0.26	6.14	94.2
FAR	**0.25**	**0.49**	**89.2**	**0.20**	**4.93**	**95.8**

Prior-Guided Estimator. We implement in Kornia [61] and use 2k random samples without early stopping and inlier threshold on L2 Sampson Error σ of 3×10^{-7}, finding these results most closely matched OpenCV [8] output using LoFTR settings. For the Prior, we use $\tau = 0.1$ and $\alpha = 3.33$ found through grid search on the validation set.

4. Experiments

We design our experiments to measure the effectiveness of FAR in achieving our stated goals: flexible, accurate and robust 6DoF pose estimation. We first validate robustness by measuring model performance as a function of increasingly perturbed ground truth. We next test precision to moderate view change and robustness to large view change by comparing to the state of the art in wide-baseline relative pose. Having demonstrated accuracy and robustness, we next verify model flexibility to choice (or lack) of dense feature method, correspondence estimation method, and dataset size. Finally, we compare to the state of the art on additional indoor and outdoor datasets.

4.1. Robustness to Correspondence Perturbations

Here, we assume the image correspondences are given and study how variations of our method and baselines perform with varying noise-levels applied to the correspondences.

Dataset. We use image pairs collected using the Habitat [66] embodied simulator upon Matterport3D [12], following the setup of Jin *et al*. [30]. It has 32k train / 5k val / 8k test pairs with small to moderate overlap (average 53° rotation, 2.3m translation, 21% overlap). The variety in view change enables the study of both precision upon moderate cases and robustness to highly challenging cases.

Metrics. Throughout Matterport3D experiments, we report three metrics for rotation and translation: median error, mean error, and percentage of errors within a threshold. These are standard metrics, which identify our two quali-

Table 2. **Ablations on Matterport3D.** (Top) We improve significantly upon LoFTR using a combination of learned and classical. (Middle) This result holds for the case of no input features, where we use the Vanilla TF. (Bottom) Scaling Solver translation is important to FAR performance; selecting separate weights for R and T improves robustness.

	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
FAR: Transformer $\mathbf{T}_t$	0.38	0.64	85.4	4.51	9.94	94.2
FAR: One Round $\mathbf{T}_1$	**0.25**	**0.49**	89.0	**0.20**	5.08	95.7
FAR: Updated $\mathbf{T}_u$	**0.25**	0.50	88.4	**0.20**	5.35	95.0
FAR: Full $\mathbf{T}$	**0.25**	**0.49**	**89.2**	**0.20**	**4.93**	**95.8**
Transformer: Vanilla TF						
LoFTR + Solver + Scale $\mathbf{T}_s$	0.85	1.21	56.3	0.26	9.66	91.2
FAR: Transformer $\mathbf{T}_t$	0.42	0.75	79.1	3.87	10.8	92.5
FAR: One Round $\mathbf{T}_1$	**0.37**	**0.67**	81.8	0.26	6.41	93.8
FAR: Updated $\mathbf{T}_u$	**0.37**	0.68	81.5	**0.25**	6.69	93.7
FAR: Full $\mathbf{T}$	**0.37**	**0.67**	**81.9**	0.26	**6.14**	**94.2**
Prediction Selection						
Unscaled Solver t_s	0.31	0.55	87.4	0.21	5.09	**95.8**
One Weight ($w_r = w_t$)	**0.25**	0.50	88.7	**0.20**	5.04	**95.8**
FAR	**0.25**	**0.49**	**89.2**	**0.20**	**4.93**	**95.8**

ties of interest: precision (median) and robustness (mean and percentage). We follow prior work [30, 71] in using rotation threshold of 30° and translation threshold of 1m. For the ground truth study, for brevity we report median error across a variety of settings, which is an indicative summary of performance. Additional results are in Supplemental.

Setup. Beginning with ground truth correspondences, we (1) Apply Gaussian noise with standard deviation from 0 to 32 pixels, upon 480x640 images; (2) Replace true correspondences with randomly sampled coordinates (outliers), with P(Outliers) from 0 to 0.875. Models are trained and evaluated upon each noise and outlier setting independently; e.g. FAR is trained and evaluated four times to make Correspondence Noise Graph in Figure 4, left.

Ablations. We consider the following cases:

(1) Solver $\mathbf{T}_s$. Using LoFTR's solver, i.e., RANSAC [22]+5-Point Algorithm [52]

(2) FAR: Transformer $\mathbf{T}_t$. Pose Transformer output pose. This is a Vanilla Transformer, since dense features are not available as input; only correspondences.

(3) FAR: Updated $\mathbf{T}_u$. Solver output using FAR's Prior

(4) FAR: Full $\mathbf{T}$. Full FAR: learned combin. of $\mathbf{T}_u$ and $\mathbf{T}_t$

Ablation Results. Figure 4 shows Solver has nearly perfect results on ground truth correspondences but is not robust to noise or many outliers. FAR: Transformer is less precise on ground truth but is more robust as outlier frequency and particularly noise increases. The prior is highly effective at leaving FAR: Updated robust to outliers, showing close to 0° median error even with 87.5% outliers. The full method offers the best of all: precise estimation on ground truth correspondences with the best or equal to best robustness to noise and outliers.

Table 3. **Approach Flexibility to Features and Correspondences.** FAR yields improvement using features from 8-Pt ViT or LoFTR; and correspondences from SuperGlue or LoFTR.

