Learning Robust Correspondences For Relative Pose Estimation

Marc Benedí San Millán
Technische Universität München
marc.benedi@tum.de

Prof. Dr. Matthias Nießner
Technische Universität München
niessner@tum.de

Abstract

We present an end-to-end learnable, differentiable method for pairwise relative pose registration of RGB-D frames. Our method is robust to big camera motions thanks to a self-supervised weighting of the predicted correspondences between the frames. Given a pair of frames, our method estimates matches of points and their visibility score. A self-supervised model predicts a confidence weight for visible matches. Finally, visible matches and their weight are fed into a differentiable weighted Procrustes aligner which estimates the rigid transformation between the input frames.

1. Introduction

Relative camera pose estimation is a core component in many applications in computer vision, robotics and computer graphics. The range of applications varies from 3D rigid reconstruction, structure from motion (SfM), camera relocalization and many others. The problem consists in estimating the rigid transformation that aligns two inputs: source and target.

Classic approaches follow a similar pipeline [17]: (1) feature extraction; (2) feature matching among frames; (3) estimate a rigid transformation $T \in \mathbb{SE}(3)$ that aligns the correspondences.

Classic methods require considerable overlap between the RGB frames and fail when the camera motion is big. We developed a method that is robust to such cases by estimating a weight for each match. These weights are used to downvote poor correspondences such that their impact during pose estimation is minimized.

The proposed method has three components: (1) We generate matches between the two RGB frames and predict a visibility score for each one of them; then, (2) a dCNN predicts a confidence weight for each match; and finally, (3) a differentiable Weighted Procrustes aligner regresses the rigid transformation between the two cameras.

The rest of the paper is organized as follows: Sec.2 presents the related work. Sec. 3 presents the rigid reconstruction problem and describes the proposed method. Sec. 4 presents the experiments and results. Finally, Sec. 5 concludes the paper.

2. Related Work

2.1. Feature-based correspondence matching

As previously mentioned, the second step in many reconstruction pipelines is matching the extracted features from the input frames. Traditionally, these methods were hand-crafted descriptors. For RGB images, descriptors and matching mechanisms such as SIFT[11], SURF[1] and ORB[13] have been historically used. However, these descriptors tend to perform poorly in large viewpoint changes (where image overlap is small), textureless objects or repetitive patterns.

Recent state-of-the-art methods leverage the use of CNNs which learn how to extract features from images. outperform hand-crafted descriptors

Recent state-of-the-art methods leverage the use of CNNs that learn how to extract and match image features. These methods have shown to be more resilient to situations where hand-crafted methods fail. Fischer and Dosovitskiy [7] proposed a method using a CNN that outperforms sift on the descriptor matching task.

Božič *et al.* [4] presented a method for non-rigid reconstruction. In their work, they propose a dCNN architecture to predict a likelihood heatmap around the correspondence point. We based our Correspondence and Visibility prediction component (Sec. 3) in their work.

2.2. Pose optimization

Pose optimization consists on finding a transformation that aligns two objects. Iterative Closest Points (ICP) [2] is an iterative algorithm that estimates a rigid transformation which minimizes the difference between two point clouds. A single iteration performs two steps: (1) Find correspondences; and (2) Estimate a transformation such that pairwise distance between correspondences is minimized.

Kendall *et al.* [10] introduced a method for absolute pose regression using CNNs to extract features from RGB im-

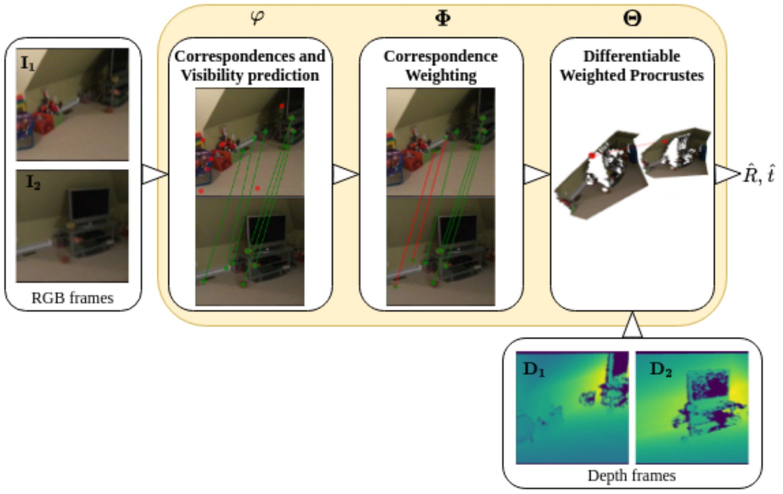


Figure 1. Pipeline of our method. Given a pair of RGB-D images, I_1, D_1 I_2, D_2, we estimate the relative pose between these frames as $\hat{R} \in SO(3)$ and $\hat{t} \in \mathbb{R}^3$. First, I_1, I_2 are fed into the *Correspondence and visibility prediction* component, the visible predicted correspondences are weighted in the *Correspondence Weighting* component. Finally, they are back-projected into 3D and feed into the *Weighted Procrustes aligner* which estimates the relative pose.

ages and regress the 6-DOF pose using fully connected layers. They leveraged transfer learning from a network trained for classification. Inspired by it, other methods approached relative pose regression.

