DEPARTMENT OF INFORMATICS

TECHNISCHE UNIVERSITÄT MÜNCHEN

Master's Thesis in Informatics

Learning Robust Correspondences Estimation

Marc Benedí San Millán

DEPARTMENT OF INFORMATICS

TECHNISCHE UNIVERSITÄT MÜNCHEN

Master's Thesis in Informatics

Learning Robust Correspondences Estimation

Robuste Korrespondenzschätzung Lernen

Author:	Marc Benedí San Millán
Supervisor:	Prof. Dr.-Ing. Matthias Nießner
Advisor:	Prof. Dr.-Ing. Matthias Nießner
Submission Date:	15.09.2022

I confirm that this master's thesis in informatics is my own work and I have documented all sources and material used.

Munich, 15.09.2022 Marc Benedí San Millán

Acknowledgments

I would like to thank my supervisor and advisor Matthias Nießner for his time and support throughout the duration of this project. His knowledge and experience enabled me to not deviate from the final goal and to stay motivated during all these months. I am also thankful for the multiple opportunities he provided during my studies which allowed me to get involved with the Visual Computing Laboratory.

I would also like to extend my gratitude to my fellow students at the laboratory for the many discussions and technical advice and to Kerem Yıldırır for the fruitful coffee breaks.

Most importantly, I am grateful to my family and friends for all the support and encouragement during this journey.

Danke.
Thank you.
Gràcies.

Marc Benedí San Millán
Munich, September of 2022

Abstract

Relative camera pose estimation is a fundamental problem in Computer Vision. While it has been widely studied over the decades by classical approaches and more recently by learning-based methods, it still remains a challenging problem. Particularly, wide-baseline camera transformation in indoor scenarios results in frames with small overlap and few correspondences.

In this Master's Thesis, we present LRCE, an end-to-end learnable differentiable approach for solving pairwise relative camera pose estimation for RGB-D frames. The key idea is to learn confidence scores for the correspondences in a self-supervised manner using an end-to-end loss based on the distance between the predicted and the ground truth poses. Additionally, our end-to-end differentiable pipeline enables refining correspondence prediction from the signal provided by the final alignment.

Finally, we evaluate our method on the ScanNet dataset by analyzing the performance of correspondence matching and pose estimation. We demonstrate that our approach is able to improve the pose alignment by weighting the correspondences with the predicted scores. We also show that the signal from the pose alignment improves the performance of the correspondence matching.

Contents

1. Introduction

Camera pose estimation is a core problem in many computer vision applications such as 3D reconstruction, Structure from Motion (SfM) or Simultaneous Localization and Mapping (SLAM). At its core definition, the problem consists in finding the camera location from a frame with respect to an arbitrary coordinate system such as another camera from a different frame or a global coordinate system.

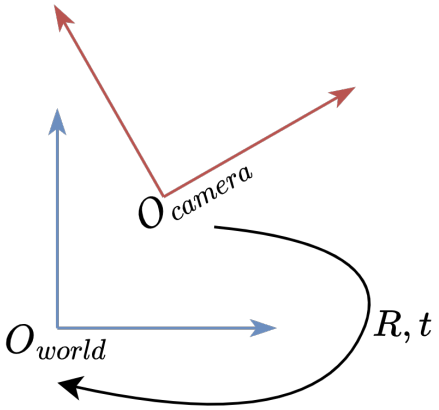

Figure 1.1.: Diagram of the camera pose estimation problem: The goal is finding the transformation between the camera O_{camera} coordinate system to another one, in this example, the world coordinate system O_{world}. The transformation has two components: rotation R and translation t.

Because relative pose estimation is such a core problem in Computer Vision, it has been studied for many decades. For calibrated cameras, there exist multiple classical approaches which solve it by constructing the Essential matrix. The Essential matrix relates correspondences between a pair of images through epipolar geometry. Likewise, it's possible to solve the problem for uncalibrated cameras using the Fundamental matrix.

Traditionally, pose estimation is approached in multiple steps [1]: i) feature detection, ii) feature description, iii) feature matching, and iv) pose optimization. The first step, feature detection, consists in selecting points of interest from both images, such as corners, and avoid selecting points on texture-less areas such as walls. Feature descriptors are then computed for each point using information from its neighborhood. Then,

the points from the first image are matched with points from the second image using their descriptors. A common approach for that is matching points which are nearest neighbors in the descriptor space. Finally, the matches are used as constraints for the optimization problem of finding a rigid transformation which minimizes the distance between the matches. Additionally, it might be necessary to add a post-processing step to remove outliers, that is, correspondences which are not correct.

In recent years, Deep Learning has been used to revisit many Computer Vision problems. The rapid evolution of hardware allows solving problems with approaches which were computationally unfeasible in the past. This is not only due to the increase in hardware's performance but also to the growth of available data. Moreover, every year sensors used to generate datasets are more accurate which is a crucial requirement for Deep Learning approaches.

Classic feature detectors and descriptors such as SIFT [2] and ORB [3] rely on hand-crafted detectors and descriptors which fail in wide-baseline scenarios, texture-less areas and other challenging situations. Incorrect matches have a fatal effect on pose optimization, therefore, many methods rely on post-processing the matches to filter the outliers. To do so, matches are filtered by the distance of the keypoints in feature space, or by iterative approaches such as RANSAC [4]. However, these methods are slow and perform poorly when there are too many outliers.

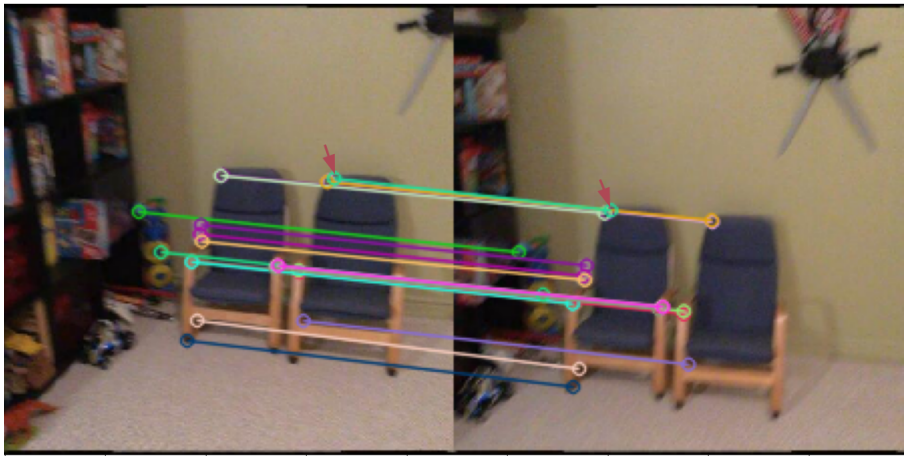

Figure 1.2.: Visualization of correspondences produced by SIFT. Correspondences are pairs of coordinates from each image which, ideally, back-project to the same point in space. The red arrows illustrate an example of an outlier, a wrong match, which have a strong impact during pose estimation.

Learning-based approaches tackle this problem by directly regressing the relative pose from a pair of frames, or by learning one (or more) steps from the classic pipeline: feature detection, feature description, feature matching, outlier removal and pose estimation. Nonetheless, methods that approach these steps do so in a decoupled manner. This

results in the steps not being able to provide feedback to each other, and therefore, not being able to learn from the final pose alignment.

To this end, we propose LRCE, a deep learning architecture to estimate the relative camera transformation between a pair of RGB-D frames. The key idea to our method is an end-to-end differentiable trainable pipeline which allows all components to be informed by the final alignment of the predicted pose.

A differentiable weighted Procrustes aligner allows us to obtain gradients from the final pose objective which is used to learn to weight the correspondences in a self-supervised manner. Therefore, our method solves the pose optimization problem in a robust manner.

Our model is formed by the following three components: i) Correspondence and Visibility Prediction, ii) Correspondence Weighting, and iii) Differentiable Weighted Procrustes.

Given a pair of RGB-D images, the model predicts the relative camera transformation between the *source* frame and the *target* frame. First, the Correspondence Prediction component computes 2D sparse matches and visibility scores between *source* and *target* only using RGB channels. The visibility score allows the model to filter out occluded points from the *source* frame which don't have a match in the *target* frame. Afterwards, the network uses the 2D correspondences to retrieve 3D matches using the depth frames and predicts a confidence score for each weight. Finally, the 3D matches and confidence scores are fed into a differentiable weighted Procrustes which optimizes the relative pose.

We balance the use of supervised and self-supervised training for our model. First, Correspondence Prediction and Visibility prediction are trained in a supervised fashion using ground truth correspondences and visibility labels, which are computed from the dataset's frames and camera poses. Then, we train relative pose estimation with the weights in an end-to-end self-supervised manner.This end-to-end loss also informs the Correspondence and Visibility Prediction module improving its performance.

We evaluate our method in ScanNet [5] dataset. We show how feature matching results improve when using the proposed weighting scheme. Similarly, we evaluate pose estimation by comparing against the baselines and showing how the performance of our method improves when using the self-supervised confidence scores as weights.

In summary, the main contributions of this thesis are the following:

- a correspondence and visibility prediction module for pairs of RGB frames
- a self-supervised module that learns correspondence weighting enabling robust outlier rejection
- an end-to-end differentiable pipeline which enables all the modules to be optimized with final pose objective

The source code is available[1].

The rest of the thesis is structured as follows:

Chapter 2 reviews existing approaches which tackle feature detection, feature matching, outlier filtering and the problem of pose estimation.

Chapter 3 presents the proposed method, LRCE, a CNN architecture with three main components: Correspondence and Visibility prediction, Correspondence weighting and a Differentiable Weighted Procrustes aligner.

Chapter 4 starts by reviewing the implementation and training details. Then it evaluates and discusses the results for the tasks of matching, correspondence weighting and pose optimization. Finally, ablation studies show the performance of the method with different configurations.

Chapter 5 discusses the limitations of our approach and presents possible improvements and further work.

Chapter 6 concludes this thesis by restating the key ideas and main contributions.

[1] `https://github.com/marcbenedi/LRCE`

2. Related Work

In this chapter, we review different research efforts for the tasks related to the pose estimation problem We start in Section 2.1 discussing classic feature matching and continue in Section 2.2 with learning-based feature matching. In Section 2.3, we examine the literature for outlier filtering which enables a more robust optimization of the pose, and we end in Section 2.4 discussing methods which tackle the pose estimation problem.

2.1. Classic Feature Matching

Feature matching is classically approached as a multi-step pipeline: i) keypoint detection, ii) feature description, iii) feature matching, and iv) outlier removal.

Classical methods solve these steps using hand-crafted geometry descriptors. Harris Corner Detection [6] is a corner detection algorithm based on the difference in intensity in all directions for a neighbourhood. However, corners look different in different images. To tackle this issue, SIFT [2] introduces a scale-invariant corner detector which uses nearest-neighbour search to create matches. Other hand-crafter methods are SURF [7], FAST [8] and ORB [3] which are widely used in many tasks because of their good performance.

