Supplementary Material: Self-supervised Correspondence Estimation via Multiview Registration

Mohamed El Banani[1]* Ignacio Rocco[2] David Novotny[2] Andrea Vedaldi[2]
mbanani@umich.edu irocco@meta.com dnovotny@meta.com vedaldi@meta.com
Natalia Neverova[2] Justin Johnson[1,2] Ben Graham[2]
nneverova@meta.com justincj@umich.edu benjamingraham@meta.com
[1]University of Michigan [2]Meta AI

1. Implementation Details

We include a subset of our source code in the supplemental material. The submitted version was taken from a larger code base and edited to improve clarity through comments and remove any identifying information. Our approach is implemented in PyTorch [10], but we make heavy use of PyKeOps [1] and FAISS [8] for fast CUDA implementations of kNN, as well as PyTorch3D [11] and Open3D [12] for 3D transformations and alignment. We discuss several key design choices below and refer the reader to our code submission for the exact details. For more details on SE(3) Transformation Synchronization, we refer the reader to Sec. 3.

Feature Extraction. We use a modified ResNet-18 [7] as our feature extractor. Since the ResNet architecture was designed for image classification, it performs aggressive downsampling to reduce the spatial dimension of the feature grid to allow the network to be more light-weight while increasing the receptive field for each pixel. However, our application would benefit from maintaining a high resolution to allow for accurate matching. As a result, we modify ResNet to remove most of the down-sampling, only down-sampling by a factor of $2\times$ twice within the network. During training, we down-sample the input to a dimension of 240×320. We find that this allows us to increase the speed of training, without impacting the test-time performance on images of resolution 480×640. We set the output feature dimension to 128. While previous work on self-supervised learning was restricted to small feature dimensions due to a slow kNN implementation [4, 5], we use the faster kNN implementations provided by PyKeOps and FAISS.

Correspondence Estimation. We use the kNN functions provided by FAISS [8] and PyKeOps [1] in our implementation. While FAISS provides a faster kNN implementation, PyKeOps provides more flexibility in the distance function that can be used. As a result, we use FAISS for the initial feature-based correspondence estimation, and use PyKeOps for finding kNN based on both features and geometry for the geometry-aware ratio test. In all cases, we filter the correspondence and only keep the top 500 correspondences.

Confidence Threshold. We only apply confidence thresholding to non-adjacent frames; *i.e.*, $|i-j|>1$. We do this as we find that some adjacent pairs can still have a low pair-wise confidence despite having large overlap. Through excluding adjacent pairs from thresholding, we can guarantee that synchronization is possible for all sequences. We set the confidence threshold to $\gamma=0.4$ which allows us to ensure large overlap as shown in Figure 3 in the main paper.

Refinement. Given the synchronized views, we resample correspondences based on both feature similarity and spatial proximity of points. This allows us to sample better correspondences as shown in Figure 4 of the main paper. We set the weighting between feature and spatial distance to $\alpha=10.0$ based on preliminary experiments.

* Work done during an internship at Meta AI.

2. Qualitative Results

We include additional qualitative results to provide a better sense of our model's performance. We also clarify some of the color schemes used throughout the paper.

Correspondence color. We color-code our correspondence using their 3D error. Specifically, correspondences with an error of less than 5 cm were plotted in dark green, errors between 5 cm and 10 cm were plotted in yellowish green, errors between 10 cm and 15 cm were plotted in orange, and errors larger than 15 cm were plotted in red.

Correspondence Estimation. We provide additional qualitative examples of correspondence estimation results in Fig. 2. We find that for easy cases that involve the camera panning or zooming, all approaches perform fairly well (rows 1-3). Meanwhile, cases with large camera rotation can be challenging to all models, with different models failing for different cases. We find that our model can overcome those challenges in some cases where some prior approaches have limited performance (rows 4-9). In cases with repeated textures, our model can inaccurately predict a consistent set of correspondences that are accurate, as shown in row 10 of Fig. 2. We find that LoFTR can succeed in such a case, likely due to its use of cross-attention, which is noted by the authors of both LoFTR and SuperGlue. Future iterations of self-supervised correspondence estimation should explore the incorporation of attention modules and integrating it with geometric-aware matching. Finally, we observe that some cases are challenging to all models, especially when there is a very large camera motion such as looking at the same object from opposing sides (row 11) or when there is limited overlap and plain textures (row 12).