Melekhov *et al*. [12] presented a model for camera relocalisation regressing relative poses between the sensor's input and a database of known locations. They built a Siamese network encoder with shared weights and fully connected layers as the final regressor. Unlike absolute pose, relative pose estimation can generalize to unseen scenes [14]. RCPNet is another example of a method directly estimating the relative pose using a CNN encoder and FC layers as a regressor [16].

2.3. Outlier filtering/Robust optimization

Bad correspondences are common when produced with feature matching methods. This outliers have a strong negative impact during the alignment estimation phase. It is necessary to filter them out before that. Fischler and Bolles [8] proposed a general parameter estimator to cope with outliers in the data. Their method, RANSAC, aims at estimating a solution that fits as much as possible the input data by using the minimum number of correspondences.

Other methods use deep learning techniques to filter outliers. Choy *et al*. [5] presented a CNN for correspondence confidence prediction. They feed the predicted confidences to a Weighted Procrustes aligner to estimate relative poses between 3D scans. Božič *et al*. [3] also used a CNN-based architecture to learn correspondence weighting in a self-supervised manner.

3. Method

In this section, we will outline our method for regressing relative poses from image pairs. We define the *target image* as I_1 and the *source image* as I_2. Likewise, we define the *target and source depth maps* as D_1 and D_2. Finally, we aim at estimating the relative pose that aligns *source* to *target* as a 3D rotation $R \in SO(3), (R \in \mathbb{R}^{3\times3}, R^T R = I, det(R) = 1)$, and a translation vector $t \in \mathbb{R}^3$.

I_1, I_2 are of size 224×224 with 3 color channels. D_1, D_2 have the same dimensions but only a single channel containing the depth information. Thus, we define $H = 224$ and $W = 224$.

3.1. Overview

Given two RGB frames I_1, I_2 and their respective depth maps D_1, D_2, our method performs the following steps:

1. Select $Q := \{q_1, ..., q_n\}$ a set of n query pixels from I_1.
2. Using the *correspondence and visibility prediction* component (φ), predict a set $M := \{m_1, ..., m_n\}$ of pixels from I_2 such that q_i is the predicted match of m_i. Additionally, we predict a visibility score for each match $V := \{v_1, ..., v_n\}, v_i \in [0, 1]$.
3. A match m_i is considered occluded in I_2 if $v_i < 0.5$ and visible otherwise. We mask out the non-visible matches from Q and M obtaining Q', M' of length m.
4. The *correspondence weighting* module (Φ) predicts a confidence score for all pairs $(q'_i, m'_i), q'_i \in Q', m'_i \in$

M'. We name this set of weights as $W := \{w_1, ..., w_m\}$.

5. Using the camera parameters, Q' and M' are back-projected from pixels to 3D points. We define Y and X as the set of back-projected pixels for Q' and M' respectively.

6. X, Y and W are fed into the *weighted Procrustes aligner* (Θ) which outputs the estimated rotation $\hat{R}$ and translation $\hat{t}$.

We want $R, \hat{R}$ and $t, \hat{t}$ to be as close as possible.

3.2. Correspondence and visibility prediction (φ)

The goal of this component is to predict the coordinates of a pixel m in I_2 that corresponds to a pixel q from image I_1 and a visibility score v. We based the architecture of this component on the work of Božič *et al.* [4].

To feed q to the network, we generate a heatmap $\mathcal{H}_q$ of dimensions $H \times W$. $\mathcal{H}_q[q] = 1.0$ and the rest of the values decay to zero following a Gaussian Kernel $G(q; \sigma = 7)$ centered around q with standard deviation seven pixels. Similarly, at train time, we generate the heatmap $\mathcal{H}_{gt}$ around the ground-truth match m.

$$\begin{aligned}&\varphi : \mathbb{R}^{H\times W\times 3} \times \mathbb{R}^{H\times W\times 3} \times \mathbb{R}^{W\times W} \rightarrow \mathbb{R}^{H\times W} \times \mathbb{R},\\ &\varphi(I_1, I_2, \mathcal{H}_{\text{q}}) \rightarrow (\hat{\mathcal{H}}, \hat{v})\end{aligned} \tag{1}$$

We want the network to predict the likelihood of the pixel coordinates of q in I_2. For this, we will apply a sigmoid activation function σ_{sg} to the predicted heatmap $\hat{\mathcal{H}}$ in order to map all values between zero and one. Additionally, we apply the softmax activation function σ_{sg} so that it creates a complete probability distribution. However, some query pixels may not be visible in I_2. This could be due the match m being outside the image frame or it being occluded. For this reason, we add a head to the network which predicts the visibility score v. We apply a sigmoig activation function σ_{sg} to v to map it to the range $[0, 1]$.

During optimization, we give more importance to the pixels closer to m. For that, we define a pixel weighting function as: $w_{\mathcal{H}}(p) = 1 + 10G(m; \sigma = 7)(p)$.