Feats.	Corr.	Pose Est.	Translation (m)			Rotation (°)		
			Med.↓	Avg.↓	≤1m↑	Med.↓	Avg.↓	≤ 30↑
8-Pt ViT	-	8-Pt ViT	0.64	**1.01**	67.4	8.01	19.1	85.4
8-Pt ViT	SuperGlue	FAR	**0.62**	**1.01**	68.3	**7.02**	**16.6**	86.8
8-Pt ViT	LoFTR	FAR	0.63	**1.01**	**68.5**	7.06	17.0	**86.9**
LoFTR	LoFTR	RANSAC+5Pt	-	-	-	0.23	9.49	91.4
-	LoFTR	FAR (Vanilla TF)	0.37	0.67	81.9	0.26	6.14	94.2
LoFTR	LoFTR	FAR	**0.25**	**0.49**	**89.2**	**0.20**	**4.93**	**95.8**

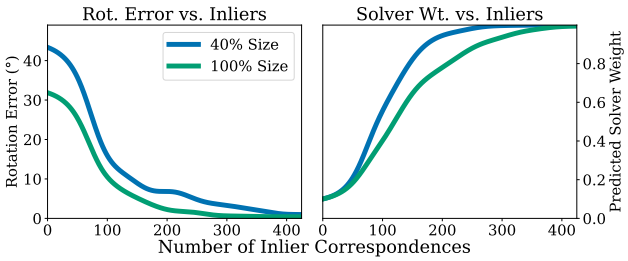

Figure 5. **Evolving with Dataset Size**. The Transformer learns to rely more heavily upon the solver if data is limited (40% data size), and learns to use Transformer pose estimations as data scales and performance improves (100% data size).

4.2. Wide-Baseline Pose on Matterport3D

In this section, we use the same Matterport3D dataset and the metrics used in Sec. 4.1, but the inputs are images rather than the GT correspondences.

Baselines. We compare against state-of-the-art solver-based and learning-based baselines. For solver-based methods, we choose the popular LoFTR [70] and SuperGlue [65]. In the learned space, we compare to end-to-end classical-estimation-inspired ViT, 8-Point [62]; planar mapping and optimization methods NOPE-SAC [71], PlaneFormers [1] and Sparse Planes [30]; and 3D reconstruction method Associative3D [55]. In this set of experiments our FAR builds upon LoFTR backbone and correspondences. We additionally report results using only correspondence and descriptor as input, as "FAR (Vanilla TF)".

Results. Table 1 shows the quantitative results on Matterport3D. Among the prior works, end-to-end methods such as 8-Point ViT [62] perform well in absolute Translation, while correspondence-solver methods e.g. LoFTR [70] perform best in rotation. FAR sets a new standard in both metrics, surpassing the best prior baseline (NOPE-SAC-Reg) by a large margin. It reduces the median and mean translation errors by about 50%: from 0.52 to 0.25 and from 0.94 to 0.49, respectively. Additionally, it decreases the mean rotation error by almost 50% compared to the best prior work (LoFTR), from 9.66 to 4.93. Even with only correspondence available as input, "FAR (Vanilla TF)" is typically better than all prior work by a large margin.

Ablations. To investigate the source of FAR's out-

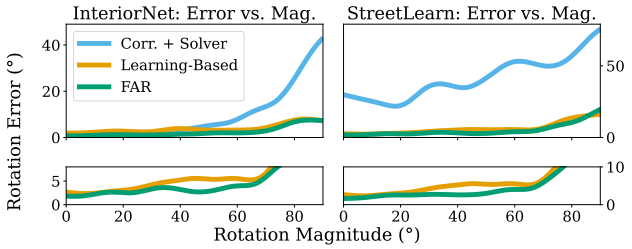

Figure 6. **Rotation Error on InteriorNet and StreetLearn**. Even when Correspondence + Solver method is relatively poor, we can still leverage it to improve regression results. "Learning-Based": 8-Point ViT [62]. "Corr. + Solver": LoFTR [70].

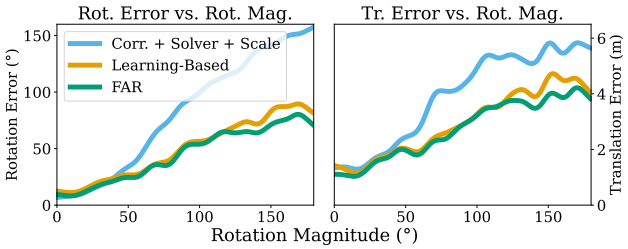

Figure 7. **Error on Map-free Relocalization**. FAR leverages Solver output to improve regression results on this highly challenging dataset. "Corr. + Solver + Scale": LoFTR + DPT [59] trained on KITTI [23]. "Learning-Based": 6D Reg.

performance, we conduct ablations on Matterport3D using the same setup of Sec. 4.1, except correspondences are predicted. In addition to ablations discussed in Section 4.1, we consider model output after One Round ($\mathbf{T}_1$) to study the impact of the prior-guided solver upon the final model. We further explore performance by using two different Transformer architectures: the dense feature-based Transformer: 8-Point ViT (referred to as FAR in other experiments), and the correspondence-only Transformer: Vanilla TF. Finally, we evaluate the design choices outlined in Section 3.2 under the Prediction Selection category: Unscaled Solver t_s, where Solver translation is not scaled by Transformer magnitude; and One Weight ($w_r = w_t$), which uses equal weights for Transformer translation and rotation prediction.

Ablation Results. As shown in Tab. 2, we can clearly observe the same trends from Fig. 4: the solver is precise in most cases characterized by low median rotation error. However the solver suffers from high mean error (due to outliers) and poor translation errors. Incorporating FAR's prior significantly improves the solver's mean rotation error. In contrast, Transformer regression outputs are not nearly as precise, with median rotation error of above 4°, but it reduces the ratios of large errors (those greater than $1m$ or 30°). FAR enhances the best results achieved by both the Transformer and Solver. These patterns hold true for the Vanilla Transformer as well. In *Prediction Selection*, we see predicting Solver translation scale is important for translation performance, while separate weighting for rotation and translation improves robustness.

4.3. Approach Flexibility

We then evaluate the flexibility of FAR in terms of the feature extractor, correspondence estimator, and dataset size.

