

# Learning to Optimize Non-Rigid Tracking

Yang Li<sup>1,4</sup> Aljaž Božič<sup>4</sup> Tianwei Zhang<sup>1</sup> Yanli Ji<sup>1,2</sup> Tatsuya Harada<sup>1,3</sup> Matthias Nießner<sup>4</sup>

<sup>1</sup>The University of Tokyo, <sup>2</sup>UESTC, <sup>3</sup>RIKEN, <sup>4</sup>Technical University Munich

## Abstract

One of the widespread solutions for non-rigid tracking has a nested-loop structure: with Gauss-Newton to minimize a tracking objective in the outer loop, and Preconditioned Conjugate Gradient (PCG) to solve a sparse linear system in the inner loop. In this paper, we employ learnable optimizations to improve tracking robustness and speed up solver convergence. First, we upgrade the tracking objective by integrating an alignment data term on deep features which are learned end-to-end through CNN. The new tracking objective can capture the global deformation which helps Gauss-Newton to jump over local minimum, leading to robust tracking on large non-rigid motions. Second, we bridge the gap between the preconditioning technique and learning method by introducing a ConditionNet which is trained to generate a preconditioner such that PCG can converge within a small number of steps. Experimental results indicate that the proposed learning method converges faster than the original PCG by a large margin.

## 1. Introduction

Non-rigid dynamic objects, *e.g.*, humans and animals, are important targets in both computer vision and robotics applications. Their complex geometric shapes and non-rigid surface changes result in challenging problems for tracking and reconstruction. In recent years, using commodity RGB-D cameras, the seminal works such as DynamicFusion [20] and VolumeDeform [13] made their efforts to tackle this problem and obtained impressive non-rigid reconstruction results. At the core of DynamicFusion and VolumeDeform are non-linear optimization problems. However, this optimization can be slow, and can also result in undesired local minima. In this paper, we propose a learning-based method that finds optimization steps that expand the convergence radius (*i.e.*, avoids local minima) and also makes convergence faster. We test our method on the essential inter-frame non-rigid tracking task, *i.e.*, to find the deformation between two RGB-D frames, which is a high-dimensional and non-convex problem. The absence

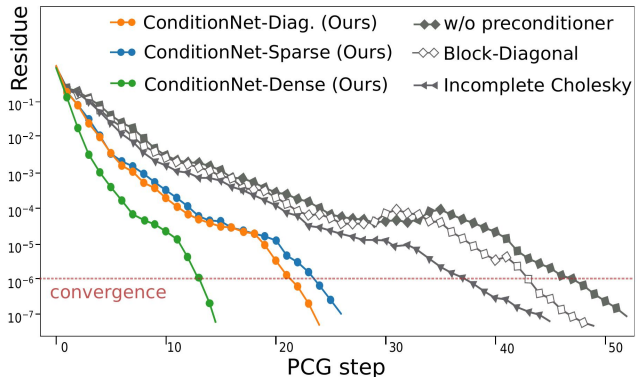


Figure 1. PCG convergence using different preconditioners. The curves show the average convergence on the testing dataset. Note that our final method (green curve) requires 3 times fewer PCG steps to achieve the same residual ( $10^{-6}$ ) than the best baseline (dashed line).

of an object template model, large non-overlapping area, and observation noise in both source and target frame make this problem even more challenging. This section will first review the classic approach and then put our contributions into context.