We used the following loss for correspondence and visibility prediction:

$$\begin{aligned}\mathcal{L}_{\mathcal{H}} = &\sum_i \Phi_{bce}(w_{\mathcal{H}}(\sigma_{\text{sg}}(\mathcal{H}) - \mathcal{H}_{\text{gt}})) +\\ &\lambda_{\text{nll}} \sum_i \Phi_{nll}(w_{\mathcal{H}}(\sigma_{\text{sm}}(\mathcal{H}) - \mathcal{H}_{\text{gt}}))\end{aligned} \tag{2}$$

Where Φ_{bce} denotes the binary cross entropy loss and Φ_{nlll} the negative log-likelihood. We use the same $\lambda_{nll} = 10$ as the original authors [4].

Finally, the output heatmap $\hat{\mathcal{H}}$ is computed as the Hadamard product $\hat{\mathcal{H}} = \sigma_{\text{sg}}(\mathcal{H}) \otimes \sigma_{\text{sm}}(\mathcal{H})$. For visibility prediction, we apply $_{\text{sg}}$ to the output of the bottleneck. We then use binary cross entropy to compute the loss:

$$\mathcal{L}_{\mathcal{V}} = \sum_i \Phi_{bce}\left(\hat{v} - v\right)$$

3.3. Correspondence weighting (Φ)

Following a similar approach as Neural Non-Rigid Tracking [3], we add correspondence weighting to the predicted matches to enable a more robust optimization.

Given a set of visible ($v_i > 0.5$) query points Q' and their predicted matches M', the *correspondence weighting* component outputs a confidence score $W := \{w_1, ..., w_m\}$ for each matched pair.

$$\begin{aligned}&\Phi : \mathbb{R}^{H\times W\times 3} \times \mathbb{R}^{H\times W\times 3} \times \mathbb{R}^{H\times W} \times \mathbb{R}^{H\times W} \rightarrow \mathbb{R}\\ &\Phi(I_1, I_2, H_q, H_p) \rightarrow w\end{aligned} \tag{3}$$

As input, this component takes the two RGB frames, the query heatmap and the predicted heatmap. Notice that only visible points will go through this.

This part of the pipeline is self-supervised, *i.e.* we do not use ground-truth weights for training. The model infers the confidence of the visible correspondences thanks to the end-to-end training.

However, we were unable to get any improvement from this component. During training, the model was unable to learn any useful weighting for the correspondences, predicting very similar values for all of them. See Section 5 for more details.

3.4. Weighted Procrustes aligner (Θ)

Given a set of query pixels Q', visible predicted matches M' and their confidence weights W, we back-project Q', M' to 3D points using the known camera intrinsics matrix. We define these sets of back-projected 3D points as X, Y respectively. We aim at finding the rotation matrix $R \in SO(3)$ and translation vector $t \in \mathbb{R}^3$ that minimize the following function:

$$\mathcal{L}(R, t, W, X, Y) = \sum_i w_i(y_i - (Rx_i + t))^2 \tag{4}$$

This problem is know as the Weighted Procrustes analysis.

As show by Choy *et al.* [5], the closed solution for R and t are

$$\begin{aligned}\hat{R} &= USV^T\\ \hat{t} &= (Y - \hat{R}X)diag(\tilde{W})\mathbf{1}\end{aligned} \tag{5}$$

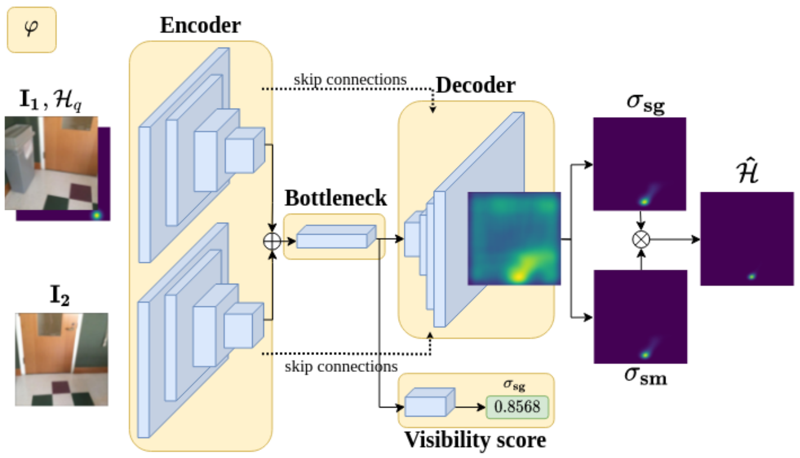


Figure 2. Architecture diagram of the *Correspondence and visualization prediction* component (φ). The component takes two RGB frames and a query heatmap $\mathcal{H}_q$ of dimensions $H \times W$ as input. The model predicts a heatmap $\hat{\mathcal{H}}$ encoding the likelihood of the coordinates of the matching point. We based this component's architecture on Božič *et al.* [4].