In spite of their popularity, hand-crafted methods strive with large viewpoint/illumination changes, texture-less areas and repetitive patterns.

2.2. Learning-based Feature Matching

In the last decade, the increasing availability of reliable sensors enabled the construction of datasets such as [5, 9]. These datasets allowed Deep Learning to revisit feature matching as a learning-based problem.

LIFT [10] is one of the first successful learning-based feature description methods. They solve detection, orientation estimation, and feature description in a unified manner using an end-to-end approach. Other approaches construct a 4D cost volume, which stores the cost of associating a pixel with its corresponding pixels. However, 4D cost volumes are computationally expensive to construct and process due to their high dimensionality caused by storing the cost of associating all keypoints on both frames. [11, 12, 13] explore patterns in the cost volume to find matches using sparse convolutions. PWC-Net [14]

finds matches across images by predicting optical flow. That is, they regress a 2D offset vectors per pixel.

Some other methods like SuperGlue [15] learn to match two sets of local features provided by an independent component using a Graph Neural Network and the attention mechanism. This combination allows them to leverage the relationship of keypoints in the same image and to the other image. Then, they construct a score matrix by computing the inner product of the features and find the matches as a partial assignment problem on the score matrix using the Sinkhorn algorithm [16].

Other works based on SuperGlue such as Roessle [17] take advantage of the situations where this problem arises, such as in video sequences, and use multiple-view to further improve the performance. In addition, they add an end-to-end loss which allows the network to learn from the final alignment.

However, these methods depend on an independent detector component which limits their performance to the selected kepoints. Additionally, it means that feature detection cannot benefit from an end-to-end loss. In contrast, our proposed method is end-to-end differentiable and all components benefit from the final pose alignment.

A similar method is LoFTR [18], which presents a detector-free matching approach that performs semi-dense pixel matching in a coarse-to-fine manner also using attention. However, their method does not get any signal from the final pose alignment.

2.3. Outlier Filtering

Despite the improvements in the performance of feature detection and matching, outliers caused by erroneous correspondences have a strong negative impact during pose optimization. In addition, sensor data, particularly depth frames, are prone to contain noise due to sensor accuracy, working range or environment conditions such as mirrors or glass.

A common approach is performing outlier removal after feature matching in an attempt to remove the outliers from the problem constraints. A widely used algorithm for such a task is Random Sample Consensus (RANSAC) [4], which aims to iteratively find solutions for randomly selected subsets of correspondences, estimate their alignment, and then select the solution with minimum error. However, RANSAC is slow because of its iterative sampling approach and suffers when the signal-to-noise ratio decreases.

Methods which learn the matching function, such as SuperGlue, filter outliers by matching them to a "dustbin". This dustbin is added by extending the score matrix with a row and a column.

Several methods leverage learning a weighting scheme for the correspondences. Deep Global Registration [19] tackles the problem of point cloud registration using a CNN

architecture. Their method directly finds correspondences in the 3D data whereas we find the matches in the 2D image domain which then we back-project to 3D using the depth data. They follow a similar approach to perform robust optimization by predicting confidence weights for each match. In addition, both approaches train correspondence weighting in a self-supervised manner using the final pose alignment as a loss function.

Other approaches learn to filter outliers as a binary classification problem. Yi, Trulls, Ono, et al. [10] present an approach to classify correspondences. Their method is supervised by exploiting the epipolar geometry. This allows them not having to label the correspondences explicitly.

2.4. Camera Pose Estimation

Pose estimation is the last step in the pose optimization pipeline. It is a very well-studied method in Computer Vision because it is a key component in many applications such as 3D reconstruction, SfM or SLAM. The 8-point algorithm [20] and its derivative works can be used to recover the Essential matrix $\mathbf{E}$ which relates correspondences between two images. From the Essential matrix, it is possible to recover 5 degrees of freedom from the relative transformation between the two cameras, since it is not possible to recover the scale of the translation component. Similarly, if the camera parameters are unknown, we can recover the Fundamental matrix $\mathbf{F}$. For a full in-depth explanation, we refer the reader to [21].

The methods mentioned above only use 2D data from images and therefore have limitations such as scale ambiguity. Another field takes advantage of 3D information and approaches the pose estimation problem as a point cloud registration, that is, finding the transformation between two sets of points. Iterative Closest Points (ICP) [22], tackles the problem with an iterative approach with two steps: i) estimate correspondences between the two sets of points, ii) find a rigid transformation which minimizes an objective function. However, ICP only converges to a good alignment if the starting positions are close enough.

Kendall, Grimes, and Cipolla [23] introduce a method for absolute pose regression. Their method uses convolutional layers to extract features from RGB images and then a fully connected layers to regress the 6 degrees of freedom of the pose. In addition, they leverage transfer learning from a network trained for a classification task. However, although both relative pose and absolute pose problems estimate a 6-DOF transformation they are inherently different since the former approach can generalize to unseen scenes whereas the latter one is trained per scene [24].

Other learning-based approaches directly regress relative poses from RGB images. Melekhov, Ylioinas, Kannala, and Rahtu [25] present a Siamese network architecture adopting the same learning objective as [23] but for relative camera pose.

Recent end-to-end approaches integrate feature matching with pose optimization. Roessle and Nießner [17] present an approach for feature matching using differentiable pose optimization. Similar to our method, their pipeline is end-to-end trainable and they predict confidence scores which are trained in a self-supervised manner from the pose objective. However, in contrast to our work, their method depends on a given set of feature descriptors, which means that feature description and the rest of the pipeline remain disconnected and cannot inform each other.

3. Method

In this chapter, we introduce LRCE, our proposed network architecture for solving the pose estimation problem. We begin with an overview of the model (Section 3.1) followed by a detailed explanation of all its components: *Correspondence and Visibility Estimation* (φ) (Section 3.2), *Correspondence Weighting* (ω) (Section 3.3) and *Differentiable Weighted Procrustes* (Section 3.4). Finally, we describe the loss used to train our components in an end-to-end fashion (Section 3.5). We refer the reader to Section 4.1 for implementation and training details.

We define the relative pose estimation problem formally as: given a pair of frames, source S and target T, find the euclidean transformation $\mathcal{T}_{TS}$ which aligns the source frame to the target frame. The transformation $\mathcal{T}_{TS}$ is defined as $\mathcal{T}_{TS}(\mathbf{p}) = R\mathbf{p} + \mathbf{t}$, where $\mathbf{p}$ belongs to S and $\mathcal{T}_{TS}(\mathbf{p})$ belongs to T. R is an orthogonal transformation, i. e. it has the following properties: $R \in SO(3), R \in \mathbb{R}^{3\times3}, R^TR = I,\ \det(R) = 1$. These force R to be a rotation matrix, i. e. without reflections. $\mathbf{t} \in \mathbb{R}^3$ is a translation vector.

Each frame has the following information associated: RGB image $\mathcal{C} \in \mathbb{R}^{H\times W\times3}$, depth image $\mathcal{D} \in \mathbb{R}^{H\times W}$ and the camera pose $\mathcal{P} \in \mathbb{R}^{4\times4}$. Additionally, we back-project the values of $\mathcal{D}$ and obtain the associated point cloud $\mathcal{B} \in \mathbb{R}^{H\times W\times3}$.

We also have known camera parameters $\mathcal{K} \in \mathbb{R}^{4\times4}$. Figure 3.1 shows a diagram of the problem and all its components.

3.1. Overview

Given two frames, S and T, our method computes the relative pose between the two cameras following these steps:

1. Select $Q := \{q_1, \ldots, q_n \,|q_i \in \mathbb{R}^2\}$, a set of n query pixels from $\mathcal{C}_S$.
2. Using the *Correspondence Estimation* module (φ), predict a set of correspondences $M := \{m_1, \ldots, m_n | m_i \in \mathbb{R}^2\}$. Ideally, m_i are the coordinates of q_i in $\mathcal{C}_T$.
3. Additionally, we predict a set of visibility scores $V := \{v_1, \ldots, v_n | v_i \in (0, 1)\}$, where v_i indicates if q_i is visible in $\mathcal{C}_T$.
4. Next, we recover the 3D points associated with Q and M from $\mathcal{B}_S$ and $\mathcal{B}_T$ respectively. We denote these sets of back-projected pixels as $\mathbf{Q} := \{\mathbf{q}_1, \ldots, \mathbf{q}_n | \mathbf{q}_i := \mathcal{B}_S[q_i] \in \mathbb{R}^3\}$ and $\mathbf{M} := \{\mathbf{m}_1, \ldots, \mathbf{m}_n | \mathbf{m}_i := \mathcal{B}_T[m_i] \in \mathbb{R}^3\}$.

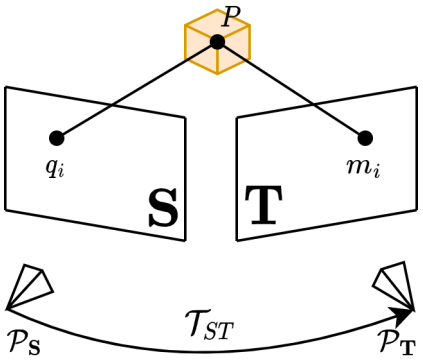

Figure 3.1.: Diagram of the pose estimation problem between two frames: S and T. The transformation $\mathcal{T}_{TS}$ transforms points from S to T. The tuple (q_i, m_i) is a valid match because their back-projection to the 3D world is the same, the point P.

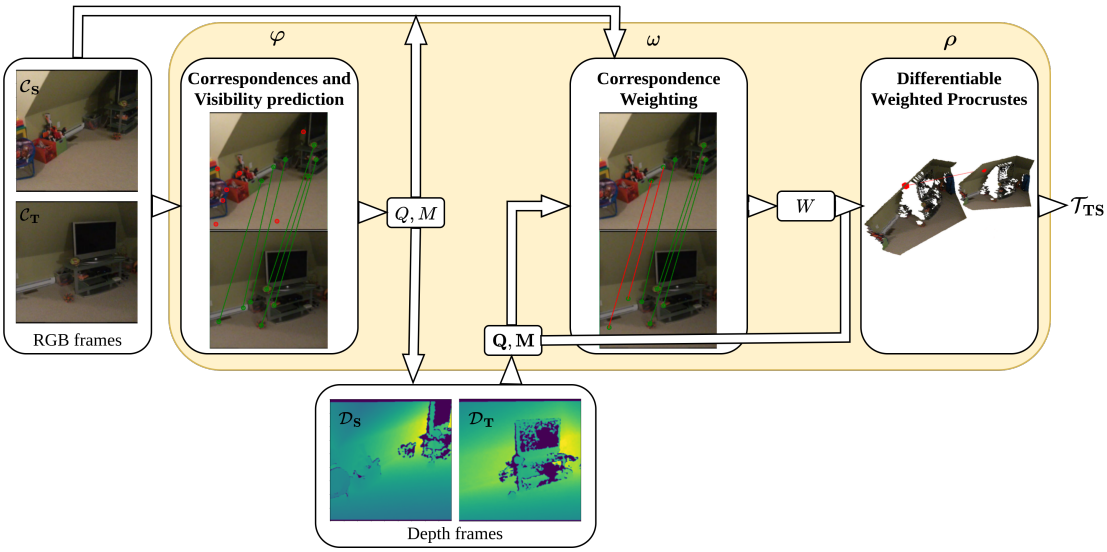

Figure 3.2.: LRCE: based on a pair of RGB-D frames: S and T, our method estimates the relative camera transformation between the frames. To this end, we propose an end-to-end CNN-based architecture which solves the problem in 3 steps: i) Correspondence and visibility prediction (φ), ii) Correspondence weighting (ω), iii) Differentiable Weighted Procrustes (ρ). Our end-to-end approach enables all learnable steps to be informed by the final pose estimation objective.