Dataset and Metrics. We continue using the Matterport3D dataset and metrics as in Section 4.1.

Alternative Approaches. To assess the versatility of FAR, we explore options that are orthogonal to our core contribution. Specifically, we examine three settings for feature estimation: the recent SOTA methods LoFTR and 8-Point ViT, as well as a scenario without dense features. Additionally, we evaluate two recent SOTA settings for correspondence estimation: LoFTR, and SuperPoint [16] + SuperGlue [65].

Results. Table 3 shows FAR improves upon 8-Pt ViT, using either SuperGlue or LoFTR correspondences. Similarly, FAR improves LoFTR, whether it employs both LoFTR features and correspondences or just the correspondences.

Dataset Scaling. We next present the proposed method when trained on a version of the Matterport3D dataset that has been randomly subsampled to 40% of its original data size. In Figure 5, we compare rotation error (left) and solver weight (right) of 40% size and full size. Note that as the training dataset size increases from 40% to 100%, both the solver weight and error decrease. This trend aligns with expectations: as the Transformer's estimated pose accuracy improves with more training data, it gains a larger influence in the final output, enhancing overall performance. The result suggests a fixed weighting of learned and solver output is not sufficient for best results.

4.4. Wide-Baseline Pose on Additional Datasets

We evaluate our method's performance on various datasets to assess its versatility. We follow Cai *et al*. [10] and use InteriorNet [40], a synthetic collection of indoor home scenes, and StreetLearn [48], which features outdoor city street photos. Both datasets consist of 90° field-of-view image crops upon panoramas. Image pairs are chosen from different panoramas with varying overlaps, facilitating the assessment of precision and robustness in scenarios with both large ($> 45°$) and small ($< 45°$) overlaps. Additionally, we use Map-free Relocalization [2], a challenging dataset of user-collected videos surrounding outdoor places of interest e.g. sculptures or fountains. SfM has been applied to the videos, so translation with scale can be evaluated.

Metrics. For InteriorNet and StreetLearn, we report rotation error only, in line with prior work [10, 62], using a 10° threshold. For Map-free Relocalization, we calculate median translation and rotation errors per video, then average these. We also include the Virtual Correspondence Repro-

Table 4. **Rotation Performance on InteriorNet and StreetLearn.** Correspondence-based methods (top) struggle to generalize to this data, while regression methods learn helpful features. Building off 8-Point ViT features, we can still utilize LoFTR correspondences to boost performance. Errors were calculated only on successful pairs for SIFT and SuperPoint; gray text indicates failure over 50% of pairs.

	InteriorNet						StreetLearn					
	Large Overlap (°)			Small Overlap (°)			Large Overlap (°)			Small Overlap (°)		
Method	Med.↓	Avg.↓	≤ 10↑	Med.↓	Avg.↓	≤ 10↑	Med.↓	Avg. ↓	≤ 10↑	Med.↓	Avg.↓	≤ 10↑
SIFT* [45]	2.95	7.78	55.5	10.0	18.2	18.5	3.13	18.9	22.4	13.8	38.8	5.7
SuperPoint* [16]	2.79	5.46	65.9	5.82	11.6	11.7	1.79	6.38	16.5	6.85	6.80	1.0
LoFTR [70]	**0.54**	1.85	97.0	2.64	14.3	70.4	24.8	36.4	31.6	51.2	58.6	19.9
Reg6D [84]	6.91	10.5	67.8	11.4	21.9	44.1	6.02	12.3	69.1	7.59	15.1	63.4
Cai *et al.* [10]	1.10	2.89	97.6	1.38	10.2	89.8	2.91	9.12	87.5	3.49	13.0	84.2
8-Point ViT [62]	1.83	2.90	97.9	2.38	4.48	**96.3**	2.43	4.08	90.1	3.25	9.19	87.7
FAR	0.60	**1.16**	**98.5**	**1.22**	**3.42**	95.4	**1.81**	**3.01**	**96.7**	**2.07**	**7.89**	**92.4**

Table 5. **6DoF Performance on Map-free Relocalization**. We compare against the strongest baselines from [2]; for all comparisons see the original paper. FAR uses the 6D Reg backbone, adding LoFTR or SuperGlue correspondences to beat 6D Reg.

Method	Pose Error				VCRE		
	Avg. Med.↓		Prec.↑	AUC↑	Avg. Med.↓	Prec.↑	AUC↑
LoFTR	199cm	30.6°	0.15	0.35	168px	0.35	0.61
SuperGlue	188cm	25.4°	0.17	0.35	160px	0.36	0.60
Reg Ang. [2]	210cm	33.7°	0.09	-	200px	0.30	-
6D Reg [2]	168cm	22.5°	0.06	-	147px	0.40	-
FAR (SG)	149cm	**17.2°**	0.17	0.35	**135px**	**0.44**	0.67
FAR (LoFTR)	**148cm**	18.1°	**0.18**	**0.39**	137px	**0.44**	**0.68**

jection Error (VCRE) metric to measure reprojection errors (see [2] for details).

Baselines. We compare our method with SOTA correspondence and pose estimation techniques. For InteriorNet and StreetLearn, we compare to Cai *et al.*'s [10] correlation volume-based learning, SuperPoint [16], and the classical SIFT method [45]. We use LoFTR adapted for InteriorNet using Matterport3D, and for StreetLearn using MegaDepth, due to the lack of depth data in these datasets for training correspondences. Since LoFTR cannot be finetuned, we find 8-Point ViT features are more effective, and use these as input to FAR, along with LoFTR correspondences. See Supplemental for details.

Arnold *et al.* [2] train a variety of pose estimation methods on Map-free, including "6D Reg", which creates a correlation volume [31, 32] and warps accordingly, followed by a ResNet [27], and is supervised upon 6D rotation [84].