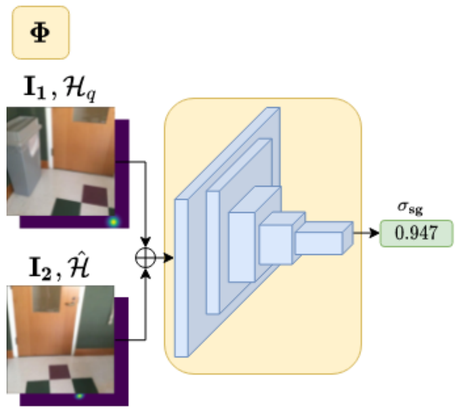

$\mathbf{\Phi}$
$\mathbf{I_1}, \mathcal{H}_q$
$\mathbf{I_2}, \hat{\mathcal{H}}$
σ_{sg}
0.947

Figure 3. Correspondence weighting component architecture. For the visible matches estimated by the *Correspondence and visualization prediction* component, this model will predict a vector of weights W such that inaccurate matches will have a smaller value than accurate matches.

where $\tilde{W}$ are the normalized weights, $U\Sigma V^T = SVD(\Sigma_{xy})$, $\Sigma_{xy} = YKWKX^T$, $K = I - \sqrt{\tilde{W}}\sqrt{\tilde{W}}^T$, and $S = diag(1, ..., 1, det(U), det(V))$.

Finally, we compute the following loss function to measure the error between the estimated parameters and the ground-truths:

$$\mathcal{L}_{align} = ||R_p^T R_{gt} - I|| + ||t_p - t_{gt}|| \tag{6}$$

The two terms measure the distance on $SE(3)$, but unlike Wang [15], we chose $L1$ to offer more robustness against outliers.

We combine the loss of all the components to enable end-to-end training: $\mathcal{L} = \mathcal{L}_{\mathcal{H}} + \lambda_v \mathcal{L}_v + \lambda_{align}\mathcal{L}_{align}$.

In all experiments we use the empirically determined weights $\lambda_v = 1$ and $\lambda_{align} = 1$.

4. Experiments

For these experiments we used ScanNet [6] which provides us with sequences of RGB-D frames with camera poses and camera parameters. We first trained the correspondence prediction, followed by visibility prediction, then correspondence weighting, and finally end-to-end training. However, these two steps were unable to learn anything.

Correspondence prediction. We generated a train split with 380 scenes, 80k pairs of frames and 3.5M of visible correspondences. The validation split included 90 scenes, 20k pairs of frames with 800k visible correspondences. Using this dataset, we trained the correspondence prediction component with an initial learning rate of 0.01. Every 30k iterations we decrease it by a factor of 0.1. The model's loss converged after 60k iterations.

Visibility prediction. While training these part, all parts except the visibility block were frozen. The train split contained 90 scenes, 20k pairs of frames with 2M pairs of correspondences (800k visible, 1.2M occluded). The validation split contained 10 scenes with 2k pairs and 220k correspondences (90k visible, 130k occluded). We used the same parameters as the previous section.

Correspondence weighting. While training this part, the component *correspondence and visibility prediction* (φ) was frozen. The train split contained 1201 scenes with 307k pairs of frames. The validation split contained 312 scenes with 80k pairs of correspondences. For this part, a batch contained only a single pair with a sparse set of query pixels. In this manner, the batch can be back-projected to 3D and feed into the weighted Procrustes aligner. We used the same parameters as in the previous section. As mentioned, this component was unable to learn proper weighting for the correspondences.

End-to-end training. We used the same splits and hyper-parameters as in the previous part. Again, the network was unable to improve the results.

4.1. Correspondence accuracy

In this section, we compere the correspondence prediction of our network against ORB [13] and SIFT [11]. In figure 5, we can see the distance (in pixels) between the ground-truth match and the predicted match. We can see how the precision of the correspondences predicted by our network is similar to the ones obtained by using ORB and SIFT.

We also tested our network with the 7-Scenes dataset [9]. We used the train and validation split provided by RelPoseNet [12]. These splits provide relative transformations much bigger than the ones we used from ScanNet, sometimes including frames without overlap. Figure 6 shows the distance (in pixels) between the ground-truth match and the predicted match. We can see how our network is able to predict matches with similar precision as when using ScanNet. This demonstrates that our model is able to generalize to unseen datasets. However, it can be seen that for big angles the error increases. This is due to the fact that the pairs in 7-Scenes dataset combine big translation with big rotation.

4.2. Visibility prediction

Our model achieves an accuracy of 0.83 on the validation dataset.

4.3. Comparison with ICP

In this section we compare our model against ICP using point-to-point (pt2pt), point-to-plane (pt2pl), and both (ICP). Figure 9 shows the Root Mean Squared Error (RMSE) per rotation and translation threshold. We can see how our method is able to achieve better results than ICP. We can conclude that our model is able to estimate a initial relative transformation which is closer to the one produced by ICP. Therefore, when ICP is executed using the network's prediction as initial estimate, it can avoid local minimum and get closer to the optimal solution.

Figure 10 shows the output of the back-projected frames aligned with the different methods. Notice that the overlap among the frames is small. We can see how the alignment of ICP fails to reconstruct the scene. Similarly, the network reconstruction only aligns the sparse predicted correspondences, however, when combining both, the reconstructed scene is very similar to the real one. Again, this confirms that the predicted pose is close to the global minimum, which helps ICP avoiding getting stuck in a local minimum.