5. To perform a robust optimization of the relative pose, the *Correspondence Weighting* module (ω) outputs a set of confidence scores $W := \{w_1, \ldots, w_n | w_i \in (0,1)\}$. As input, it takes the values of $\mathcal{C}_S$ at pixels Q, the values of $\mathcal{C}_T$ at coordinates M, the values of $\mathcal{B}_S$ at coordinates Q, and $\mathcal{B}_T$ at coordinates M.

6. Finally, the *Differentiable Weighted Procrustes* (ρ) regresses the pose $\mathcal{T}_{TS}$ which aligns $\mathbf{Q}$ and $\mathbf{M}$ using W to weight the matches during optimization.

We refer the reader to Figure 3.2 for a diagram of our pipeline.

3.2. Correspondence and Visibility Estimation (φ)

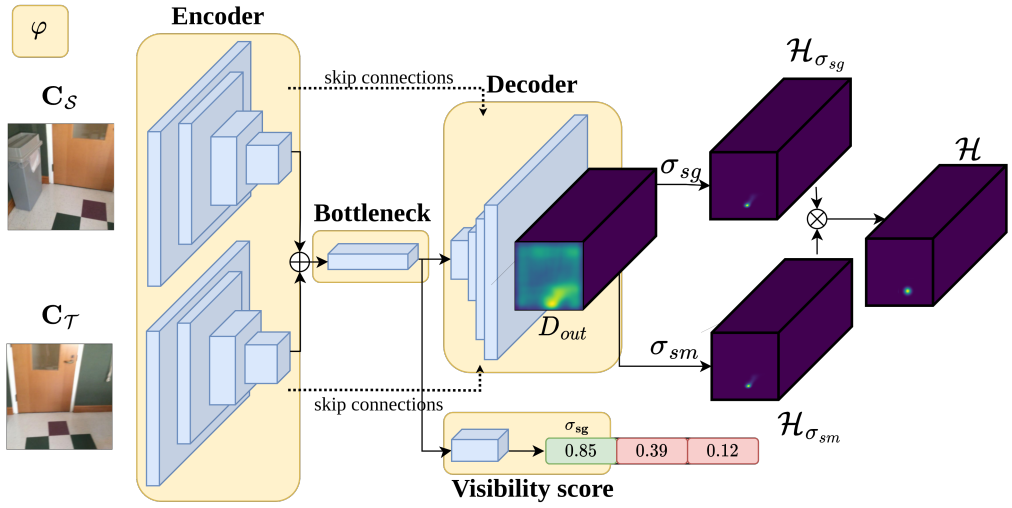

Figure 3.3.: Architecture of the *Correspondence and Visibility prediction* component (φ). Input to this component are two RGB frames, $\mathcal{C}_S$ and $\mathcal{C}_T$, from which n heat maps $\mathcal{H}_i$ of dimensions $h \times w$ are predicted.

The first step to solve the pose estimation problem is finding a set of pixels in both images which represent the same point in the 3D world. See Figure 3.1 for a diagram of the described scenario. Motivated by the work of [26], we propose a CNN architecture based on a Siamese network with two towers and shared parameters. As input, this component takes two RGB images, $\mathcal{C}_S$ and $\mathcal{C}_T$, of dimensions $H \times W \times 3$. As output, the method estimates a set M of 2D pixel coordinates for a set of query pixels Q and their visibility scores V. The set of query pixels is always the same for all pairs of images and it depends on the hyper-parameter n which can be changed to accommodate different memory requirements. We select the query pixels such that they are evenly distributed across the image. Naturally, the method predicts dense correspondences when $n = H \cdot W$.

$$\begin{aligned} &\varphi : \mathbb{R}^{H\times W\times 3} \times \mathbb{R}^{H\times W\times 3} \to \mathbb{R}^{n\times 2} \times \mathbb{R}^{n}, \\ &\varphi\left(\mathcal{C}_S, \mathcal{C}_T\right) \to (M, V) \end{aligned} \tag{3.1}$$

Heat map The goal is to predict a set of heat maps $\mathcal{H} := \left\{\mathcal{H}_i, \ldots, \mathcal{H}_n | \mathcal{H}_i \in [0,1]^{h\times w}\right\}$. Each heat map $\mathcal{H}_i$ encodes the likelihood of the location of q_i in $\mathcal{C}_T$ but at a resolution of $h \times w$ instead of $H \times W$. For more details, we refer the reader to Section 4.1. Let D_{out} be the output of the last layer of the decoder. We compute two heat map volumes $\mathcal{H}_{sg}$ and $\mathcal{H}_{sm}$:

$$\begin{aligned} \mathcal{H}_{sg} &= \sigma_{sg}\left(D_{out}\right), \\ \mathcal{H}_{sm} &= \sigma_{sm}\left(D_{out}\right) \end{aligned} \tag{3.2}$$

where σ_{sg} and σ_{sm} indicate the *sigmoid* and *softmax* activation functions respectively. Note that σ_{sm} is applied channel-wise, i. e. it is equivalent to applying σ_{sm} to each $\mathcal{H}_i$ independently. $\mathcal{H}_{sg}$ maps all values to the range $[0,1]$. This is equivalent to independent binary classification problems per pixel. $\mathcal{H}_{sm}$ ensures that $\sum_j \mathcal{H}_i(j) = 1$, therefore, each $\mathcal{H}_i$ makes a complete probability distribution. Finally, $\mathcal{H}$ is computed as $\mathcal{H} = \mathcal{H}_{sg} \otimes \mathcal{H}_{sm}$, where $\otimes$ denotes the Hadamard product.

From heat maps to 2D-coordinates To retrieve the set of 2D-coordinates M from $\mathcal{H}$, it is necessary to locate the coordinates of the pixel with maximum value. The *argmax* operator can be used to obtain such coordinates, however, it is non-differentiable. An alternative is using the *soft-argmax* operator which is differentiable and enables the gradients to flow from the end-to-end loss. In short, the *soft-argmax* of a tensor T is defined as $\sum_i \sigma_{sm}(T)_i \odot G_i$, where G is a tensor of the same dimensions as T where each element contains its coordinates. In other words, this operator is a weighted sum of the coordinates in G. In our case, each heat map $\mathcal{H}_i$ is already normalized (sums up to 1), therefore, we just multiply it element-wise with a tensor of dimensions $h \times w \times 2$ and sum the result. For each heat map $\mathcal{H}_i$ we obtain m_i, therefore, $\mathcal{H}$ results in M. We then scale the values from the range $(0,0) - (h,w)$ to $(0,0) - (H,W)$. Note that the values of m_i are continues, thus achieving sub-pixel accuracy.

Finally, we apply bilinear interpolation to $\mathcal{B}_T$ at each m_i obtaining $\mathbf{M}$.

Visibility As previously seen, the heat maps express the likelihood of a pixel being a match, therefore, they sum up to 1. This means that the *Correspondence Estimation* component cannot produce an empty heat map and express whereas a query pixel is visible in $\mathcal{C}_T$. This can happen if the match is outside the image boundaries or occluded by some object. Therefore, this component learns to predict if the query pixel q_i is visible or not in $\mathcal{C}_T$.

This module takes as input the output of the bottleneck B and outputs a 1-dimensional vector $V := \{v_1, \ldots, v_n\}$. We apply the sigmoid operator to the set so that all $v_i \in [0, 1]$.

$$\begin{aligned} &\varphi : \mathbb{R}^{H \times W \times 3} \times \mathbb{R}^{H \times W \times 3} \rightarrow \mathbb{R}^{h \times w \times n}, \\ &\varphi\left(\mathcal{C}_S, \mathcal{C}_T\right) \rightarrow V \end{aligned} \tag{3.3}$$

Loss

During optimization, we supervise the heat maps using a similar loss to [26] but extended to optimize a volume with n heat maps instead of a single one:

$$\begin{aligned} \mathcal{L}_{\mathcal{H}} = &\sum_i V_{gt} \Phi_{bce}(w_{\mathcal{H}}(\mathcal{H}_{sg} - \mathcal{H}_{\text{gt}})) + \\ &\lambda_{\text{nll}} \sum_i V_{gt} \Phi_{nll}(w_{\mathcal{H}}(\mathcal{H}_{sm} - \mathcal{H}_{\text{gt}})) \end{aligned} \tag{3.4}$$

where Φ_{bce} denotes binary-cross entropy loss, Φ_{nll} denotes negative-log-likelihood loss, $\mathcal{H}_{gt}$ denotes the ground-truth heat maps and $w_{\mathcal{H}}$ is a weight tensor which assigns higher weights to the elements closer to the ground-truth match pixel coordinates. We use the ground truth visibility labels to mask out occluded correspondences and $\lambda_{\text{nll}} = 10$.

$\mathcal{H}_{gt_i}$ are generated by creating a zero tensor of dimensions $h \times w$ and setting the value of at m_{gt_i} to 1. To prevent the network from predicting only zero values, we apply a Gaussian kernel around m_{gt_i} with standard deviation of 1. This decays the neighboring pixel values to zero. In a similar manner, $w_{\mathcal{H}}$ are defined as $1 + 10\mathcal{H}_{gt}$.

We supervise visibility as a binary classification problem:

$$\mathcal{L}_V = \sum_i \Phi_{bce}\left(V - V_{gt}\right) \tag{3.5}$$

The ground-truth visibility labels are generated using the ground-truth relative poses and the depth frames. For more details about how the ground truth labels are obtained see Section 4.2.

3.3. Correspondence Weighting (ω)

For each match $(\mathbf{q}_i, \mathbf{m}_i)$, we additionally predict a weight $w_i \in (0, 1)$ to diminish the effect of outliers and therefore enabling a more robust optimization.

The weighting function ω takes as input two tensors of dimensions $h \times w \times 6$. These two tensors are constructed by appending the RGB values of the query/match pixels and the associated 3D values from the back-projected depth frame. In other words: $(\mathcal{C}_S[q_i] \parallel \mathbf{q}_i)$ and $(\mathcal{C}_T[m_i] \parallel \mathbf{m}_i)$.