Correspondence Refinement. We provide qualitative examples of estimated correspondences before and after refinement in Fig. 3. In many cases, the initial feature-based correspondences are already fairly accurate. In those cases, we find that refinement results in the correspondences being more spread out and increasing in accuracy. More interesting cases involve a very noisy initial set that can be refined into a dense, accurate set of correspondences. This can be seen clearly in rows 5-7 in Fig. 3. Finally, in some difficult cases, our initial estimation is extremely noisy, and our model is unable to recover from that.

3. Camera Synchronization

Here we explain the Camera Synchronization algorithm (Section 3.4) in a bit more detail.

Notation for SE(3) matrices. Recall that for pairs of frame $i < j$,

$$\mathbf{T}_{i,j} = \underset{\mathbf{T} \in \mathrm{SE}(3)}{\arg\min} \sum_{\text{inliers } (p,q,w) \in \mathcal{C}_{i,j}} w \|\mathbf{x}_q - \mathbf{T}(\mathbf{x}_p)\|_2^2$$

is our estimate for the relative transformation from camera i to camera j. We can write $\mathbf{T}_{i,j}$ as a 4x4 matrix consisting of a rotation and translation,

$$\mathbf{T}_{i,j} = \left[\begin{array}{c|c} R & 0 \\ \hline t & 1 \end{array}\right], \qquad R \in \mathrm{SO}(3),\ T \in \mathbb{R}^3.$$

$\mathbf{T}_{i,j}$ acts on points $\mathbf{x} = (x_1, x_2, x_3, 1)$ in homogeneous form by right multiplication

$$\mathbf{T}_{i,j}(\mathbf{x}) = (x_1, x_2, x_3, 1) \times \mathbf{T}_{i,j}.$$

Confidence-weighted transformations. Recall that $c_{i,j}$ is a confidence value attached to $\mathbf{T}_{i,j}$ for $i < j$. Let $\mathbf{S}_+ \subset \mathbf{R}^{4\times 4}$ denote the set of 4×4 matrices with the form:

$$\alpha \left[\begin{array}{c|c} R & 0 \\ \hline t & 1 \end{array}\right], \qquad \alpha \geq 0, R \in \mathbb{R}^{3\times 3},\ T \in \mathbb{R}^3.$$

Elements of $\mathbf{S}_+$ can be projected onto SE(3) by dividing by α, and then using SVD to project R onto SO(3).

Note that $\mathbb{S}_+$ is closed in the sense that if $A, B \in \mathbf{S}_+$ and $\alpha \geq 0$, then $A + B$, $A \times B$ and αA are all in $\mathbf{S}_+$ too.

Confidences as jump probabilities We will make two simpifying assumptions. First, we will assume that the $c_{i,j}$ have been scaled so that the rows sum to one: for all i, $\sum_j c_{i,j} = 1$. Second, we assume that for each i, $c_{i,i+1} > 0$. With these assumptions in place, $C = [c_{i,j}]$ is the stochastic matrix for an N-state Markov chain (X_t),

$$c_{i,j} = \mathbb{P}[X_{t+1} = j \mid X_t = i], \quad t = 0, 1, 2, \ldots.$$

The Markov chain is [9]:

- lazy: $\mathbb{P}[X_1 = i \mid X_0 = i] = c_{i,i} = 1/2$ as $c_{i,i} = \sum_{j \neq i} c_{i,j}$,
- connected: for all i, j, for some t sufficiently large $\mathbb{P}[X_t = j \mid X_0 = i] = (C^t)_{i,j} > 0$, and
- time-reversible: $\pi_i C_{i,j} = \pi_j C_{j,i}$ for all i, j with $\pi \in [0,1]^N$ the equiilibrium distribution.