Results. Tab. 4 and Fig. 6 show that 8-Point ViT [62] achieves impressive mean errors, under 5°, on InteriorNet, even for small overlap pairs. FAR still adds precision on top of the 8-Point ViT. On the challenging StreetLearn data, FAR significantly outperforms the state of the art, despite LoFTR not generalizing well to StreetLearn.

6D Reg is the strongest overall baseline on Map-free Relocalization, so we use its features for FAR, taking correspondences from LoFTR or SuperGlue. In both cases, FAR improves upon 6D Reg and other methods (see Tab. 5 and Fig. 7). These results across datasets show FAR's adaptability to different backbones and its robustness to sub-optimal correspondence estimates, highlighted in Fig. 8.

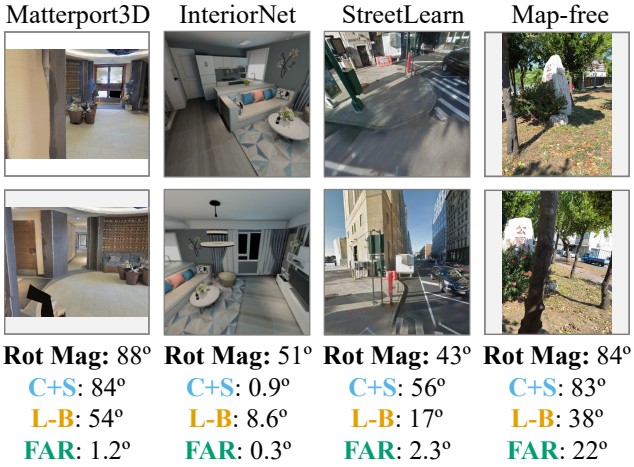

Figure 8. **Success Cases**. For some challenging wide-baseline image pairs, our method often dramatically outperforms the baselines. "Learning-Based": LoFTR [70] with 8-Pt. ViT [62] head (Matterport3D), 8-Pt. ViT (InteriorNet, StreetLearn), 6D Reg [2] (Map-free). "Corr. + Solver": LoFTR.

5. Conclusion

In this work, we address the problem of 6DoF relative camera pose estimation given a wide-baseline image pair. Our introduced FAR represents a simple yet potent approach that merges the best aspects of correspondence-based and learning-based methods. This results in precise and robust outcomes, adaptable to various backbones and solvers.

Limitations and Societal Impact. FAR consists of several components and implements Prior-Guided RANSAC in Kornia, slowing inference speed to 3.3 it/sec on 10 1080Ti GPUs; analysis vs. other methods appears in Supplemental Figure 11. Training upon affluent homes of Matterport3D can result in worse performance on more typical residences.

Acknowledgments and Disclosure of Funding. Thanks to Jeongsoo Park and Sangwoo Mo for their helpful feedback. Thanks to Laura Fink and UM DCO for their tireless computing support. Toyota Research Institute provided funds to support this work.

References

[1] Samir Agarwala, Linyi Jin, Chris Rockwell, and David F. Fouhey. PlaneFormers: From sparse view planes to 3d reconstruction. In *ECCV*, 2022. 2, 5, 6

[2] 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. 2, 3, 4, 7, 8, 15, 20

[3] Daniel Barath, Jiri Matas, and Jana Noskova. MAGSAC: marginalizing sample consensus. In *CVPR*, 2019. 2, 4

[4] Daniel Barath, Jana Noskova, Maksym Ivashechkin, and Jiri Matas. MAGSAC++, a fast, reliable and accurate robust estimator. In *CVPR*, 2020. 2, 4

[5] Axel Barroso-Laguna, Eric Brachmann, Victor Adrian Prisacariu, Gabriel J Brostow, and Daniyar Turmukhambetov. Two-view geometry scoring without correspondences. In *CVPR*, 2023. 1, 2, 4

[6] Herbert Bay, Tinne Tuytelaars, and Luc Van Gool. SURF: Speeded up robust features. In *ECCV*, 2006. 2

[7] Eric Brachmann, Alexander Krull, Sebastian Nowozin, Jamie Shotton, Frank Michel, Stefan Gumhold, and Carsten Rother. DSAC-differentiable RANSAC for camera localization. In *CVPR*, 2017. 2, 4

[8] G. Bradski. The OpenCV Library. *Dr. Dobb's Journal of Software Tools*, 2000. 5

[9] Guillaume Bresson, Zayed Alsayed, Li Yu, and Sébastien Glaser. Simultaneous localization and mapping: A survey of current trends in autonomous driving. *T-IV*, 2017. 1

[10] Ruojin Cai, Bharath Hariharan, Noah Snavely, and Hadar Averbuch-Elor. Extreme rotation estimation using dense correlation volumes. In *CVPR*, 2021. 1, 2, 7, 8

[11] 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. 18

[12] Angel Chang, Angela Dai, Thomas Funkhouser, Maciej Halber, Matthias Niessner, Manolis Savva, Shuran Song, Andy Zeng, and Yinda Zhang. Matterport3D: Learning from RGB-D data in indoor environments. In *3DV*, 2017. 5

[13] Hongkai Chen, Zixin Luo, Lei Zhou, Yurun Tian, Mingmin Zhen, Tian Fang, David Mckinnon, Yanghai Tsin, and Long Quan. ASpanFormer: Detector-free image matching with adaptive span transformer. In *ECCV*, 2022. 2

[14] Kefan Chen, Noah Snavely, and Ameesh Makadia. Wide-baseline relative camera pose estimation with directional learning. In *CVPR*, 2021. 2

[15] Jan Czarnowski, Tristan Laidlow, Ronald Clark, and Andrew J Davison. DeepFactors: Real-time probabilistic dense monocular SLAM. *RA-L*. 2