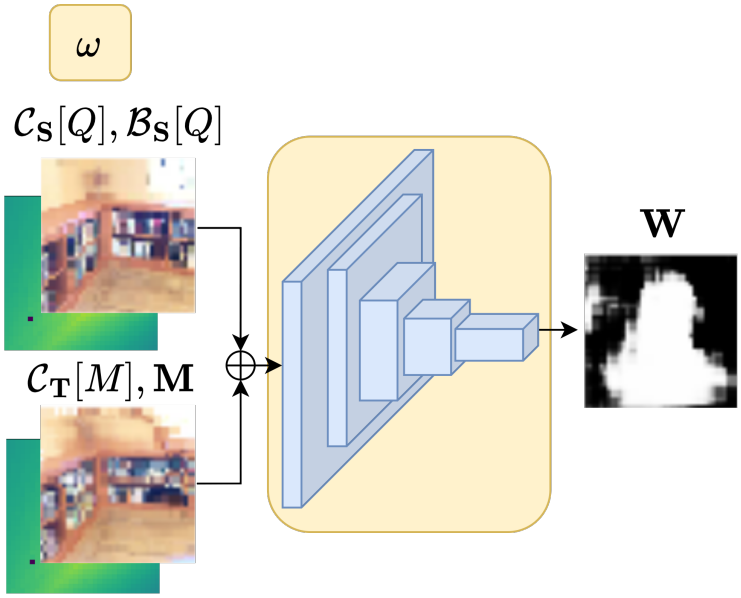

Figure 3.4.: Architecture of the *Correspondence Weighting* component (ω). Input to this component are the color and back-projected point of S, T at Q, M respectively. It predicts a confidence score per match.

$$\begin{aligned} &\omega : \mathbb{R}^{h\times w\times 3} \times \mathbb{R}^{h\times w\times 3} \times \mathbb{R}^{h\times w\times 3} \times \mathbb{R}^{h\times w\times 3} \rightarrow \mathbb{R}^{n}, \\ &\omega\left(\mathcal{C}_S[Q], \mathcal{C}_T[M], \mathbf{Q}, \mathbf{M}\right) \rightarrow W \end{aligned} \tag{3.6}$$

At test time, we multiply W element-wise with V. Intuitively, this allows the network to learn to filter outliers with a supervised term (visibility) and a free variable (weights).

Loss

We allow this component to be optimized in a self-supervised manner. We do not generate ground-truth labels for the weights as we want the model to learn to filter inliers/outliers in a manner that reduces the pose objective function.

3.4. Differentiable Weighted Procrustes (ρ)

Pose Optimization is the final stage which minimizes the alignment objective between the two set of points $\mathbf{Q}$ and $\mathbf{M}$. The alignment objective we minimize is:

$$\frac{1}{n}\sum_{i=1}^{n} w_i ||\mathbf{m}_i - R\mathbf{q}_i - \mathbf{t}||^2 \tag{3.7}$$

where $R, \mathbf{t}$ are defined as at the beginning of Chapter 3.

Note that because of the quadratic term, outliers would have a big effect if they were not down-weighted by w_i.

The solution to this problem is known as the Weighted Procrustes analysis. As shown by [27], the closed solution for R and $\mathbf{t}$ is:

$$\begin{aligned} R &= USV^T \\ \mathbf{t} &= (\mathbf{M} - R\mathbf{Q})diag(\tilde{W})\mathbf{1} \end{aligned} \tag{3.8}$$

where $\tilde{W}$ are the normalized weights, $USV^T = SVD(\Sigma_{\mathbf{QM}})$, $\Sigma_{\mathbf{QM}} = \mathbf{M}KWK\mathbf{Q}^T$, $K = I - \sqrt{\tilde{W}}\sqrt{\tilde{W}}^T$, and $S = diag(1, \ldots, 1, det(U), det(V))$.

3.5. End-to-end Loss

As specified in the previous section, we don't use ground-truth annotations for the correspondence weighting. We use two objective functions: *Pose loss* and *Correspondence loss*. The differentiable pipeline allows gradients to flow from these objectives to the *Correspondence Weighting* (ω) component. Additionally, the *Correspondence and Visibility Prediction* (φ) parameters are informed by these objectives, further improving their performance. See Section 4.7 for the corresponding ablation study.

Pose loss This function minimizes the distance between the estimated relative pose, $\mathcal{T}_{TS}$, and the ground-truth pose $\mathcal{T}_{TS}^{gt}$.

We empirically found that representing the pose as a dual-quaternion resulted in the best performance. Dual-quaternions combine the rotation and translation components of the relative pose into a single representation which is compact, as it doesn't require additional memory such as the matrix-vector representation, is continuous, and can be interpolated.

For details about dual-quaternions algebra, we refer the reader to [28].

$$\mathcal{L}_{pose} = \frac{1}{n}||dq(\mathcal{T}_{TS})dq(\mathcal{T}_{TS}^{gt})^* - I_{dq}||^2 \tag{3.9}$$

We use Equation 3.9 as our pose objective. dq is a mapping which transforms poses in matrix notation to dual-quaternion. The * denotes the conjugate operation. For unit quaternion, the conjugate is the same as the inverse. I_{dq} denotes the identity dual-quaternion: $[1, 0, 0, 0]\ [0, 0, 0, 0]$

Correspondence loss Additionally, we minimize the distance between the 3D matches $\mathbf{M}$ and its ground-truth $\mathbf{M}_{gt}$. We use V_{gt} to mask occluded matches from the optimization.

$$\mathcal{L}_{matches} = V_{gt}||\mathbf{M} - \mathbf{M}_{gt}||^{q} \tag{3.10}$$

Where $q = 0.4$.

4. Experiments and Results

In this chapter, we present experiments and their quantitative and qualitative results to evaluate the approach presented in this thesis for the different tasks it solves. First, we discuss implementation details (Section 4.1) and the data used to train and evaluate the model (Section 4.2). We then evaluate the performance of our method in the three tasks it tackles: Matching (Section 4.3), Correspondence Weighting (Section 4.4) and Pose Optimization (Section 4.5). Finally, we discuss the runtime performance (Section 4.6) and ablation studies (Section 4.7) to justify different design choices.

4.1. Implementation Details

In this section, we elaborate on the implementation details of our method. First, we will define the dimensions used in Chapter 3:

- The dimensions of the frames used as input to the *Correspondence and Visibility Prediction* component (φ) are $H = 224$ and $W = 224$. We center and then crop the images from ScanNet to resize them from 640×480 to 224×224.
- The dimensions of the predicted heat maps $\mathcal{H}_i$ are $h = 32$ and $w = 224$.
- The number of matches we predict is $n = 1024$. Therefore, the dimensions of the output volume $\mathcal{H}$ are $32 \times 32 \times 1024$.

Table 4.1.: Number of parameters per component of our proposed architecture.

Component	**# Parameters**
Correspondence and Visibility Prediction (φ)	19949982
Encoder	809208
Bottleneck	886464
Decoder	17885213
Visibility	369097
Correspondence Weighting (ω)	8977
Total	**19958959**

Our model requires approximately 20M parameters. See Table 4.1 for a detailed view of the number of parameters per component.

For more details about the architecture, we refer the reader to Appendix A.

Training details. We implement our approach in PyTorch [29]. We use Stochastic Gradient Descent as our optimizer with momentum 0.9, a learning rate of 0.05 and a batch size of 16. Additionally, we decay the learning rate by a factor of 0.1 every 30k iterations. For regularization, we use a weight decay of 1^{-5}.

During training, we apply the following augmentations: We invert the pairs with a probability of 0.5. This means that instead of estimating matches and the pose from $s \rightarrow t$, we do it from $t \rightarrow s$. We also apply random changes to the brightness, contrast, saturation and hue of the RGB frames.

We train our method in two steps. Firstly, we train the correspondence and visibility prediction until convergence which is reached after 60k iterations. The loss used for this first step is $\lambda_{\mathcal{H}}\mathcal{L}_{\mathcal{H}} + \lambda_V\mathcal{L}_V$, where $\lambda_{\mathcal{H}} = 1$ and $\lambda_V = 1$. Afterwards, we include the correspondence weighting component in the optimization and we train the whole architecture with the end-to-end loss. $\lambda_{\mathcal{H}}\mathcal{L}_{\mathcal{H}} + \lambda_V\mathcal{L}_V + \lambda_{pose}\mathcal{L}_{pose} + \lambda_{matches}\mathcal{L}_{matches}$ with $\lambda_{\mathcal{H}} = 0.1$, $\lambda_V = 0.1$, $\lambda_{pose} = 10$ and $\lambda_{matches} = 0.5$. This second phase converges after 80k iterations.

4.2. Dataset

We use ScanNet [30] to train and evaluate our method for correspondence matching, correspondence weighting and robust pose optimization. ScanNet is a video dataset of 2.5 million views in more than 1500 scans of indoor environments. Each scan is a video sequence of frames, and for each frame, it provides the following information: color image, depth image and camera pose. ScanNet also provides the camera parameters for each scan.[1]

To train and test our method, we sample pairs of frames belonging to the same scan sequence where the overlap between the frames lies within an interval. Unlike previous works [15, 17], which enforce the overlap to be in the range $[0.4, 0.8]$, we empirically found that an overlap within $[0.1, 1.0]$ includes more variability among the pairs resulting in a more robust method for wide-baseline scenarios. However, this might create a more unstable optimization during training since the number of correspondences can be smaller resulting in a degenerate case. Figure 4.2 shows the distribution of poses for both overlap constraints. The overlap is computed using the ground truth camera poses, depth frames and camera parameters of the scene. We obtain 601k train pairs and 154k validation pairs.

For each frame f of a sequence, we compute its overlap with frames $f + i, i \in [10, 20, \ldots, 100]$. If the overlap does not lie in the interval $[0.1, 1.0]$, it is discarded.

[1] ScanNet provides more data and annotations but we only mention the data used by our method.

We label each pixel from the frame as visible, occluded or unknown. A pixel in f is visible on $f + i$, if the distance between the 3D correspondences is at most 0.03 meters. If the distance is bigger than 0.08 meters or the coordinates of the correspondence fall outside the image boundaries, then the pixel is labeled as occluded. In any other case, such as invalid depth values or distance between 0.03 and 0.08 meters, the pixel is labeled as unknown.

Figure 4.1 shows the relative poses in the dataset for train and validation splits.