By the Perron–Frobenius theorem, and the laziness property, the eigenvalues of C can be written as

$$1 = \lambda_1 > \lambda_2 \geq ... \lambda_N \geq 0. \tag{1}$$

The spectral gap $1 - \lambda_2 > 0$ so convergence to equilibrium is exponential,

$$\mathbb{P}[X_t = j \mid X_0 = i] = (C^t)_{i,j} = \pi_j + \mathrm{O}(\lambda_2^t).$$

The pairwise-transformations matrix In Section 3.4, equation (5), we define a $4N \times 4N$ matrix $\mathbf{A}$,

$$\mathbf{A} = \begin{bmatrix} c_{1,1}\mathbf{I}_4 & c_{1,2}\mathbf{T}_{1,2} & \cdots & c_{1,N}\mathbf{T}_{1,N} \\ c_{2,1}\mathbf{T}_{2,1} & c_2\mathbf{I}_4 & \cdots & c_{2,N}\mathbf{T}_{2,N} \\ \vdots & \ddots & & \vdots \\ \vdots & & \ddots & \vdots \\ c_{N,1}\mathbf{T}_{N,1} & c_{N,2}\mathbf{T}_{N,2} & \cdots & c_{N,N}\mathbf{I}_4 \end{bmatrix} \in \mathbf{S}_+^{N \times N} \subset \mathbb{R}^{4N \times 4N},$$

consisting of an $N \times N$ grid of elements of $\mathbf{S}_+$. To motivate the definition of $\mathbf{A}$, we can interpret it as generating a random walk on the set $\{1, 2, \ldots, N\} \times SE(3)$,

$$\mathbb{P}[(X_{t+1}, Y_{t+1}) = (j, Y_t \times \mathbf{T}_{i,j}) \mid X_t = i] = c_{i,j}, \qquad t = 0, 1, \ldots$$

The expected value in $\mathbf{S}_+$ of the Y-component of the walk is a weighted sum of the products of pairwise transformations. Pairwise transformations with greater confidence contribute more strongly to the sum.

The solution to the synchronization problem is related to the eigenvectors of $\mathbf{A}$. *If* there is a global collection of cameras $(\mathbf{T}_i)$ such that $\mathbf{T}_{i,j} = \mathbf{T}_i^{-1}\mathbf{T}_j$, then $\mathbf{A}$ will have four independent eigenvectors with eigenvalue one, i.e.

$$[\mathbf{T}_1 \ldots \mathbf{T}_N] \times (\mathbf{A} - \mathbf{I}_{4N}) = \mathbf{0}, \qquad [\mathbf{T}_1 \ldots \mathbf{T}_N] \in \mathrm{SE}(3)^N \subset \mathbb{R}^{4 \times 4N}.$$

All other eigenvalues λ will satisfy $|\lambda| \leq \lambda_2$ (c.f. inequality (1) for the eigenvalues of Markov chain X_t, and properties of matrix determinants). As integer $k \to \infty$, each of the N rows of $\mathbf{A}^k$ will converge to a globally consistent set of cameras. The different rows will yield essentially the same solution, but differing by an $\mathrm{SE}(3)$ transformation of the global coordinates.