[16] Daniel DeTone, Tomasz Malisiewicz, and Andrew Rabinovich. SuperPoint: Self-supervised interest point detection and description. In *CVPRW*, 2018. 2, 7, 8

[17] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. *ICLR*, 2021. 3

[18] 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. In *CVPR*, 2019. 2

[19] Johan Edstedt, Ioannis Athanasiadis, Mårten Wadenbäck, and Michael Felsberg. DKM: Dense kernelized feature matching for geometry estimation. In *CVPR*, 2023. 1, 2

[20] Sovann En, Alexis Lechervy, and Frédéric Jurie. RPNet: An end-to-end network for relative camera pose estimation. In *ECCVW*, 2018. 2

[21] William Falcon and The PyTorch Lightning team. PyTorch Lightning, 2019. 5

[22] Martin A Fischler and Robert C Bolles. Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. *Communications of the ACM*, 1981. 1, 2, 4, 6

[23] Andreas Geiger, Philip Lenz, and Raquel Urtasun. Are we ready for autonomous driving? the kitti vision benchmark suite. In *CVPR*, 2012. 1, 7

[24] Richard Hartley and Andrew Zisserman. *Multiple view geometry in computer vision*. Cambridge university press, 2003. 1, 4

[25] Richard I Hartley. Estimation of relative camera positions for uncalibrated cameras. In *ECCV*. Springer, 1992. 12

[26] Richard I Hartley. In defense of the eight-point algorithm. *TPAMPI*, 19(6):580–593, 1997. 2, 4

[27] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *CVPR*, 2016. 8

[28] Dihe Huang, Ying Chen, Yong Liu, Jianlin Liu, Shang Xu, Wenlong Wu, Yikang Ding, Fan Tang, and Chengjie Wang. Adaptive assignment for geometry aware local feature matching. In *CVPR*, 2023. 2

[29] Hanwen Jiang, Zhenyu Jiang, Kristen Grauman, and Yuke Zhu. Few-view object reconstruction with unknown categories and camera poses. *arXiv preprint arXiv:2212.04492*, 2022. 1, 2

[30] Linyi Jin, Shengyi Qian, Andrew Owens, and David F Fouhey. Planar surface reconstruction from sparse views. In *ICCV*, 2021. 2, 5, 6

[31] Alex Kendall, Hayk Martirosyan, Saumitro Dasgupta, Peter Henry, Ryan Kennedy, Abraham Bachrach, and Adam Bry. End-to-end learning of geometry and context for deep stereo regression. In *ICCV*, 2017. 8

[32] Sameh Khamis, Sean Fanello, Christoph Rhemann, Adarsh Kowdle, Julien Valentin, and Shahram Izadi. StereoNet: Guided hierarchical refinement for real-time edge-aware depth prediction. In *ECCV*, 2018. 8

[33] Fadi Khatib, Yuval Margalit, Meirav Galun, and Ronen Basri. GRelPose: Generalizable end-to-end relative camera pose regression. *arXiv preprint arXiv:2211.14950*, 2022. 2

[34] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *ICLR*, 2015. 5

[35] Dominik A Kloepfer, Joao F Henriques, and Dylan Campbell. Scenes: Subpixel correspondence estimation with epipolar supervision. 2024. 2

[36] Lei Lai, Zhongkai Shangguan, Jimuyang Zhang, and Eshed Ohn-Bar. Xvo: Generalized visual odometry via cross-modal self-training. In *ICCV*, 2023. 2

[37] Zihang Lai, Sifei Liu, Alexei A Efros, and Xiaolong Wang. Video autoencoder: self-supervised disentanglement of static 3d structure and motion. In *ICCV*, 2021. 1

[38] Viktor Larsson, Magnus Oskarsson, Kalle Astrom, Alge Wallis, Zuzana Kukelova, and Tomas Pajdla. Beyond grobner bases: Basis selection for minimal solvers. In *CVPR*, 2018. 2

[39] Hongdong Li and Richard Hartley. Five-point motion estimation made easy. In *ICPR*, 2006. 4

[40] Wenbin Li, Sajad Saeedi, John McCormac, Ronald Clark, Dimos Tzoumanikas, Qing Ye, Yuzhong Huang, Rui Tang, and Stefan Leutenegger. InteriorNet: Mega-scale multi-sensor photo-realistic indoor scenes dataset. In *BVMC*, 2018. 7

[41] Amy Lin, Jason Y Zhang, Deva Ramanan, and Shubham Tulsiani. RelPose++: Recovering 6d poses from sparse-view observations. *arXiv preprint arXiv:2305.04926*, 2023. 1

[42] Chen-Hsuan Lin, Wei-Chiu Ma, Antonio Torralba, and Simon Lucey. BARF: Bundle-adjusting neural radiance fields. In *ICCV*, 2021. 1

[43] Philipp Lindenberger, Paul-Edouard Sarlin, and Marc Pollefeys. LightGlue: Local Feature Matching at Light Speed. In *ICCV*, 2023. 1

[44] H Christopher Longuet-Higgins. A computer algorithm for reconstructing a scene from two projections. *Nature*, 293 (5828):133–135, 1981. 4

[45] David G Lowe. Distinctive image features from scale-invariant keypoints. *IJCV*. 2, 8

[46] Jiasen Lu, Dhruv Batra, Devi Parikh, and Stefan Lee. ViLBERT: Pretraining task-agnostic visiolinguistic representations for vision-and-language tasks. *NeurIPS*, 32, 2019. 3

[47] Jiri Matas, Ondrej Chum, Martin Urban, and Tomás Pajdla. Robust wide-baseline stereo from maximally stable extremal regions. *Image and vision computing*, 22(10):761–767, 2004. 2