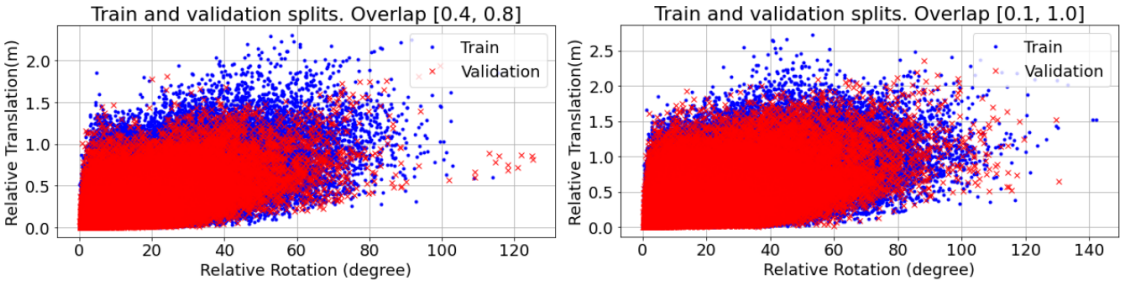

Figure 4.1.: Visualization of the poses where overlap between frames lies in $[0.4, 0.8]$ (left) and $[0.1, 1.0]$ (right). We can observe that the distribution of poses generated with the less constrained interval (right) has more variation.

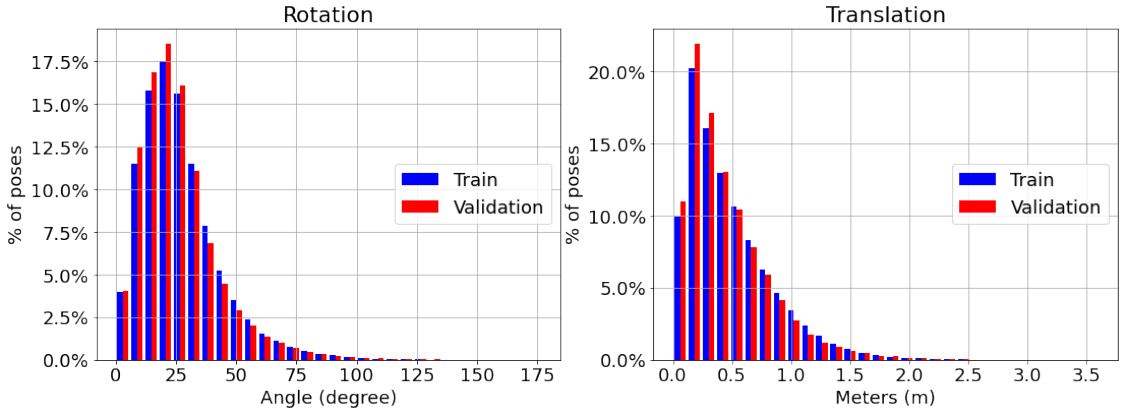

Figure 4.2.: Distribution of the poses for train and validation splits. We can see how both components (rotation and translation) contain large elements. This exposes our method to wide-baseline pairs which are more challenging.

4.3. Matching Evaluation

We first evaluate correspondence estimation for pairs of RGB images.

Baselines. We compare our method with two classic feature descriptors and matching algorithms: SIFT [2] and ORB [3]. We use the implementation provided by OpenCV [31] with the recommended configuration which empirically proved to be the best one. We also compare our method with the current state-of-the-art, LoFTR [18], with the publicly available implementation[2]. We filter out the correspondences with a confidence score $\theta < 0.2$ as recommended by the authors in the original work.

For our method, LRCE, we evaluate four different weighting schemes and filter out the matches with weight $w < 0.5$. The first weighting scheme, LRCE(A), uses a weight of 1 for all correspondences, therefore, it considers all predicted matches. The second scheme, LRCE(V), uses the predicted visibility score as weight. The third, LRCE(W), uses the weight score to filter out correspondences, and finally, LRCE(V*W), combines both visibility and weight scores.

Note that SIFT, ORB, LRCE(A), LRCE(V) methods only use RGB frames. The variants LRCE(W) and LRCE(V*W) also use the depth frame in order to predict the weights.

Quantitative Results. Table 4.2 shows the 2D/3D performance of our network compared to the other methods. 2D/3D-err columns show the average distance between the predicted matches and the ground truth. The 2D/3D-acc columns denote the accuracy of the predicted matches. A match is considered correct in 2D if the distance is less than 20 pixels. For 3D, the threshold is set at 0.10 meters. The current state-of-the-art, LoFTR, outperforms all other methods by a significant margin. For our method, LRCE, we see that adding the weighting scheme, either visibility or weights, significantly improves the performance. As previously explained in Section 3.2, without visibility the method cannot express when a query pixel is not visible in the target frame. When using weights, the method has information about the depth information and therefore the performance increases. Finally, by combining both, the method can reason about the supervised classification variable (visibility) and the self-supervised output (weights).

Figure 4.3 illustrates the error in pixels/meters for the 2D/3D correspondences grouped by translation/rotation threshold. We can observe how LoFTR is very resilient to the translation component of the relative pose, both in 2D and 3D. However, we can see how LRCE outperforms LoFTR for big rotations.

For visibility classification, our model achieves an accuracy of 0.89, a precision of 0.84 and a recall of 0.92 on the validation dataset.

Qualitative Results. In Figure 4.4 we can see some examples of frames and the matches produced by the different baselines and our method. We filter out correspondences using the same thresholds as in paragraph 4.3. Hand-crafted algorithms produce very few matches, which is critical for indoor scenes where objects can be easily occluded

[2]https://github.com/zju3dv/LoFTR

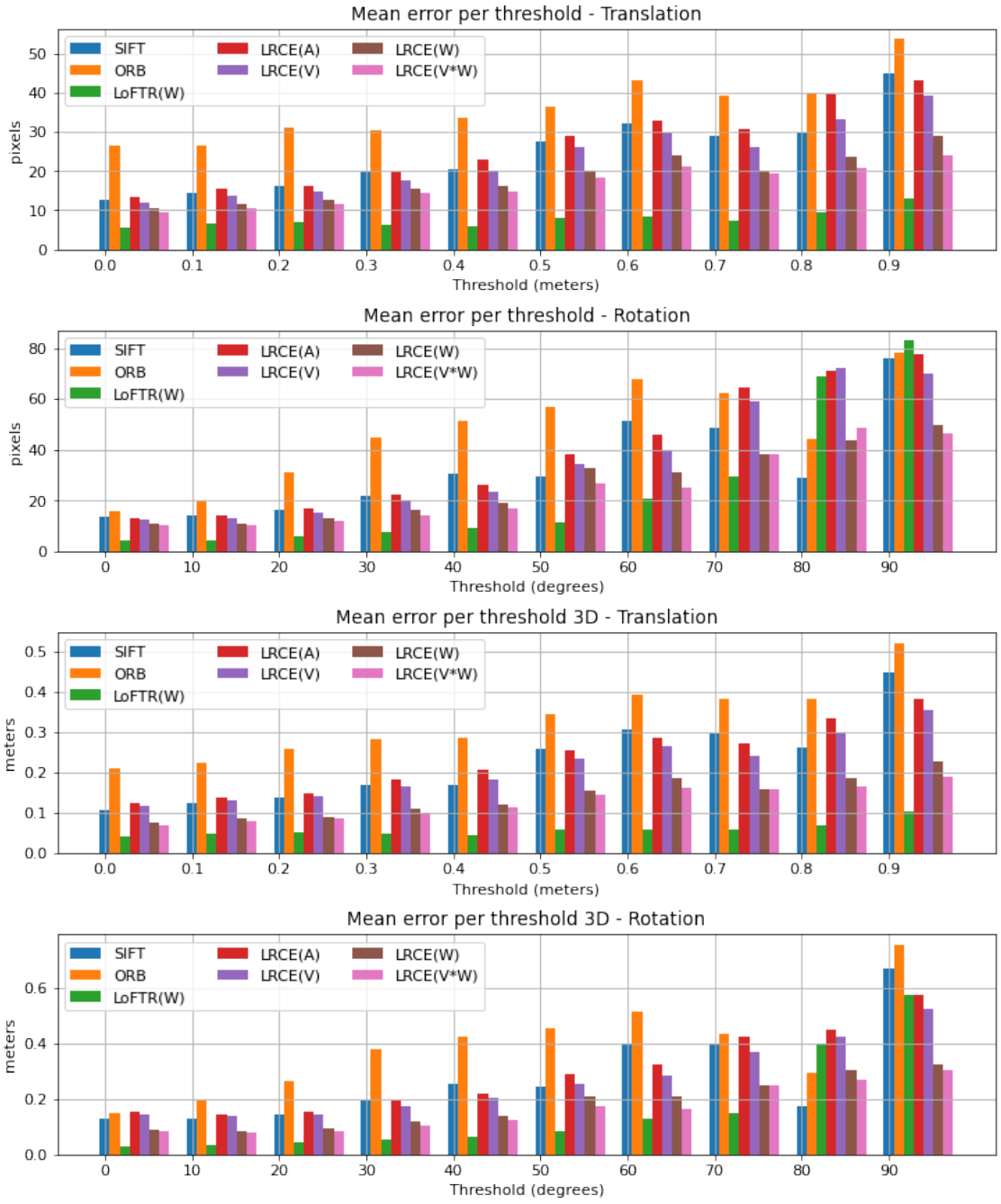

Figure 4.3.: Average 2D/3D error in pixels/meters per translation/rotation threshold. The current state-of-the-art, LoFTR, is robust to translation and outperforms the rest of the methods. Our method, LRCE, is robust to large rotations and outperforms the rest on this condition.

Table 4.2.: LoFTR outperforms all baselines followed by LRCE(V*W). 2D/3D errors are the average of pixel/point errors, and 2D/3D accuracy is the percentage of pixels/points with distance of at most 20 pixels/0.10 meters.

Method	**2D-err**	**3D-err**	**2D-acc**	**3D-acc**
SIFT [2]	18.56	0.16	0.88	0.86
ORB [3]	31.27	0.27	0.78	0.75
LoFTR [18]	**6.78**	**0.05**	**0.98**	**0.97**
LRCE(A)	23.04	0.21	0.72	0.59
LRCE(V)	17.84	0.15	0.85	0.76
LRCE(W)	13.63	0.10	0.92	0.85
LRCE(V*W)	11.88	0.08	0.93	0.89

and the overlap can be minimal. Learning-based methods produce more matches from which we show at most 16 to avoid cluttering.

4.4. Correspondence Weighting

In this section, we analyze the behaviour of the *Correspondence Weighting* module which is trained in a self-supervised manner from the pose objective.

Figure 4.5 shows the histograms of the predicted weight and visibility scores for visible, occluded and unknown matches. Firstly, we can see that the *Visibility Prediction* component successfully learns how to classify the matches, as it is evident that the predicted labels for visible matches are all near one. Similarly, we can see how the predicted labels for occluded points are close to zero. We also show the distribution for unknown matches, which we mask out from the loss during optimization as we do not know its ground truth label.