More generally, if no such perfect solution exists, then we want to find

$$\begin{aligned} &\underset{\{\mathbf{T}_i \in \mathrm{SE}(3):1 \leq i \leq N\}}{\arg\min} \|[\mathbf{T}_1 \ldots \mathbf{T}_N] \times (\mathbf{A} - \mathbf{I}_{4N})\|_F^2 \\ &= \underset{\{\mathbf{T}_i \in \mathrm{SE}(3):1 \leq i \leq N\}}{\arg\min} \sum_j \left\| \mathbf{T}_j - \sum_i c_{i,j}\mathbf{T}_i\mathbf{T}_{i,j} \right\|_F^2 \\ &= \underset{\{\mathbf{T}_i \in \mathrm{SE}(3):1 \leq i \leq N\}}{\arg\min} \sum_{i,j} c_{i,j} \|\mathbf{T}_j - \mathbf{T}_i\mathbf{T}_{i,j}\|_F^2 . \end{aligned}$$

The solution in [6] involves calculating an eigen decomposition directly. Let $\mathbf{A}^{\text{rot}}$ denote the $3N \times 3N$ matrix obtained from $\mathbf{A}$ by taking the top 3×3 elements from each sub-block of A. In the notation of [6, supplementary Sec. 2], our $\mathbf{A}^{\text{rot}}$ is equal to their "$L/2 + D$". Each of the $3N$ eigenvectors of $\mathbf{A}^{\text{rot}}$ (suitably padded with zeros to increase their length from $3N$ to $4N$, e.g.

$$[x_1, x_2, x_3, \ldots, x_{3N-2}, x_{3N-1}, x_{3N}] \to [x_1, x_2, x_3, 0, \ldots, x_{3N-2}, x_{3N-1}, x_{3N}, 0]),$$

becomes an eigenvectors of $\mathbf{A}$. Three of these eigenvectors with largest eigenvalues, projected onto $SO(3)$, solve [6, Eq. 5],

$$\underset{\{R_i \in SO(3):1 \leq i \leq N\}}{\arg\min} \sum_{i,j} c_{i,j} \|R_j - R_i \times (\mathbf{T}_{i,j})_{1:3,1:3}\|_F^2$$

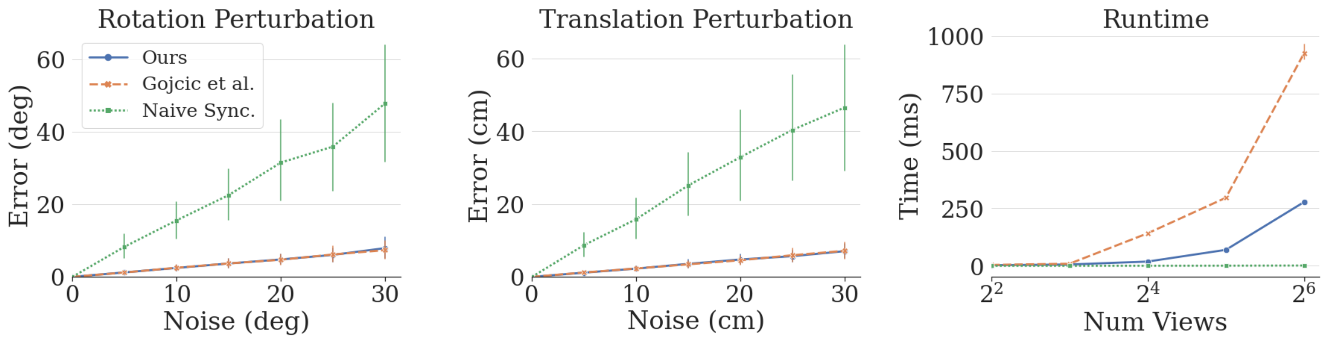

Figure 1. **Synchronization Benchmark.** Our approach achieves the same error as Gojcic *et al*. [6], while being faster and more numerically stable.