[48] Piotr Mirowski, Andras Banki-Horvath, Keith Anderson, Denis Teplyashin, Karl Moritz Hermann, Mateusz Malinowski, Matthew Koichi Grimes, Karen Simonyan, Koray Kavukcuoglu, Andrew Zisserman, et al. The StreetLearn environment and dataset. *arXiv preprint arXiv:1903.01292*, 2019. 7

[49] Dmytro Mishkin, Jiri Matas, Michal Perdoch, and Karel Lenc. WxBS: Wide baseline stereo generalizations. *BMVC*, 2015. 2

[50] Raul Mur-Artal, Jose Maria Martinez Montiel, and Juan D Tardos. ORB-SLAM: a versatile and accurate monocular SLAM system. *T-RO*. 1

[51] Junjie Ni, Yijin Li, Zhaoyang Huang, Hongsheng Li, Hujun Bao, Zhaopeng Cui, and Guofeng Zhang. PATS: Patch area transportation with subdivision for local feature matching. In *CVPR*, 2023. 1, 2

[52] David Nistér. An efficient solution to the five-point relative pose problem. *TPAMI*, 2004. 2, 4, 6

[53] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. PyTorch: An imperative style, high-performance deep learning library. *NeurIPS*, 32, 2019. 5

[54] Phil Pritchett and Andrew Zisserman. Wide baseline stereo matching. In *ICCV*, 1998. 2

[55] Shengyi Qian, Linyi Jin, and David F Fouhey. Associative3D: Volumetric reconstruction from sparse views. In *ECCV*, 2020. 2, 5, 6

[56] Rahul Raguram, Ondrej Chum, Marc Pollefeys, Jiri Matas, and Jan-Michael Frahm. USAC: A universal framework for random sample consensus. *IEEE transactions on pattern analysis and machine intelligence*, 2012. 4

[57] René Ranftl and Vladlen Koltun. Deep fundamental matrix estimation. In *ECCV*, 2018. 2

[58] René Ranftl, Katrin Lasinger, David Hafner, Konrad Schindler, and Vladlen Koltun. Towards robust monocular depth estimation: Mixing datasets for zero-shot cross-dataset transfer. *TPAMI*, 2020. 5

[59] René Ranftl, Alexey Bochkovskiy, and Vladlen Koltun. Vision transformers for dense prediction. In *ICCV*, 2021. 7

[60] Carolina Raposo, Miguel Lourenço, Michel Antunes, and Joao Pedro Barreto. Plane-based odometry using an RGB-D camera. In *BMVC*. 5

[61] E. Riba, D. Mishkin, D. Ponsa, E. Rublee, and G. Bradski. Kornia: an open source differentiable computer vision library for PyTorch. In *WACV*, 2020. 5, 12

[62] 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. 1, 2, 3, 5, 6, 7, 8, 13, 19

[63] Barbara Roessle and Matthias Nießner. End2End multi-view feature matching with differentiable pose optimization. In *ICCV*, 2023. 2

[64] Ethan Rublee, Vincent Rabaud, Kurt Konolige, and Gary Bradski. ORB: An efficient alternative to SIFT or SURF. In *ICCV*, 2011. 2

[65] Paul-Edouard Sarlin, Daniel DeTone, Tomasz Malisiewicz, and Andrew Rabinovich. SuperGlue: Learning feature matching with graph neural networks. In *CVPR*, 2020. 1, 2, 3, 5, 6, 7

[66] Manolis Savva, Abhishek Kadian, Oleksandr Maksymets, Yili Zhao, Erik Wijmans, Bhavana Jain, Julian Straub, Jia Liu, Vladlen Koltun, Jitendra Malik, et al. Habitat: A platform for embodied AI research. In *ICCV*, 2019. 5

[67] Johannes L Schonberger and Jan-Michael Frahm. Structure-from-motion revisited. In *CVPR*, 2016. 1

[68] Samarth Sinha, Jason Y Zhang, Andrea Tagliasacchi, Igor Gilitschenski, and David B Lindell. SparsePose: Sparse-view camera pose regression and refinement. In *CVPR*, 2023. 2

[69] 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. 15

[70] Jiaming Sun, Zehong Shen, Yuang Wang, Hujun Bao, and Xiaowei Zhou. LoFTR: Detector-free local feature matching with transformers. In *CVPR*, 2021. 1, 2, 3, 5, 6, 7, 8, 13, 19

[71] Bin Tan, Nan Xue, Tianfu Wu, and Gui-Song Xia. NOPE-SAC: Neural one-plane RANSAC for sparse-view planar 3d reconstruction. *TPAMI*, 2023. 2, 5, 6

[72] Zachary Teed and Jia Deng. DROID-SLAM: Deep visual SLAM for monocular, stereo, and RGB-D cameras. *NeurIPS*, 2021. 2

[73] Zachary Teed, Lahav Lipson, and Jia Deng. Deep patch visual odometry. *arXiv preprint arXiv:2208.04726*, 2022. 2

[74] Philip HS Torr and Andrew Zisserman. MLESAC: A new robust estimator with application to estimating image geometry. *Computer vision and image understanding*, 78(1):138–156, 2000. 4

[75] Jianyuan Wang, Christian Rupprecht, and David Novotny. PoseDiffusion: Solving pose estimation via diffusion-aided bundle adjustment. In *ICCV*, 2023. 1

[76] Wenshan Wang, Yaoyu Hu, and Sebastian Scherer. TartanVO: A generalizable learning-based VO. In *CoRL*, 2021. 2

[77] Tong Wei, Yash Patel, Alexander Shekhovtsov, Jiri Matas, and Daniel Barath. Generalized differentiable RANSAC. In *ICCV*, 2023. 2

[78] Ross Wightman. PyTorch image models. `https://github.com/rwightman/pytorch-image-models`, 2019. 5