Secondly, we can observe how the *Correspondence Weighting* component learns to filter out occluded matches as all the predictions are close to zero. This means that this components successfully enables robust optimization as outliers will be down-weighted during pose estimation. For visible matches, we observe that the distribution contains most of the predictions on the $[0.5, 1.0]$ range, meaning they will have higher influence during pose estimation. However, some of the visible matches are assigned a lower weight. This has two explanations: 1) The *Correspondence Prediction* component failed at estimating an accurate match, therefore it is down-weighted to nullify its effect during pose estimation. 2) A match may be visible in the RGB frames, but it may become an outlier when back-projected to 3D coordinates due to noise in the depth frame or it being near surface discontinuities such as edges.

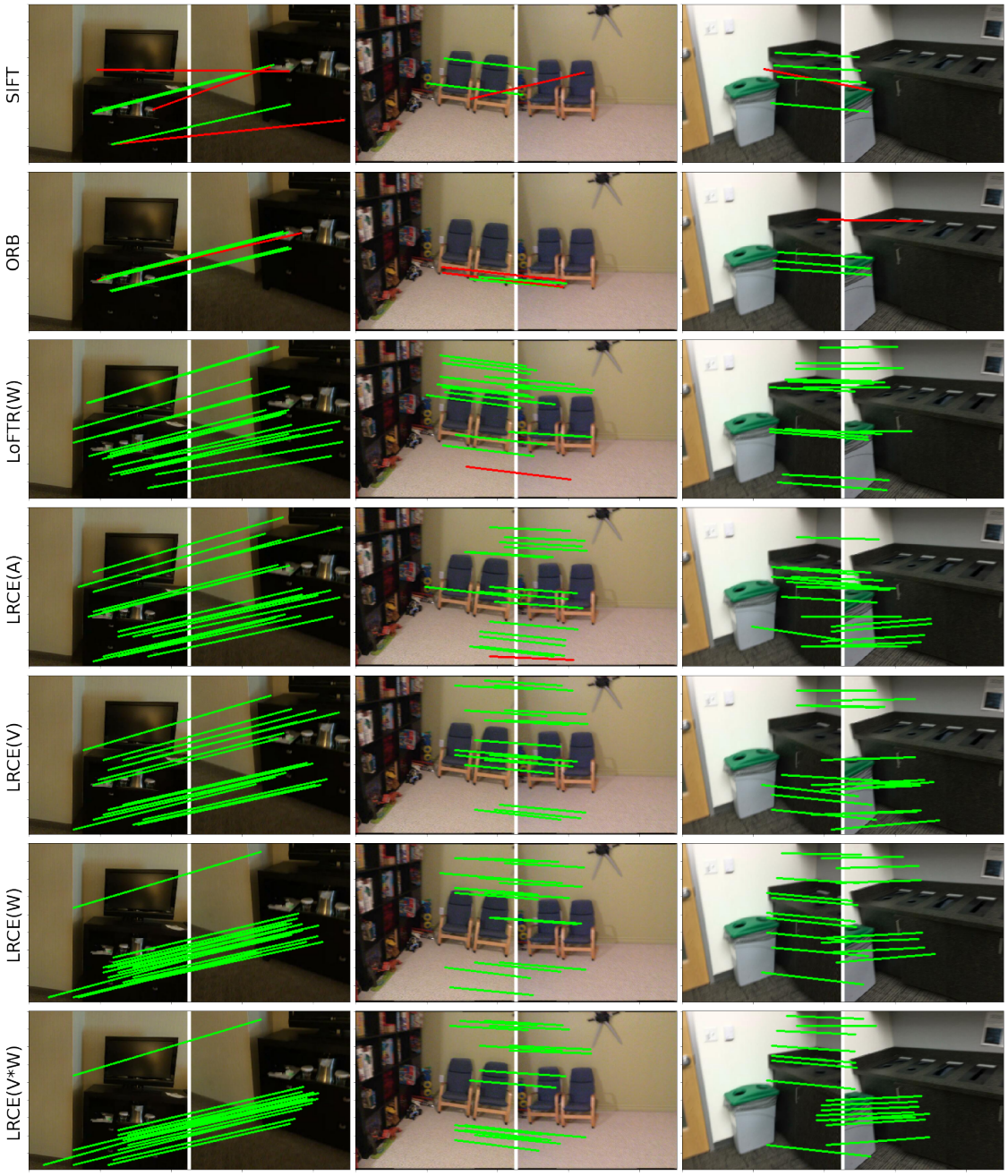

Figure 4.4.: Visualization of the matches predicted by the baselines and our method. Green indicates a correct match (within 20 pixels of the ground truth) and red an incorrect match. To avoid cluttering, we show 16 randomly selected correspondences after filtering out by the corresponding weighting scheme.

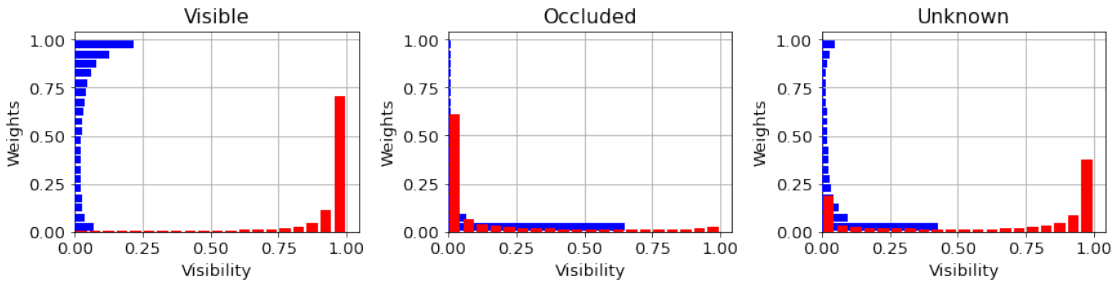

Figure 4.5.: Histograms for the predicted weights and visibility scores for visible, occluded and unknown matches. We can observe the model correctly classifies visible and occluded matches. Additionally, it is able to successfully learn to filter outliers in a self-supervised manner.

4.5. Pose Optimization

In this section, we evaluate the results for the task of relative camera pose optimization.

Baselines. For this task, we compare our method against LoFTR and ICP [22] implemented using Open3D [32].

Quantitative Results. Figure 4.6 displays cumulative curves for translation and rotation. We represent the rotation error as the angle between the ground truth and predicted rotations. For translation, we represent the error as the angle between the normalized vectors instead of the L2-norm. This is due to LoFTR not predicting the scale of the translation.

Table 4.3 lists the accuracy score per method. A pose is considered correct if the distance between the predicted and ground truth rotation and translation is below 20 degrees.

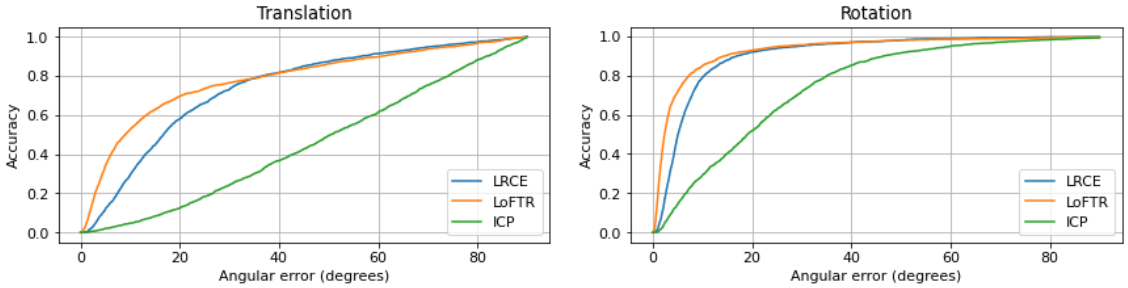

Figure 4.6.: Cumulative plots for Translation (left) and Rotation (right) of the accuracy of the model for allowed error. The Y-axis shows the percentage of predicted poses and the X-axis the allowed error.

Table 4.3.: Pose accuracy per method. The pose is considered correct if the angle error for both translation and rotation is under 20 degrees. LoFTR outperforms the rest of the methods, followed by LRCE. Due to the wide-baseline setup of the camera poses, ICP achieves a low accuracy.

Method	Accuracy
ICP	0.09
LRCE	0.58
LoFTR	**0.68**

Qualitative Results. Figure 4.7 shows examples of input frames aligned with the ground truth camera pose, our method and ICP.

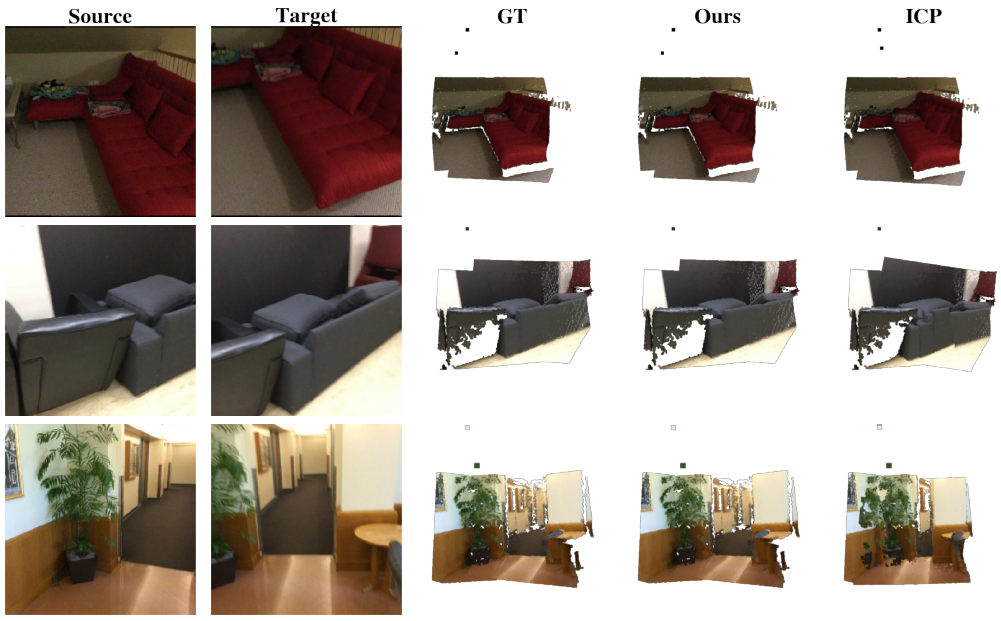

Figure 4.7.: Reconstruction from estimated camera poses on ScanNet pairs. Our method successfully estimates the camera transformation. In the last example, the pose regressed by ICP shows a clear misalignment.

4.6. Runtime Performance

We measure the execution time of a single forward pass of our method. The hardware used for this measurement is an NVIDIA GeForce RTX 3090 and 8 CPU cores. The inputs are two RGB-D frames of dimensions 224×224 and the output is the estimated relative pose. A single forward pass runs in approximately 16 ms. This is divided into two parts: 10.89 ms are from the *Correspondence and Visibility Prediction* and 5.11 ms from

Correspondence Weighting and *Weighted Procrustes*.