**Non-rigid Registration** The non-rigid surface motions can be roughly approximated through the “deformation graph” [24]. In this deformable model, all of the unknowns, *i.e.*, the rotations and translations, are denoted as  $\mathcal{G}$ . Given two RGB-D frames, the goal of non-rigid registration is to determine the  $\mathcal{G}$  that minimizes the typical objective function:

$$\min_{\mathcal{G}} \{ \mathbf{E}_{fit}(\mathcal{G}) + \lambda \mathbf{E}_{reg}(\mathcal{G}) \} \quad (1)$$

where  $\mathbf{E}_{fit}$  is the data fitting term that measures the closeness between the warped source frame and the target frame. Many different data fitting terms have been proposed over the past decades, such as the geometric point-to-point and point-to-plane constraints [16, 31, 20, 13] sparse SIFT descriptor correspondences [13], and the dense color term [31], *etc.* The term  $\mathbf{E}_{reg}$  regularizes the problem by favoring locally rigid deformation. Coefficient  $\lambda$  balances these

two terms. The energy (1) is minimized by iterating the Gauss-Newton update step [2] till convergence. Inside each Gauss-Newton update step a large linear system needs to be solved, for which an iterative preconditioned conjugate gradient (PCG) solver is commonly used.

This classic approach cannot properly handle large non-rigid motions since the data fitting term  $E_{fit}$  in the energy function (1) is made of local constraints (e.g., dense geometry or color maps), which only work when they are close to the global solution, or global constraints that are prone to noise (e.g., sparse descriptor). In the case of large non-rigid motions, these constraints cannot provide convergent residuals and lead to tracking failure. In this paper, we alleviate the non-convexity of this problem by introducing a deep feature alignment term into  $E_{fit}$ . The deep features are extracted through an end-to-end trained CNN. We assume that, by leveraging the large receptive field of convolutional kernels and the nature of the data-driven method, the learned feature can capture the global information which helps Gauss-Newton to jump over local minimums.

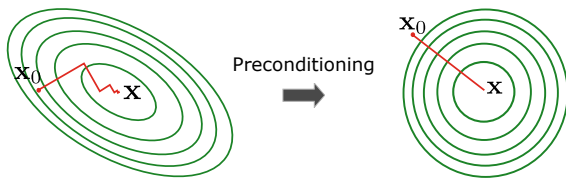


Figure 2. Example of using the Deepest Decent to solve a 2D system. Deepest Decent needs multiple steps to converge on an ill-conditioned system (left) and only one step on a perfectly conditioned system (right). Intuitively, Preconditioning is trying to modify the energy landscape from an elliptical paraboloid into a spherical one such that from any initial position, the direction of the first-order derivative directly points to the solution.

As illustrated in Fig. 2, preconditioning speeds up the convergence of an iterative solver. The general idea behind preconditioning is to use a matrix, called *preconditioner*, to modify an ill-posed system into a well-posed one that is easier to solve. As the hard-coded block-diagonal preconditioner was not designed specifically for the non-rigid tracking task, the existing non-rigid tracking solvers are still time-consuming. We argue that PCG converges much faster if the design of the preconditioner involves prior expert knowledge of this specific task. Then we raise the question: *does the data-driven method learn a good preconditioner?* In this paper, we exploit this idea by training neural network to generate a preconditioner such that PCG can converge within a few steps.

Our contribution is twofold:

- We introduce a deep feature fitting term based on end-to-end learned CNN for the non-rigid tracking problem. Using the proposed data fitting term, the non-rigid tracking Gauss-Newton solver can converge to

the global solution even with large non-rigid motions.

- We propose ConditionNet that learns to generate a problem-specific preconditioner using a large number of training samples from the Gauss-Newton update equation. The learned preconditioner increases PCG’s convergence speed by a large margin.

## 2. Related Works

### 2.1. Classic Data-terms for Non-rigid Tracking

The core of non-rigid tracking is to define a data fitting term for robust registration. Many different data fitting terms have been proposed in the recent geometric approaches, e.g., the point-to-point alignment terms in [16], and the point-to-plane alignment terms in [20, 13]. Beside dense geometric constraints, sparse color image descriptor detection and matching have been used to establish the correspondences in [13]. In additions, in [31], the potential of color consistency assumption was studied. Furthermore, to deal with the lighting change, the reflection consistence technique was proposed in [11], and the correspondence prediction using decision trees was developed in [10].