[79] Zhenpei Yang, Jeffrey Z Pan, Linjie Luo, Xiaowei Zhou, Kristen Grauman, and Qixing Huang. Extreme relative pose estimation for RGB-D scans via scene completion. In *CVPR*, 2019. 2

[80] Zhenpei Yang, Siming Yan, and Qixing Huang. Extreme relative pose network under hybrid representations. In *CVPR*, 2020. 2

[81] Chao Yu, Zuxin Liu, Xin-Jun Liu, Fugui Xie, Yi Yang, Qi Wei, and Qiao Fei. DS-SLAM: A semantic visual SLAM towards dynamic environments. In *IROS*, 2018. 1

[82] Jason Y Zhang, Deva Ramanan, and Shubham Tulsiani. RelPose: Predicting probabilistic relative rotation for single objects in the wild. In *ECCV*, 2022. 2

[83] Chen Zhao, Yixiao Ge, Feng Zhu, Rui Zhao, Hongsheng Li, and Mathieu Salzmann. Progressive correspondence pruning by consensus learning. In *ICCV*, 2021. 2

[84] Yi Zhou, Connelly Barnes, Jingwan Lu, Jimei Yang, and Hao Li. On the continuity of rotation representations in neural networks. In *CVPR*, 2019. 3, 5, 8, 12

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

Appendix A. 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. Architecture is overviewed in Tables 6-7 and detailed in Tables 8-12.

Solver $\mathbf{T}_s$. Pose estimation from a correspondence estimator followed by a Kornia [61] 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). 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 12. 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 8 and Table 9, respectively.

FAR: One Round $\mathbf{T}_1$. Predicted 6DoF pose from one round of FAR, which consists of the weighted linear combination in 6D [84] space of $\mathbf{T}_t$ and $\mathbf{T}_s$, weighted by w as described in Equation 1. 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 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 C.4. In Table 8, 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.

Appendix B. 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 [25]. 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

Table 6. **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 7. **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 8. **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 [70] 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 [62] 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 9. **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 10. **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 11. **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 12. **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

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 resid-

uals under a standard Gaussian distribution as,

$$\beta'(\mathbf{T}_{\mathbf{H},k}, \mathbf{T}) = \log(\prod_{l=1}^{L} \exp(-r_l^2)/Z), \quad (3)$$

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,

$$\beta(\mathbf{H}, \mathbf{T}) = \max\left(\beta'(\mathbf{T}_{\{\mathbf{H},1\}}, \mathbf{T}), \beta'(\mathbf{T}_{\{\mathbf{H},2\}}, \mathbf{T})\right) \quad (4)$$

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,

$$\text{score}(\mathbf{H}) = \alpha\beta(\mathbf{H}, \mathbf{T}) + \sum_{(\mathbf{p},\mathbf{q})\in\mathbb{M}} \mathbb{1}\Big(\mathbf{E}(\mathbf{p}, \mathbf{q}|\mathbf{H}) < \sigma\Big), \quad (5)$$

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 $\mathbf{H}$

Appendix C. 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 [69] scheduler, except if using 6DReg backbone; here we follow prior work [2] 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.

C.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 use ground truth correspondence computed via LoFTR's correspondence algorithm from true pose and depth, which consists of a mutual nearest neighbor check.

C.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 C.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 $\mathbf{T}$.

C.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.

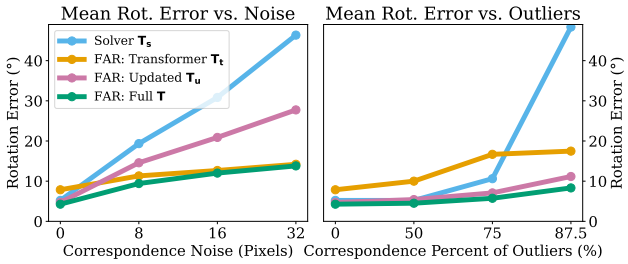

Figure 9. **Ground Truth Robustness Study on Matterport3D: Mean Rotation Error**. Using true correspondence, the solver has low mean error, which is nonzero because of some image pairs having limited ground truth correspondences, leading to small mean error. As with median error, adding noise or outliers 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.

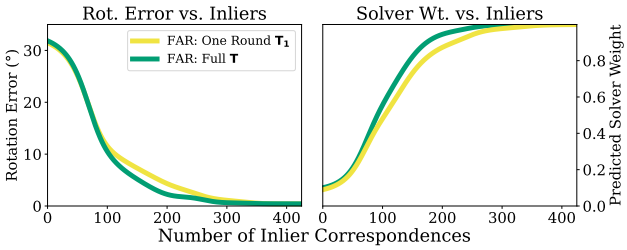

Figure 10. **FAR: Full $\mathbf{T}$ vs. FAR: One Round $\mathbf{T}_1$.** After one round, FAR produces high-quality results, making further improvement difficult. However, a second forward pass through the solver injected with a prior (FAR: Full $\mathbf{T}$) improves solver estimates. Correspondingly, 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).

C.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 C.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.

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. We nevertheless find this training beneficial.

Appendix D. Additional Results

D.1. Ground-Truth Robustness Experiment

Figure 9 presents mean rotation errors of methods confronted with ground truth correspondences, with varying amounts of noise and outliers. It corresponds to Figure 4, except mean rotation error is reported here rather than median rotation error in Figure 4. The results correspond to those in Figure 4: 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.

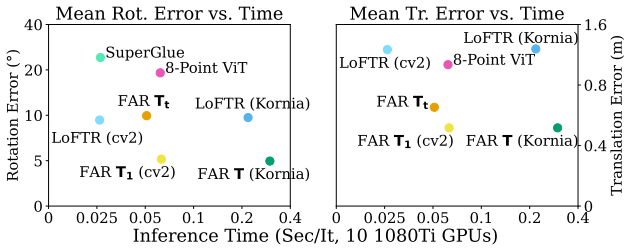

Figure 11. **Efficiency on Matterport (Log Scale vs. Log Scale).** The efficient frontier includes LoFTR+Solver, $\mathbf{T}_1$ and $\mathbf{T}$. FAR $\mathbf{T}_t$ is strictly better than 8-Point ViT. FAR $\mathbf{T}_1$ cuts error almost in half for little time cost. $\mathbf{T}$ improves further, but 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.