Rather than computing the eigendecomposition of $\mathbf{A}$ directly, we instead use power-iteration. Raising $\mathbf{A}$ to large powers filters out the effect of the smaller eigenvalues. To do this efficiently, starting from $\mathbf{A}$, we repeatedly takes squares to calculate $\mathbf{A}^2$, then $\mathbf{A}^4$, and so on until $\mathbf{A}^{2^t}$. Each element in $\mathbf{A}^{2^t}$ is then projected into $\mathrm{SE}(3)$ using SVD as described above; call the resulting matrix $\mathbb{A}$. $\mathbb{A}$ is composed of N 'rows', each with shape $4 \times 4N$; each of these rows is n approximate solution to the synchronization problem. The difference between the rows is that each row is centered around a different camera. We choose $t = O(\log N)$ so $2^t > N$; the the number of FLOPs needed to calculate $\mathbb{A}$ is thus $\mathrm{O}(N^3 \log N)$. In practice, the time spent on synchronization is small compared to feature extraction and matching, as synchronization is independent of the resolution of the images. For very larger N, a database of key-frames could be used to reduce the size of N.

Numerical stability. Empirically, we find that training using synchronizations extracted from $\mathbf{A}^{2^t}$ was stable. Using an eigensolver to implement the method of [6] led to exploding gradients. The derivative with respect to a set of eigenvectors is unstable when the eigenvalues a clustered together, as is normally the case with the largest eigenvalues of $\mathbf{A}^{\mathrm{rot}}$; when the pairwise rotations are compatible, the largest eigenvalues will be approximately equal.

Performance. We compare our synchronization approach to naive synchronization which aggregates the transformations using adjacent views and the eigendecomposition approach proposed by Gojcic *et al*. [6]. We compare the three algorithms on their ability to handle rotation and translation perturbation in the pairwise estimates as well as their runtime. As seen in Fig. 1, our approach achieves the same performance as the eigendecomposition approach while being faster. Both approaches greatly naive synchronization since they are able to use information from all pairs. Furthermore, since our approach only relies on power iteration, it does not suffer from the numerical instability in the backward gradient discussed above.

4. Ethical Considerations and Societal Implications

Our work presents a method for self-supervised learning of correspondence estimation from RGB-D video. Our main contribution is to demonstrate how multiview registration could be used to learn better features from RGB-D video that perform on-par with supervised learning approaches. Advancements in this area can enable more powerful feature learners, which can improve the overall performance of larger frameworks that use it such as SLAM or structure-for-motion. Our technical contributions pertain to allowing models to learn from a different type of data, as a result, our societal impact is mediated by the kind of data that we currently train on as well as the data that we can train on in the future.

In this work, we evaluated our approach on ScanNet [2]. This is a large scale dataset of indoor scenes that contains over 1500 RGB-D sequences taken at more than 700 locations. The data was collected by 20 volunteers across several countries with most sequences captured in the United States of America and Germany. Each volunteer used a specialized capture setup to record video sequences in private locations to which they had access. With very few exceptions, the data set is made up of empty rooms. One salient issue with ScanNet is that most of the locations come from areas with a relatively high household income: houses, offices, and university housing in major western cities. This is problematic given that prior work has shown that computer vision models trained on data from countries with high household income generalize poorly to images coming from countries with a lower mean household income [3]. Since our models were all trained on this dataset, we would expect them to generalize poorly to images from outdoor scenes or indoor scenes coming from different geographic areas or demographics. We note that while this could be alleviated by training our model on datasets coming from other countries, we are unaware of any large RGB-D video datasets that would meet such criteria.

While our approach was trained on ScanNet, we developed a self-supervised approach since we hope that it can scale to large-scale data. Given the increasing prevalence of RGB-D sensors on phones, we can expect that people will start uploading videos to the web. Such videos will not be careful scans of scenes as the ones captured for ScanNet, and hence will not be well-suited for 3D reconstruction algorithms. Our goal is to build systems that could easily leverage this data by being self-supervised. However, such web-data will introduce other complications. Below, we reflect on two ethical considerations: structural biases in representation and privacy.

Structural Bias in Representation. While the RGB-D cameras are becoming more available to consumers, their relatively high cost means that that they will be adopted more by wealthier demographics. Hence, while we expect that the uploaded videos will be more diverse than ScanNet, the high cost of the devices means that they will likely be used primarily by wealthier individuals resulting in a bias in representation both across and within different countries. As a result, anyone considering using this technology for learning features should be aware of such limitations, which could potentially be alleviated by a more thoughtful collection process.