### 2.2. Learning based tracking

This line of research focuses on solving motion tracking tasks from a deep learning perspective. One of the promising ideas is to replace the hand-engineered descriptors with the learned ones. For instance, the Learned Invariant Feature Transform (LIFT) is proposed in [28], the volumetric descriptor for 3D matching is proposed in [29], and the coplanarity descriptor for plane matching is proposed in [23]. For non-rigid localization/tracking, Schmidt *et al.* [22] use Fully-Convolutional networks to learn the dense descriptors for upper torso and head of the same person; Aljaž *et al.* [3] proposed a large labeled dataset of sparse correspondence for general non-rigidly deforming objects, and a Siamese network based non-rigid 3D patch matching approach. Regression networks have also been used to directly map input sensor data to motions, including the camera pose tracking [30], the dense optical flow tracking [9], and the 3D scene flow estimation [17]. The problem of motion regression is that the regressors could be overwhelmed by the complexity of the task, therefore, leading to severe over-fitting. A more elegant way is to let the model focus on a simple task, such as feature extraction while using classic optimization tools to solve the rest. This resulted in the recent works that combine Gauss-Newton optimization and deep learning to learn the most suitable features for image alignment [6], pose registration [12, 19], and multi-frame direct bundle-adjustment [25]. Inspired by these works, we integrate the entire non-rigid optimization method into the end-to-end pipeline to learn the optimal feature for non-rigid tracking, which requires dealing with or-

ders of magnitude more degree of freedoms than the previous cases. The details are described in Section 3.

### 2.3. Preconditioning Techniques

Preconditioning as a method of transforming a difficult problem into one that is easier to solve has centuries of history. Back to 1845, the Jacobi’s Method [5] was first proposed to improve the convergence of iterative methods. Block-Jacobi is the simplest form of preconditioning, in which the preconditioner is chosen to be the block diagonal of the linear system that we want to solve. Despite its easy accessibility, we found that applying it shows only a marginal improvement in our problem. Other methods, such as Incomplete Cholesky Factorization, multiGrid method [26] or successive over-relaxation [27] method have shown their effectiveness in many applications. In this paper, we exploit the potential of data-driven preconditioner to solve the linear system in the non-rigid tracking task. The details are shown in Section 4.

## 3. Learning Deep Non-Rigid Feature

### 3.1. Scene Representation

The input of our method is two frames that are captured using a commodity RGB-D sensor. Each frame contains a color map and a depth map both at the size of  $640 \times 480$ . Calibration was done to ensure that color and depth were aligned in temporal and spatial domain. We denote the source frame as  $\mathbf{S}$ , and the target frame as  $\mathbf{T}$ .

We approximate the surface deformation with the deformation graph  $\mathcal{G}$ . Fig. 3 shows an example of our deformation graph. We uniformly sample the image, resulting in a rectangle mesh grid of size  $w \times h$ . A point in the mesh grid is treated as a node in the deformation graph. Each node connects exactly to its 8 neighboring nodes. To filter out the invalid nodes, a binary mask  $\mathcal{V} \in \mathbb{R}^{w \times h}$  is constructed by checking if the node is from the background, holds invalid depth, or lies on occlusion boundaries with large depth discontinuity. Similarly, edges are filtered by the mask  $\mathcal{E} \in \mathbb{R}^{w \times h \times 8}$  if they link to invalid nodes or go beyond the edge length threshold. In the deformation graph, the node  $i$  is parameterized by a translation vector  $\mathbf{t}_i \in \mathbb{R}^3$  and a matrix  $\mathbf{R}_i \in \text{SO3}$ . Putting all parameters into a single vector, we get

$$\mathcal{G} = \{\mathbf{R}_i, \mathbf{t}_i |_{i=1,2,\dots,w \times h}\}$$

### 3.2. Deep Feature Fitting Term

We use the function  $\mathcal{F}(\cdot)$ , which is based on fully convolutional networks [18], to extract feature map from source frame  $\mathbf{S}$  and target frame  $\mathbf{T}$ . The encoded feature maps are:

$$\mathbf{F}_\mathbf{S} = \mathcal{F}(\mathbf{S}), \quad \mathbf{F}_\mathbf{T} = \mathcal{F}(\mathbf{T}) \quad (2)$$