5. Conclusions and Future Work

We presented an end-to-end learnable, differentiable method for pairwise relative pose registration. It is able outperforms ICP for 3D registration and it predicts a pose that can guide ICP to obtain a better solution. Our model is able to predict accurate correspondences on unseen scenarios proving that it's able to generalize.

However, for large rotations and translation, the accuracy of the predicted correspondences decreases considerably. We were not able to take advantage of the end-to-end training due to the *correspondence weighting* component not being able to predict meaningful scores. We believe that end-to-end training is crucial to improve the performance of this method. Currently, a poor match and wrongly classifying an occluded point as visible have a strong impact during the pose estimation. We believe end-to-end training will be able to improve the performance of the *correspondence and visibility prediction* in addition of decreasing the influence of possible outliers.

Finally, working with dense correspondences can provide the *Weighted Procrustes Aligner* component with more data. However, it is critical to accurately down-weighting wrong correspondences.

References

[1] Herbert Bay, Tinne Tuytelaars, and Luc Van Gool. Surf: Speeded up robust features. volume 3951, pages 404–417, 07 2006.

[2] PJ Besl and ND Mckay. A method for registration of 3-d shapes, ieee trans. *P flattern Anal. and M ac h ine I ntell*, 1(4):23, 1992.

[3] Aljaž Božič, Pablo Palafox, Michael Zollhöfer, Angela Dai, Justus Thies, and Matthias Nießner. Neural Non-Rigid Tracking. *arXiv:2006.13240 [cs, eess]*, Jan. 2021. arXiv: 2006.13240.

[4] Aljaž Božič, Michael Zollhöfer, Christian Theobalt, and Matthias Nießner. DeepDeform: Learning Non-rigid RGB-D Reconstruction with Semi-supervised Data. *arXiv:1912.04302 [cs]*, Mar. 2020. arXiv: 1912.04302.

[5] Christopher Choy, Wei Dong, and Vladlen Koltun. Deep Global Registration. *arXiv:2004.11540 [cs, eess]*, May 2020. arXiv: 2004.11540.

[6] Angela Dai, Angel X. Chang, Manolis Savva, Maciej Halber, Thomas Funkhouser, and Matthias Nießner. Scannet: Richly-annotated 3d reconstructions of indoor scenes. In *Proc. Computer Vision and Pattern Recognition (CVPR), IEEE*, 2017.

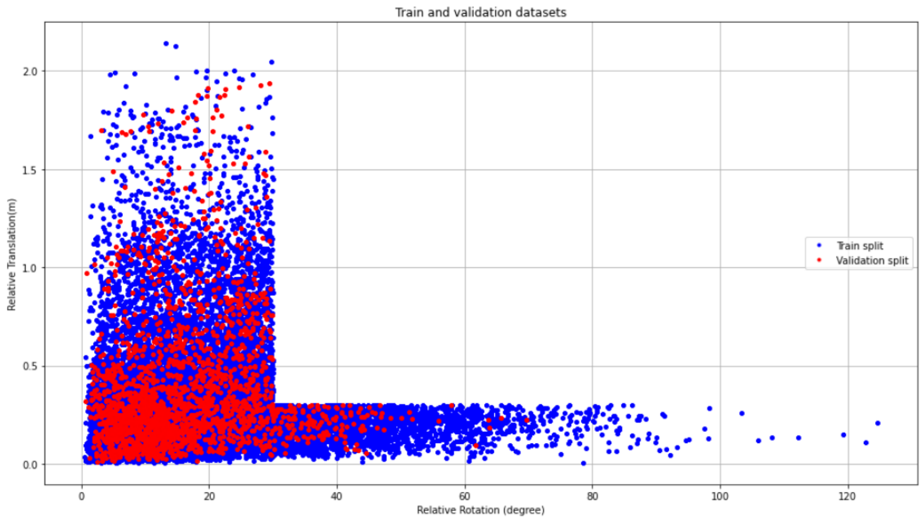
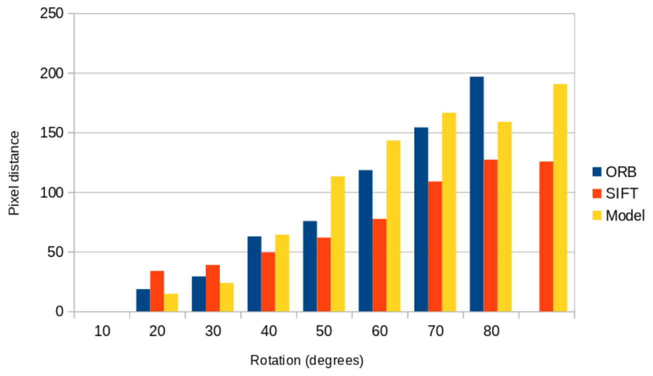


Figure 4. Visualization of the relative positions between the pairs used for training and validating this method. A dot indicates a pair of frames with a relative pose of a rotation of X degrees and translation of Y meters. It can be seen that we exclude any pair with rotation bigger than 30 degrees **and** translation bigger than 0.3 meters. This is to avoid pairs with small overlap. However, for previous experiments, pairs withing 80 degrees and 1 meter were used. This did not lead to an increase on the performance, although, we believe that using data with more relax thresholds could help on generalizing better to unseen datasets with big relative poses.