4.7. Ablation Study

In this section, we demonstrate the effects of different components of our method.

Visibility and weight scores effect. We can verify the effect of weighting for the tasks involved in this method. Firstly, Table 4.2 shows how the matching performance improved when correspondences are filtered based on their weight. This proves that a weight closer to zero is assigned to outliers.

Secondly, we evaluate the following weighting schemes in the pose estimation problem: i) *Equal (E)* sets equal weight for all matches, ii) *RANSAC(R)* filters outliers using RANSAC [4], iii) *Visibility (V)* weights correspondences using predicted visibility scores, iv) *Weights (W)* uses predicted weight scores, v) *Visibility * RANSAC (V*R)* vi) *Weights * RANSAC (W*R)* vii) *Visibility * Weights * RANSAC (V*W*R)* and viii) *Visibility * Weights (V*W)* multiply the scores provided by each weighting scheme.

Figure 4.8 shows the cumulative plots for all the weighting schemes and Table 4.4 lists their accuracies. We can observe that the method has the lowest performance when no correspondence weighting is used. Filtering outliers using RANSAC slightly boosts the performance. Using the supervised visibility score (V) or the self-supervised weights (W) further improves the performance. Finally, combining visibility and weights (V*W) obtains the best performance. Intuitively, this gives the network the flexibility to weight correspondences using the supervised label and to down weight outliers using the self-supervised score optimized from the final alignment loss.

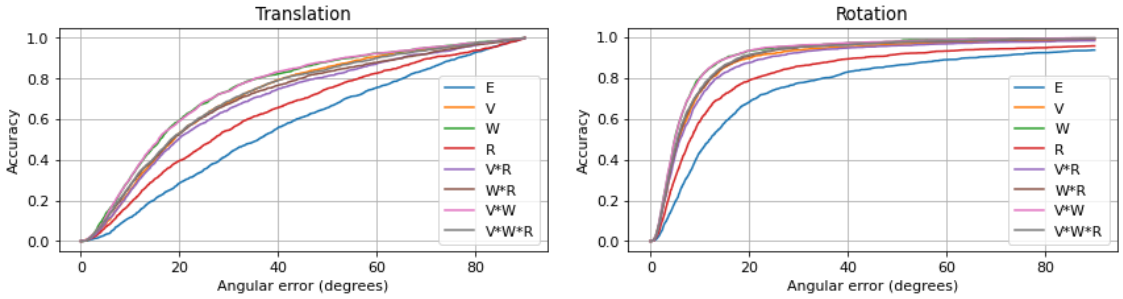

Figure 4.8.: Cumulative plots for translation (left) and rotation (right) of the accuracy per allowed error for the different weighting schemes.

Training *Correspondence and Visibility Prediction* component with end-to-end loss. In this experiment, we demonstrate that the *Correspondence and Visibility Prediction*

Table 4.4.: We can see that the weighting scheme (V*W) obtains the best performance. The network successfully learns to filter outliers in a self-supervised manner.

Method	**Accuracy**
Equal (E)	0.17
RANSAC (R)	0.29
Visibility * RANSAC (V*R)	0.46
Weights * RANSAC (W*R)	0.48
Visibility (V)	0.49
Visibility * Weights * RANSAC (V*W*R)	0.52
Weights (W)	0.58
Visibility * Weights (V*W)	**0.59**

Table 4.5.: Evaluation of correspondence prediction without the end-to-end loss.

Method	**2D-err**	**3D-err**	**2D-acc**	**3D-acc**
LRCE(A)	28.19	0.24	0.67	0.51
LRCE(V)	23.45	0.19	0.79	0.62
LRCE(W)	16.74	0.11	0.88	0.78
LRCE(V*W)	13.75	0.09	0.91	0.81

module benefits from the end-to-end loss. That is, the pose alignment gradients are able to inform the 2D correspondences.

We first train the *Correspondence and Visibility Prediction* component in a supervised manner using $\mathcal{L}_{\mathcal{H}}$ and $\mathcal{L}_{V}$ as detailed in Section 4.1. Then, we freeze this component and continue training with the *Correspondence Weighting* and the end-to-end loss.

Table 4.5 shows the evaluation of the matches produced by this model. When compared to Table 4.2, we can see that by allowing the gradients from the end-to-end loss the 2D accuracy increased by 2% and the 3D accuracy increases by 8%.

5. Limitations and Further Work

While we have shown that our method outperforms some of the baselines, it is still far from current state-of-the-art such as [18, 15, 17]. All these methods use the attention mechanism to gather information within and across frames which is a key feature for correspondence estimation. In contrast, Convolutional Neural Networks suffer from poor global context due to their limited receptive field.

Due to predicting correspondences through a heat map representation, the output of our model requires large amounts of memory. This limits us to predict low-resolution heat maps for a sparse set of correspondences which can be harmful in situations where the overlap between frames is minimum due to not having enough information to recover the pose. Additionally, constructing such volume with 2D convolutions requires a model with a large capacity because of the number of channels. Our approach uses 3D convolutions for building such volume, which allows us to slide the same filter across the height, width and depth but at the expense of run time.

To improve the current performance, we can implement a coarse-to-fine pipeline to refine the matches across different resolution levels. We can also change the method for selecting the pixels from which we compute correspondence. Instead of selecting the same arbitrary set for all pairs of frames, we can let the method suggest a set of query points based on their quality. Finally, we can train the network over a uniform distribution of relative poses to ensure that the network can generalize to wide-baseline camera transformations.

6. Conclusion

In this Master's Thesis, we present Learning Robust Correspondences Estimation (LRCE), an end-to-end differentiable neural network to tackle the relative pose estimation problem.

Our method first predicts sparse correspondences and visibility scores for a set of query pixels using the color frames. Then, utilizing the predicted matches and the depth information, the method predicts a set of confidence scores. These scores are learnt in a self-supervised manner from the end-to-end loss, which minimizes the distance between the predicted and ground truth poses. These confidence and visibility scores are then combined and used to weigh the matches in our differentiable Weighted Procrustes aligner, which estimates the relative pose in a robust manner.

Our experiments show that the end-to-end loss is able to train our model to predict confidence scores in a self-supervised manner and improve the performance of correspondence prediction. This results in a more robust pipeline for solving relative camera pose estimation.

A. Network Architecture Details

In this section, we describe in more detail the full architecture of our model. We adapted the following building blocks from DeepDeform [26]: *ResBlock*, *DownScaleBlock* and *UpScaleBlock*. Additionally, we define the block *ExpandBlock* which is used to increase its number of features. Due to memory limitations, this block is implemented with 3D convolutions. The 2D input of dimensions $h \times w \times c$ is extended to 3D as $h \times w \times d(=1) \times c$. We then increase d while decreasing c. The advantage with respect to the 2D approach is that we don't need additional parameters to increase the number of channels, as we do it by increasing the d dimension and then reinterpreting the dimensions of the tensor.

Our model is divided into two architectures: *Correspondence and Visibility Prediction (Encoder, Bottleneck, Visibility, Decoder)* and *Correspondence Weighting*. Similar to DeepDeform, we use U-Net [33] skip connections between the encoder and the decoder.

Table A.1.: **ResBlock.** It is the main building block of our architecture. It contains a skip connection between the input and the second convolution layer.

ResBlock(f0, f1)				
Layer	**Channels(input, output)**	**Kernel**	**Stride**	**Padding**
Conv2d	(f0, f1)	3	1	1
BatchNorm2d	f1	-	-	-
ReLU	-	-	-	-
Conv2d	(f1, f1)	3	1	1
BatchNorm2d	f1	-	-	-
ReLU	-	-	-	-

Table A.2.: **DownScaleBlock.** It is used in the *Encoder* component to extract features from the inputs, reducing its size while increasing the channels.

DownScaleBlock(f0, f1)				
Layer	**Channels(input, output)**	**Kernel**	**Stride**	**Padding**
Conv2d	(f0, f1)	4	2	1
BatchNorm2d	f1	-	-	-
ReLU	-	-	-	-
ResBlock	f1	-	-	-
ResBlock	(f1, f1)	-	-	-

Table A.3.: **UpScaleBlock.** It is used in the *Decoder* component to increase the size of the input while decreasing the number of channels.

UpScaleBlock(f0, f1)				
Layer	**Channels(input, output)**	**Kernel**	**Stride**	**Padding**
ResBlock	(f0, f0)	-	-	-
ConvTranspose2d	(f0, f1)	6	2	2
BatchNorm2d	f1	-	-	-
ReLU	-	-	-	-

Table A.4.: **ExpandBlock.** It is used in the *Decoder* component to decrease the number of features of the input while increasing its width. The rest of the dimensions stay the same.

ExpandBlock(f0, f1)				
Layer	**Channels(input, output)**	**Kernel**	**Stride**	**Padding**
ConvTranspose3d	(f0, f0)	(2, 3, 3)	(2, 1, 1)	(0, 1, 1)
ConvTranspose3d	(f0, f1)	(2, 3, 3)	(2, 1, 1)	(0, 1, 1)
BatchNorm3d	f1	-	-	-
ReLU	-	-	-	-

Table A.5.: **Encoder**. The encoder takes one RGB image as input and outputs a $7 \times 7 \times 96$ feature tensor.

Encoder				
Layer	**Channels(input, output)**	**Kernel**	**Stride**	**Padding**
DownScaleBlock	(3, 24)	-	-	-
DownScaleBlock	(32, 32)	-	-	-
DownScaleBlock	(32, 48)	-	-	-
DownScaleBlock	(48, 64)	-	-	-
DownScaleBlock	(64, 96)	-	-	-

Table A.6.: **Bottleneck.** It takes the concatenated output of the *Encoder* for both RGB frames, $7 \times 7 \times 192$, and outputs a tensor of dimensions $7 \times 7 \times 256$.

Bottleneck				
Layer	**Channels(input, output)**	**Kernel**	**Stride**	**Padding**
Conv2d	(192, 128)	-	-	-
BatchNorm2d	128	-	-	-
ReLU	-	-	-	-
Conv2d	(128, 192)	-	-	-
BatchNorm2d	192	-	-	-
ReLU	-	-	-	-
Conv2d	(192, 256)	-	-	-
BatchNorm2d	256	-	-	-
ReLU	-	-	-	-

Table A.7.: **Decoder**. The decoder takes the output of the *Bottleneck* and outputs a tensor of dimensions $32 \times 32 \times 1024$. We add skip connections between the intermediate results of the *Encoder* and the *UpScaleBlock*.