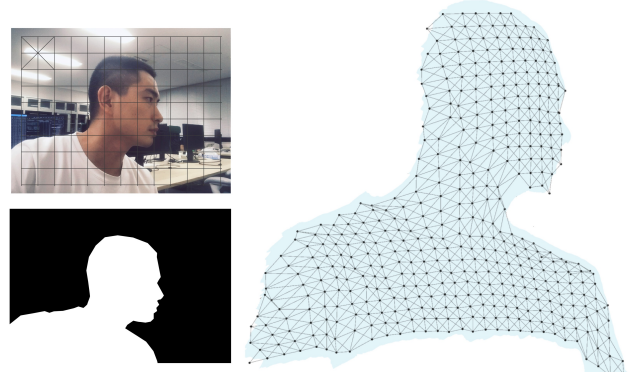


Figure 3. Our deformation graph. Left top: Uniform sampling on the pixel grid. Left bottom: Binary mask acquired using simple depth threshold or depth aided human annotation. Right: Masked 3D deformation graph.

We apply up-sampling layers in the neural network such that the encoded feature map has the size  $w \times h \times c$ , where  $c$  is the dimension of a single feature vector. Thus the feature map and the deformation graph have the same rows and columns. This means that a feature vector and a graph node have a one-to-one correspondence (to reduce GPU memory overhead and speed up the learning). We denote  $\mathbf{D}_\mathbf{S} \in \mathbb{R}^{w \times h}$  and  $\mathbf{D}_\mathbf{T} \in \mathbb{R}^{w \times h}$  as the sampled depth map from source and target frames. Given the translation vectors  $\mathbf{t}_i \in \mathcal{G}$ , and the depth value  $\mathbf{D}_\mathbf{T}(i)$ , the projected feature for the pixel  $i$  can be obtained by

$$\tilde{\mathbf{F}}_\mathbf{S}(i) = \mathbf{F}_\mathbf{S}(\pi(\mathbf{t}_i, \mathbf{D}_\mathbf{T}(i))) \quad (3)$$

where  $\pi(\cdot) : \mathbb{R}^2 \rightarrow \mathbb{R}^2$  is the warping function that maps one pixel coordinate to another pixel coordinate by applying translation  $\mathbf{t}_i$  to a back-projected pixel  $i$ , and projecting the transformed point to the source camera frame. The warped coordinate are continuous values.  $\tilde{\mathbf{F}}_\mathbf{S}(i)$  is sampled by bilinearly interpolating the 4 nearest features on the 2D mesh grid. This sampling operation is made differentiable using the spatial transformer network defined in [14]. Then the **deep feature fitting term** is defined as

$$\mathbf{E}_{fea}(\mathcal{G}) = \lambda_f \sum_{i=0}^{w \times h} \mathcal{V}_i \cdot \|\tilde{\mathbf{F}}_\mathbf{S}(i) - \mathbf{F}_\mathbf{T}(i)\|^2 \quad (4)$$

Note that compared to the classic color-consistency constraints, the learned deep feature captures high-order spatial deformations in the scans, by leveraging the large receptive field size of the convolution kernels.