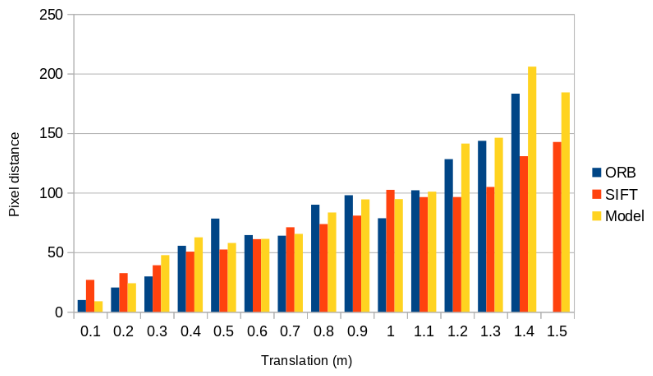


Figure 5. Plots showing the distance (in pixels) between the ground-truth correspondences and the predicted ones using the validation split from ScanNet [6]. It can be seen that the accuracy of the predicted correspondences is comparable to the one obtained with hand-crafted feature descriptors.

[7] Philipp Fischer, Alexey Dosovitskiy, and Thomas Brox. Descriptor matching with convolutional neural networks: a comparison to sift. *arXiv preprint arXiv:1405.5769*, 2014.

[8] 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*, 24(6):381–395, 1981.

[9] Ben Glocker, Shahram Izadi, Jamie Shotton, and Antonio Criminisi. Real-time rgb-d camera relocalization. In *2013 IEEE International Symposium on Mixed and Augmented Reality (ISMAR)*, pages 173–179. IEEE, 2013.

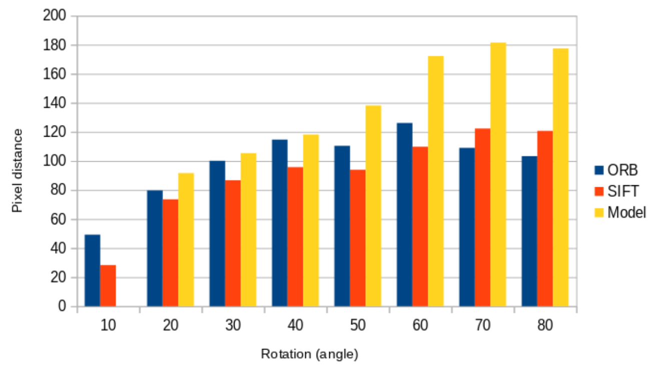
[10] Alex Kendall, Matthew Grimes, and Roberto Cipolla. Posenet: A convolutional network for real-time 6-dof camera relocalization, 2016.

[11] David G Lowe. Distinctive image features from scale-invariant keypoints. *International journal of computer vision*, 60(2):91–110, 2004.

[12] Iaroslav Melekhov, Juha Ylioinas, Juho Kannala, and Esa Rahtu. Relative camera pose estimation using convolutional neural networks, 2017.

[13] Ethan Rublee, Vincent Rabaud, Kurt Konolige, and Gary Bradski. Orb: An efficient alternative to sift or surf. In *2011 International Conference on Computer Vision*, pages 2564–2571, 2011.

[14] Torsten Sattler, Qunjie Zhou, Marc Pollefeys, and Laura Leal-Taixe. Understanding the Limitations of CNN-based Absolute Camera Pose Regression. *arXiv:1903.07504 [cs]*, Mar. 2019. arXiv: 1903.07504.



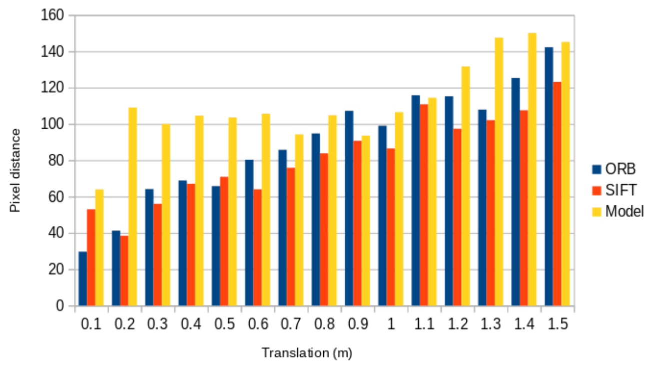


Figure 6. Plots showing the distance (in pixels) between the ground-truth correspondence and the predicted ones using the unseen 7-Scenes dataset [9]. We can see how the network is able to generalize to unseen scenarios. Due to the fact that this dataset contains bigger transformations than the ones used during training, the predictions for higher rotations are less accurate.