Privacy Concerns. People often upload videos that include other individuals who may have not consented for videos to be uploaded or even captured with them. Such videos will inevitably include such individuals. This means that to scale our approach to such data, we need robust filtering mechanisms that can detect such shots and exclude them from training. This is a common challenge for any self-supervised method that hopes to scale to web data.

References

[1] Benjamin Charlier, Jean Feydy, Joan Alexis Glaunès, François-David Collin, and Ghislain Durif. Kernel operations on the gpu, with autodiff, without memory overflows. *JMLR*, 22(74):1–6, 2021. 1

[2] Angela Dai, Angel X. Chang, Manolis Savva, Maciej Halber, Thomas Funkhouser, and Matthias Nießner. ScanNet: Richly-annotated 3D Reconstructions of Indoor Scenes. In *CVPR*, 2017. 4

[3] Terrance de Vries, Ishan Misra, Changhan Wang, and Laurens van der Maaten. Does object recognition work for everyone? In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, pages 52–59, 2019. 4

[4] Mohamed El Banani, Luya Gao, and Justin Johnson. UnsupervisedR&R: Unsupervised Point Cloud Registration via Differentiable Rendering. In *CVPR*, 2021. 1

[5] Mohamed El Banani and Justin Johnson. Bootstrap Your Own Correspondences. In *ICCV*, 2021. 1

[6] Zan Gojcic, Caifa Zhou, Jan D. Wegner, Leonidas J. Guibas, and Tolga Birdal. Learning multiview 3d point cloud registration. In *CVPR*, 2020. 3, 4

[7] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *CVPR*, pages 770–778, 2016. 1

[8] Jeff Johnson, Matthijs Douze, and Hervé Jégou. Billion-scale similarity search with gpus. *IEEE Transactions on Big Data*, 7(3):535–547, 2021. 1

[9] David A. Levin, Yuval Peres, and Elizabeth L. Wilmer. *Markov chains and mixing times*. American Mathematical Society, 2006. 2

[10] Adam Paszke, Soumith Chintala, Ronan Collobert, Koray Kavukcuoglu, Clement Farabet, Samy Bengio, Iain Melvin, Jason Weston, and Johnny Mariethoz. Pytorch: Tensors and dynamic neural networks in python with strong gpu acceleration, may 2017. 1

[11] Nikhila Ravi, Jeremy Reizenstein, David Novotny, Taylor Gordon, Wan-Yen Lo, Justin Johnson, and Georgia Gkioxari. Accelerating 3d deep learning with pytorch3d. *arXiv*, 2020. 1

[12] Qian-Yi Zhou, Jaesik Park, and Vladlen Koltun. Open3D: A modern library for 3D data processing. *arXiv*, 2018. 1