Decoder				
Layer	**Channels(input, output)**	**Kernel**	**Stride**	**Padding**
UpScaleBlock	(256, 256)	-	-	-
UpScaleBlock	(256 + 2 * 96, 256)	-	-	-
UpScaleBlock	(256 + 2 * 64, 256)	-	-	-
ConvTranspose3d	(256, 256)	(2, 3, 3)	(1, 1, 1)	(0, 1, 1)
ConvTranspose3d	(256, 192)	(2, 3, 3)	(2, 1 ,1)	(0, 1, 1)
BatchNorm3d	192	-	-	-
ReLU	-	-	-	-
ExpandBlock	(192, 128)	-	-	-
ExpandBlock	(128, 64)	-	-	-
ExpandBlock	(64, 16)	-	-	-
ConvTranspose3d	(16, 4)	(2, 3, 3)	(2, 1, 1)	(0, 1, 1)
ConvTranspose3d	(4, 1)	(2, 3, 3)	(2, 1, 1)	(0, 1, 1)

Table A.8.: **Visibility.** It takes the output of the *Bottleneck* and outputs a vector of size 1024.

Visibility				
Layer	**Channels(input, output)**	**Kernel**	**Stride**	**Padding**
ExpandBlock	(256, 32)	-	-	-
ExpandBlock	(32, 16)	-	-	-
ExpandBlock	(16, 8)	-	-	-
ExpandBlock	(8, 4)	-	-	-
ExpandBlock	(4, 2)	-	-	-
Conv3d	(2, 2)	(3, 4, 4)	(1, 1, 1)	(1, 0, 0)
Conv3d	(2, 1)	(3, 4, 4)	(1, 1, 1)	(1, 0, 0)
Sigmoid	-	-	-	-

Table A.9.: **Correspondence Weighting.** This component takes as input the color of the correspondences as a tensor of dimensions $32 \times 32 \times 6$ combined with the back-projected points associated with the correspondences $32 \times 32 \times 6$. It output vector of size 1024 with weight scores $\in [0, 1]$.

Correspondence Weighting				
Layer	**Channels(input, output)**	**Kernel**	**Stride**	**Padding**
Conv2d	(12, 16)	3	1	1
BatchNorm2d	16	-	-	-
ReLU	(16, 16)	-	-	-
ResBlock	(16, 16)	-	-	-
ResBlock	(16, 16)	-	-	-
ResBlock	(16, 16)	-	-	-
Conv2d	(16, 1)	3	1	1
Sigmoid	-	-	-	-

List of Figures

List of Tables

Bibliography

[1] M. Zollhöfer, A. G. Patrick Stotko, C. Theobalt, M. Nießner, R. Klein, and A. Kolb. "State of the Art on 3D Reconstruction with RGB-D Cameras". In: (2018).

[2] D. G. Lowe. "Distinctive Image Features from Scale-Invariant Keypoints". en. In: *International Journal of Computer Vision* 60.2 (Nov. 2004), pp. 91–110. ISSN: 1573-1405. DOI: 10.1023/B:VISI.0000029664.99615.94.

[3] E. Rublee, V. Rabaud, K. Konolige, and G. Bradski. "ORB: An efficient alternative to SIFT or SURF". In: *2011 International Conference on Computer Vision*. Nov. 2011, pp. 2564–2571. DOI: 10.1109/ICCV.2011.6126544.

[4] M. A. Fischler and R. C. Bolles. "Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography". In: *Commun. ACM* 24 (1981), pp. 381–395.

[5] A. Dai, A. X. Chang, M. Savva, M. Halber, T. Funkhouser, and M. Nießner. "ScanNet: Richly-annotated 3D Reconstructions of Indoor Scenes". In: *Proc. Computer Vision and Pattern Recognition (CVPR), IEEE*. 2017.

[6] C. Harris, M. Stephens, et al. "A combined corner and edge detector". In: *Alvey vision conference*. Vol. 15. 50. Citeseer. 1988, pp. 10–5244.

[7] H. Bay, T. Tuytelaars, and L. V. Gool. "Surf: Speeded up robust features". In: *European conference on computer vision*. Springer. 2006, pp. 404–417.

[8] D. G. Viswanathan. "Features from accelerated segment test (fast)". In: *Proceedings of the 10th workshop on image analysis for multimedia interactive services, London, UK*. 2009, pp. 6–8.

[9] B. Glocker, S. Izadi, J. Shotton, and A. Criminisi. "Real-Time RGB-D Camera Relocalization". In: *International Symposium on Mixed and Augmented Reality (ISMAR)*. IEEE, Oct. 2013. URL: https://www.microsoft.com/en-us/research/publication/real-time-rgb-d-camera-relocalization/.

[10] K. M. Yi, E. Trulls, Y. Ono, V. Lepetit, M. Salzmann, and P. Fua. "Learning to Find Good Correspondences". In: *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*. June 2018, pp. 2666–2674. DOI: 10.1109/CVPR.2018.00282.

[11] I. Rocco, M. Cimpoi, R. Arandjelović, A. Torii, T. Pajdla, and J. Sivic. "Neighbourhood Consensus Networks". In: *Advances in Neural Information Processing Systems*. Vol. 31. Curran Associates, Inc., 2018. URL: https://proceedings.neurips.cc/paper/2018/hash/8f7d807e1f53eff5f9efbe5cb81090fb-Abstract.html.

[12] I. Rocco, R. Arandjelović, and J. Sivic. "Efficient Neighbourhood Consensus Networks via Submanifold Sparse Convolutions". In: *arXiv:2004.10566 [cs]* (Apr. 2020). arXiv: 2004.10566. URL: http://arxiv.org/abs/2004.10566.

[13] I. Rocco, M. Cimpoi, R. Arandjelović, A. Torii, T. Pajdla, and J. Sivic. "NCNet: Neighbourhood Consensus Networks for Estimating Image Correspondences". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 44.2 (Feb. 2022), pp. 1020–1034. ISSN: 1939-3539. DOI: 10.1109/TPAMI.2020.3016711.

[14] D. Sun, X. Yang, M.-Y. Liu, and J. Kautz. "PWC-Net: CNNs for Optical Flow Using Pyramid, Warping, and Cost Volume". In: *arXiv:1709.02371 [cs]* (June 2018). arXiv: 1709.02371. URL: http://arxiv.org/abs/1709.02371.

[15] P.-E. Sarlin, D. DeTone, T. Malisiewicz, and A. Rabinovich. "SuperGlue: Learning Feature Matching with Graph Neural Networks". In: *arXiv:1911.11763 [cs]* (Mar. 2020). arXiv: 1911.11763. URL: http://arxiv.org/abs/1911.11763.

[16] M. Cuturi. "Sinkhorn Distances: Lightspeed Computation of Optimal Transportation Distances". In: (2013). DOI: 10.48550/ARXIV.1306.0895. URL: https://arxiv.org/abs/1306.0895.

[17] B. Roessle and M. Nießner. *End2End Multi-View Feature Matching using Differentiable Pose Optimization*. arXiv:2205.01694. arXiv:2205.01694 [cs] type: article. May 2022. DOI: 10.48550/arXiv.2205.01694. URL: http://arxiv.org/abs/2205.01694.

[18] J. Sun, Z. Shen, Y. Wang, H. Bao, and X. Zhou. "LoFTR: Detector-Free Local Feature Matching with Transformers". In: *arXiv:2104.00680 [cs]* (Apr. 2021). arXiv: 2104.00680. URL: http://arxiv.org/abs/2104.00680.

[19] Y. Wang and J. M. Solomon. "Deep Closest Point: Learning Representations for Point Cloud Registration". In: *arXiv:1905.03304 [cs]* (May 2019). arXiv: 1905.03304. URL: http://arxiv.org/abs/1905.03304.

[20] R. Hartley. "In defense of the eight-point algorithm". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 19.6 (1997), pp. 580–593. DOI: 10.1109/34.601246.

[21] R. Hartley and A. Zisserman. *Multiple View Geometry in Computer Vision*. 2nd ed. Cambridge University Press, 2004. DOI: 10.1017/CBO9780511811685.

[22] P. Besl and N. D. McKay. "A method for registration of 3-D shapes". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 14.2 (Feb. 1992), pp. 239–256. ISSN: 1939-3539. DOI: 10.1109/34.121791.

[23] A. Kendall, M. Grimes, and R. Cipolla. "PoseNet: A Convolutional Network for Real-Time 6-DOF Camera Relocalization". In: *arXiv:1505.07427 [cs]* (Feb. 2016). arXiv: 1505.07427. URL: http://arxiv.org/abs/1505.07427.

[24] T. Sattler, Q. Zhou, M. Pollefeys, and L. Leal-Taixe. "Understanding the Limitations of CNN-based Absolute Camera Pose Regression". In: *arXiv:1903.07504 [cs]* (Mar. 2019). arXiv: 1903.07504. URL: http://arxiv.org/abs/1903.07504.

[25] I. Melekhov, J. Ylioinas, J. Kannala, and E. Rahtu. "Relative Camera Pose Estimation Using Convolutional Neural Networks". In: *arXiv:1702.01381 [cs]* (July 2017). arXiv: 1702.01381. URL: http://arxiv.org/abs/1702.01381.

[26] A. Božič, M. Zollhöfer, C. Theobalt, and M. Nießner. "DeepDeform: Learning Non-rigid RGB-D Reconstruction with Semi-supervised Data". In: *arXiv:1912.04302 [cs]* (Mar. 2020). arXiv: 1912.04302. URL: http://arxiv.org/abs/1912.04302.

[27] C. Choy, W. Dong, and V. Koltun. "Deep Global Registration". In: *arXiv:2004.11540 [cs, eess]* (May 2020). arXiv: 2004.11540. URL: http://arxiv.org/abs/2004.11540.

[28] B. Kenwright. "Dual-Quaternions: From Classical Mechanics to Computer Graphics and Beyond". In: *www.xbdev.net* (2012), pp. 1–11.

[29] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Kopf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala. "PyTorch: An Imperative Style, High-Performance Deep Learning Library". In: *Advances in Neural Information Processing Systems 32*. Ed. by H. Wallach, H. Larochelle, A. Beygelzimer, F. d 'Alché-Buc, E. Fox, and R. Garnett. Curran Associates, Inc., 2019, pp. 8024–8035. URL: http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf.

[30] A. Dai, M. Nießner, M. Zollhöfer, S. Izadi, and C. Theobalt. "BundleFusion: Real-time Globally Consistent 3D Reconstruction using On-the-fly Surface Reintegration". In: arXiv:1604.01093 (Feb. 2017). number: arXiv:1604.01093 arXiv:1604.01093 [cs]. URL: http://arxiv.org/abs/1604.01093.

[31] Itseez. *Open Source Computer Vision Library*. https://github.com/itseez/opencv. 2015.

[32] Q.-Y. Zhou, J. Park, and V. Koltun. "Open3D: A Modern Library for 3D Data Processing". In: *arXiv:1801.09847* (2018).

[33] O. Ronneberger, P. Fischer, and T. Brox. "U-Net: Convolutional Networks for Biomedical Image Segmentation". In: *CoRR* abs/1505.04597 (2015). arXiv: 1505.04597. URL: http://arxiv.org/abs/1505.04597.