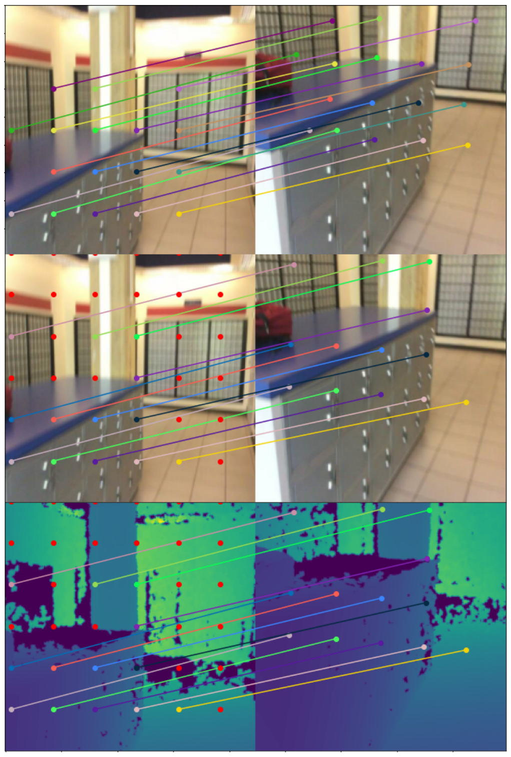


Figure 7. Predicted correspondences from ScanNet's validation split. The top pair shows the predicted correspondences, the middle pair ground-truth matches and visibility scores. A red dot indicates that the matching pixel is not visible. In some cases, non-Lambertian textures result in sensor noise, leading to incorrect ground-truth visibilities. Notice how the model is able to deal with repetitive textures such as the mail boxes.

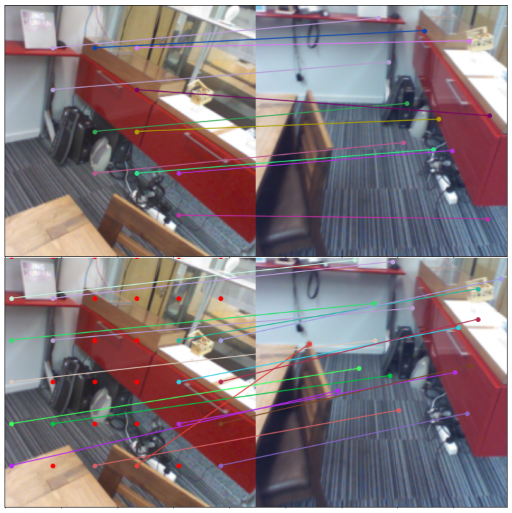
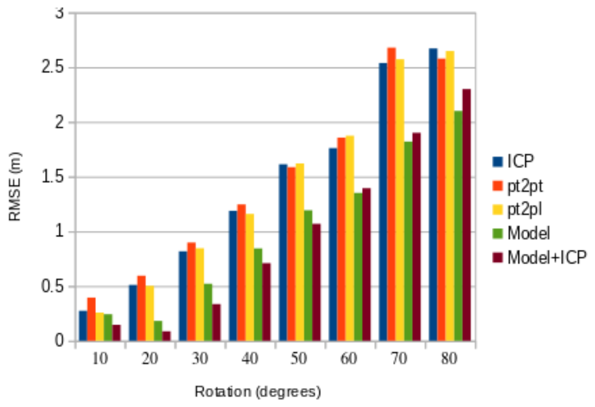


Figure 8. Predicted correspondences using 7-Scenes dataset [9]. This dataset was unseen by the network. On the top pair we can see the predicted matches and in the bottom the ground-truths. The red dots indicate that they are not visible. Due to sensor noise, some of the ground-truth visibilities are incorrect.

[15] Yue Wang and Justin M. Solomon. Deep Closest Point: Learning Representations for Point Cloud Registration. *arXiv:1905.03304 [cs]*, May 2019. arXiv: 1905.03304.

[16] Chenhao Yang, Yuyi Liu, and Andreas Zell. Rcpnet: Deep-learning based relative camera pose estimation for uavs. In *2020 International Conference on Unmanned Aircraft Systems (ICUAS)*, pages 1085–1092, 2020.

[17] Michael Zollhöfer, Andreas Görlitz Patrick Stotko, Christian Theobalt, Matthias Nießner, Reinhard Klein, and Andreas Kolb. State of the Art on 3D Reconstruction with RGB-D Cameras. 2018. Publisher: EG.



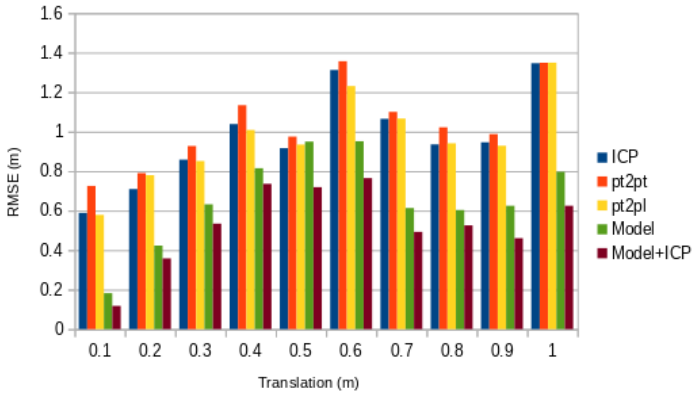


Figure 9. Comparison of reconstruction using IPC and the model. For ICP, we used point-to-point constraints (pt2pt), point-to-plane constraints (pt2pl) and both (ICP). It can be seen how the network performs better than all ICP variants. In general, the combination *Model+ICP* performs better than *Model*. This is because the predicted pose is closer to the global minimum and ICP can estimate a better pose. However, if the predicted pose is far from the global minimum, ICP will estimate a pose further away leading to a higher error.