D.2. FAR: Full vs. FAR: One Round

Figure 10 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, 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 10, right) and the full model to improve (Figure 10, 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 C, FAR: Full $\mathbf{T}$ uses cv2's unbiased solver during the first round of computation for efficiency.

D.3. Inference Time Efficiency Comparison

We plot error vs. time in Figure 11. 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).

D.4. Random Results

Results on random examples are presented in Figures 12-15.

The comparisons are to the same baselines as in Figure 8. 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 Section 4 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 Section 4 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 Section 4 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 Section 4, 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 15, bottom left), while when L-B bests FAR, it is typically by less than three de-

grees. To summarize, random results elucidate FAR is both precise and robust.

Note results on Map-free Relocalization here, as well as Figure 8, 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.

D.5. Failure Cases

We can consider some failures in the random examples from Figures 12-15. 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 [11] 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.

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 12. **Random results on Matterport3D**. C+S: LoFTR [70] + Solver. L-B: LoFTR + 8-Point ViT [62]. 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 13. **Random results on Map-free Relocalization**. C+S: LoFTR + Solver. L-B: 6D Reg [2]. 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 14. **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.

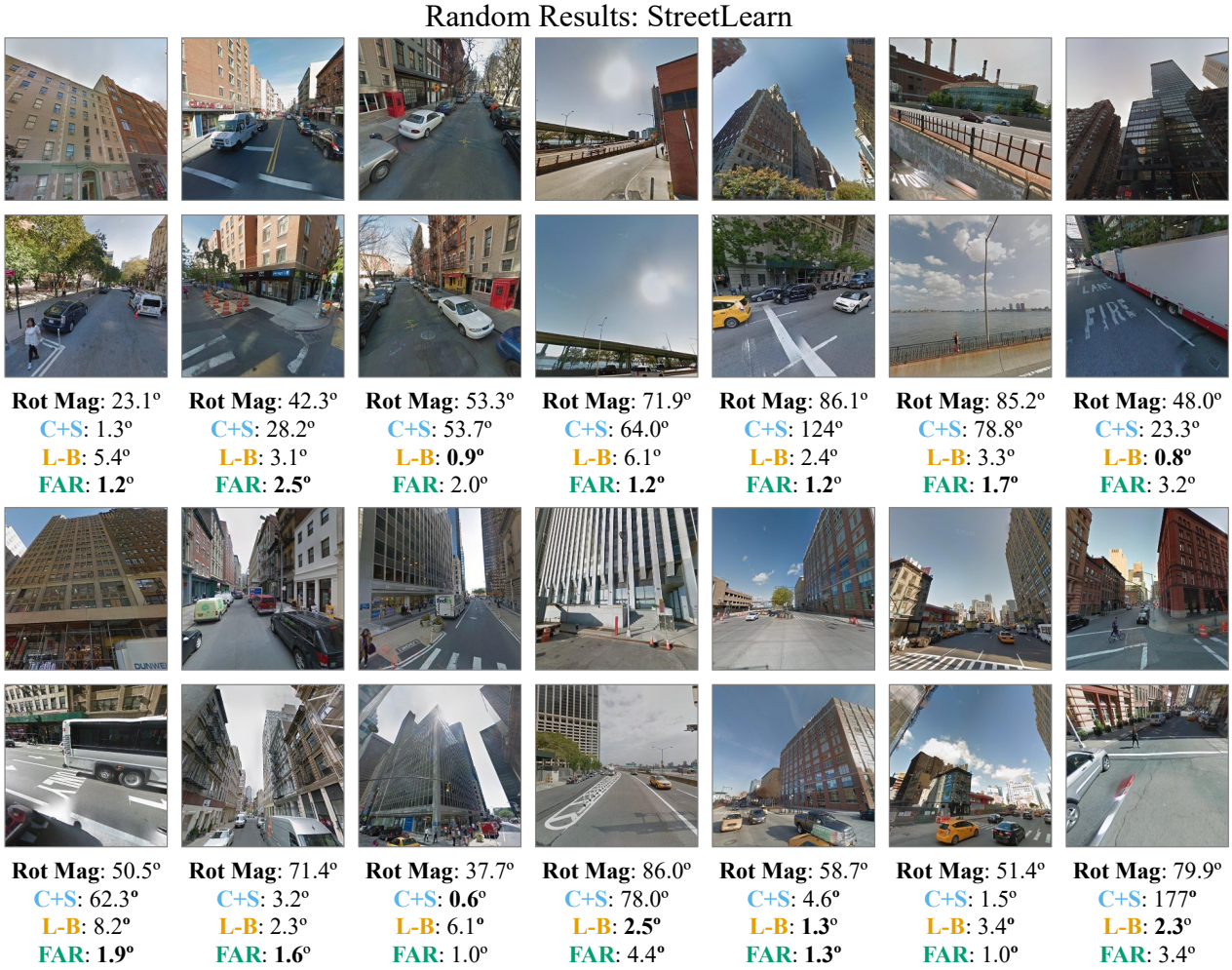

Figure 15. **Random results on StreetLearn**. C+S: LoFTR + Solver. L-B: 8-Point ViT. FAR: uses 8-Point ViT features and LoFTR correspondences. FAR often has the lowest error – here 8 times vs. 1 for C+S and 5 for L-B; and is more robust than alternatives: FAR has maximum error of 4.4°, L-B has maximum 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.