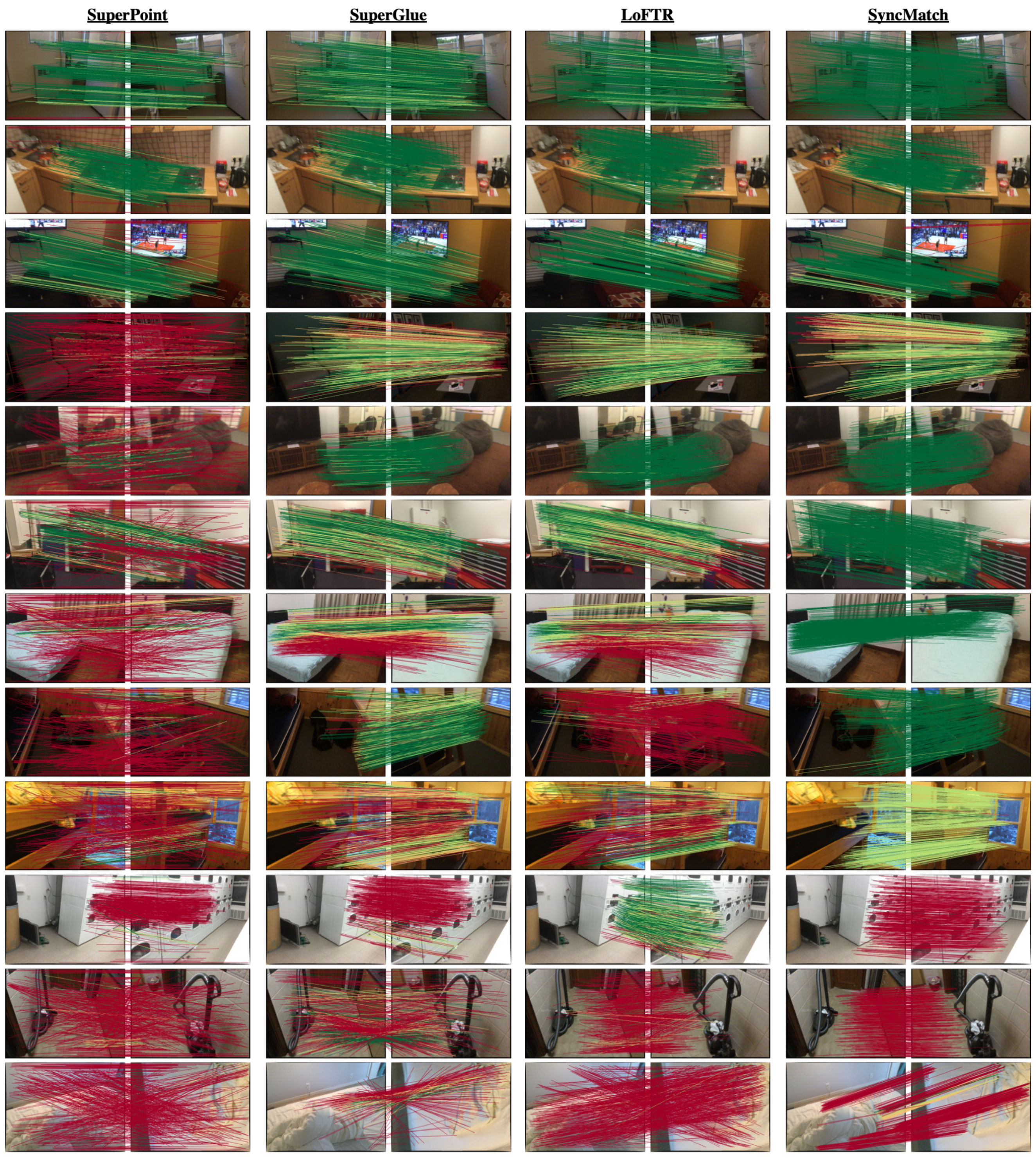

Figure 2. **Correspondence Estimation.** We present correspondence results on a variety of situations. The top three rows show a variety of positive examples where all models performs well. The following six column present present cases where our model succeeds and other models perform poorly. Those are typically cases with large camera motion where our model is capable of use the geometric information and learned features to predict accurate correspondences. Finally, we report some challenging cases for our model: repetitive textures (row 10), very large camera motion (row 11), limited overlap and plain textures (row 12). While such cases are often challenging for all models, prior approaches like SuperGlue and LoFTR can sometimes produce good correspondence for some instances.