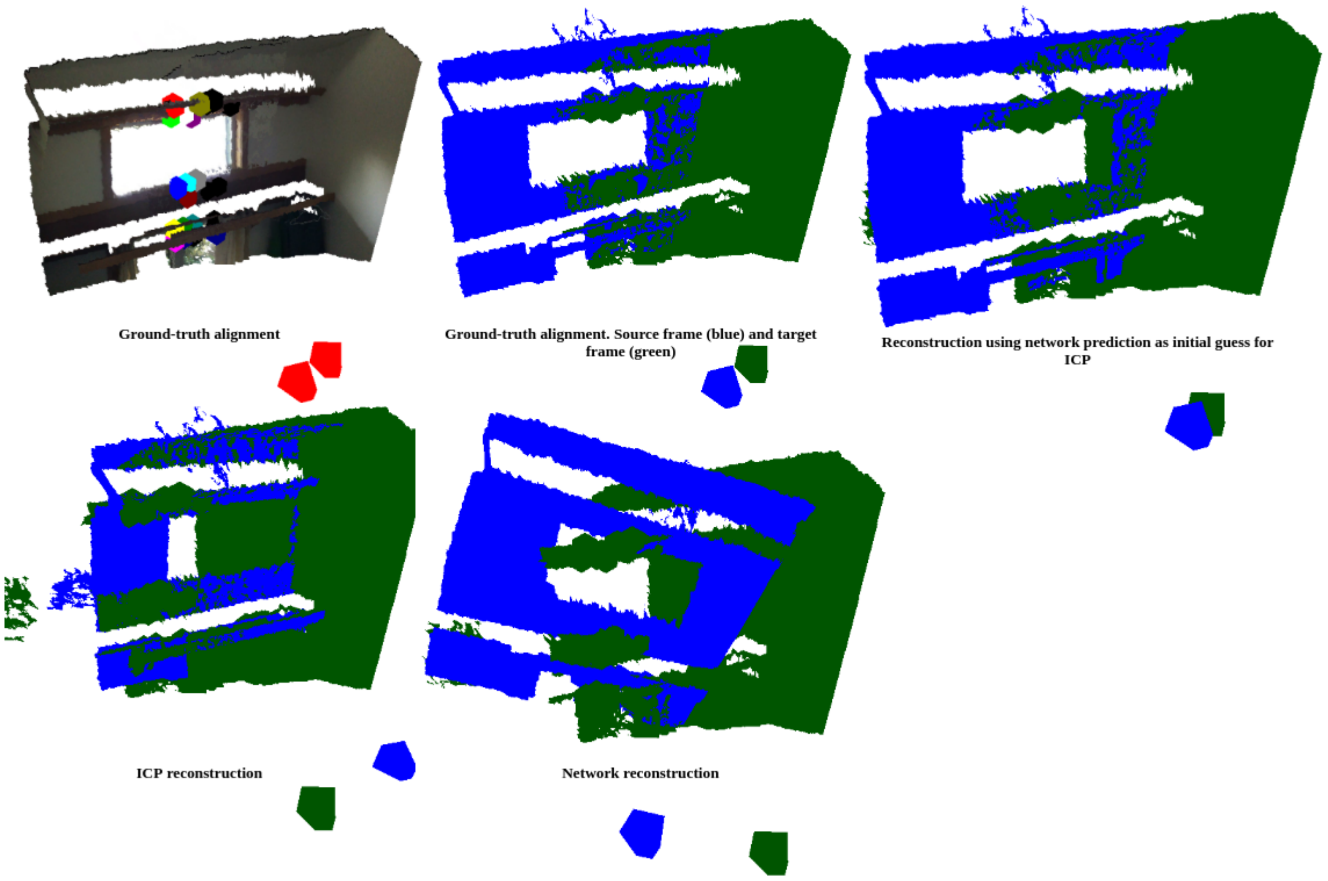

Ground-truth alignment
Ground-truth alignment. Source frame (blue) and target frame (green)
Reconstruction using network prediction as initial guess for ICP
ICP reconstruction
Network reconstruction

Figure 10. Example of 3D reconstruction. The frames used are from ScanNet [6] scene 0712 ids 2310 and 2360. For better visualization, we show in green the target frame and in blue the source frame. Notice in the ground-truth alignment that the overlap between the frames is small. We can see how ICP fails at reconstructing the scene and that the network is able to align the correspondences. Finally, when using the network's prediction to guide ICP, the obtained solution is much closer to the global minimum.