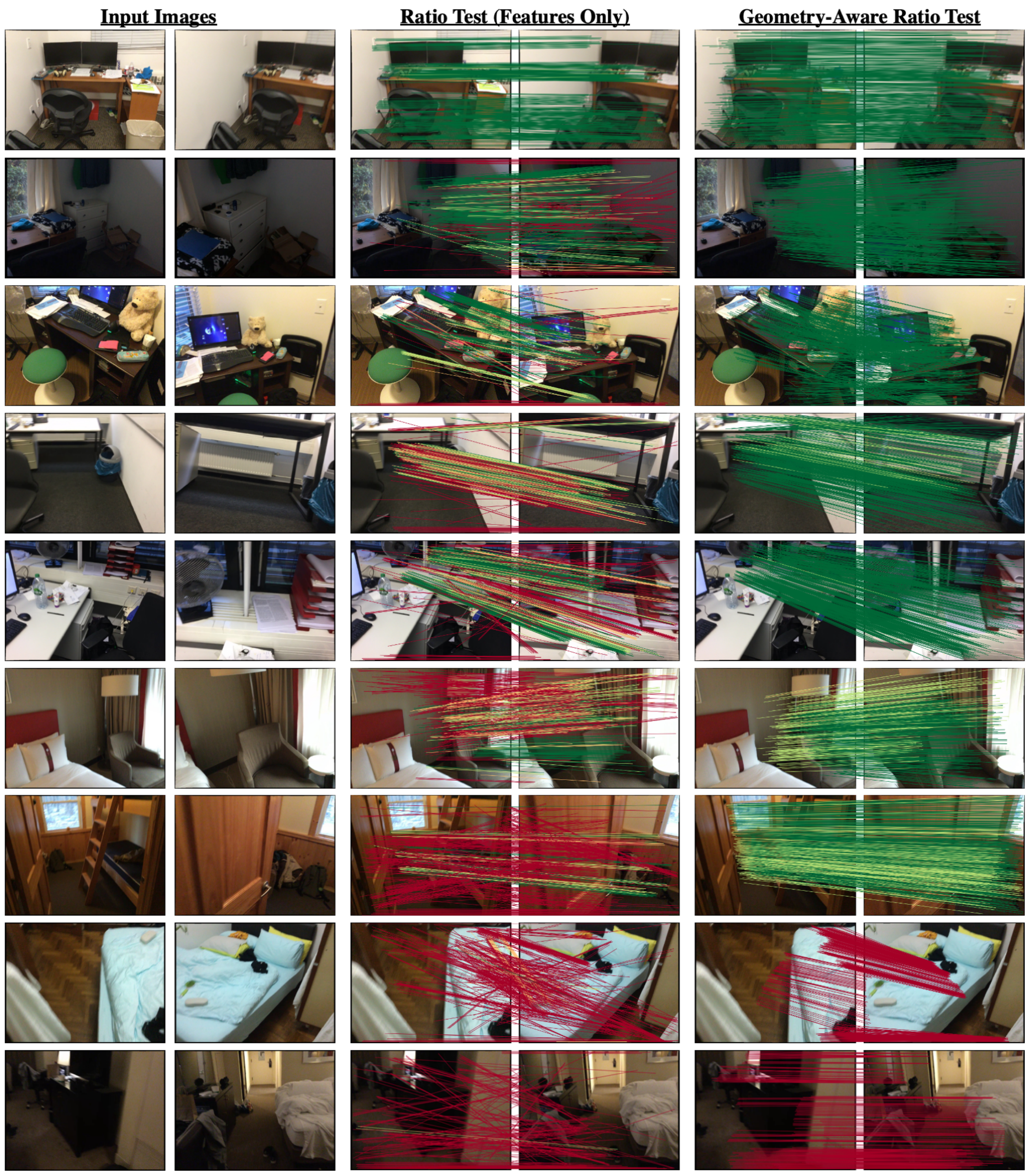

Figure 3. **Correspondence Refinement.** We show the impact of correspondence refinement using depth. We present several modes of performance. The top three rows present cases where the feature-based correspondences were already accurate and incorporating geometry simply improved the accuracy. The following three four rows cases where refinement had a large impact on correspondence quality by leveraging a small subset of accurate correspondences to align the scenes and then sample a more accurate set of correspondences. Finally, we observe failure cases where the initial set is so noisy that the model cannot generate a good transformation estimate rendering refinement useless.