### 3.3. Total Energy

To resolve the ambiguity in  $Z$  axis, we adopt a projective depth, which is a rough approximation of the point-to-plane constraint, as our **geometric fitting term**. This term measures the difference between warped depth map  $\tilde{\mathbf{D}}_\mathbf{S}$  and the

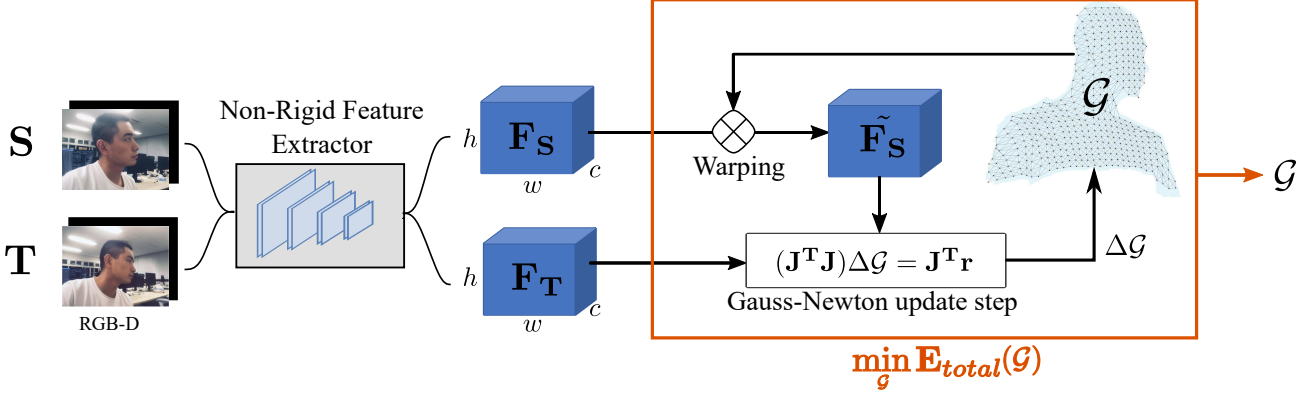


Figure 4. High-level overview of our non-rigid feature extractor training method. Jacobian  $\mathbf{J}$ 's entries for the feature term (4) can be precomputed according to the inverse composition algorithm. Other entries in Jacobian  $\mathbf{J}$  and all entries in residues  $\mathbf{r}$  are recomputed in each Gauss-Newton iteration. For simplicity, the geometric fitting term (5) and regularization term (6) are omitted from this figure.

depth map of target frame. It is defined as

$$\mathbf{E}_{geo}(\mathcal{G}) = \lambda_g \sum_{i=0}^{w \times h} \mathcal{V}_i \cdot \|\tilde{\mathbf{D}}_S(i) - \mathbf{D}_T(i)\|^2 \quad (5)$$

Finally, we regularize the shape deformation by the ARAP **regularization term**, which encourages locally rigid motions. It is defined as

$$\mathbf{E}_{reg}(\mathcal{G}) = \lambda_r \sum_{i=0}^{w \times h} \sum_{j \in \mathcal{N}_i} \mathcal{E}_{i,j} \cdot \|(\mathbf{t}_i - \mathbf{t}_j) - \mathbf{R}_i(\mathbf{t}'_i - \mathbf{t}'_j)\|^2 \quad (6)$$

Where  $\mathcal{N}_i$  denotes node- $i$ 's neighboring nodes, and  $\mathbf{t}'_j$ ,  $\mathbf{t}'_j$  are the positions of  $i$ ,  $j$  after the transformation. To summarize the above, we obtain the following energy for non-rigid tracking:

$$\mathbf{E}_{total}(\mathcal{G}) = \mathbf{E}_{fea}(\mathcal{G}) + \mathbf{E}_{geo}(\mathcal{G}) + \mathbf{E}_{reg}(\mathcal{G}) \quad (7)$$

The three terms are balanced by  $[\lambda_f, \lambda_g, \lambda_r]$ . The total energy is then optimized by the Gauss-Newton update steps:

$$(\mathbf{J}^T \mathbf{J}) \Delta \mathcal{G} = \mathbf{J}^T \mathbf{r} \quad (8)$$

where  $\mathbf{r}$  is the error residue, and  $\mathbf{J}$  is the Jacobian of the residue with respect to  $\mathcal{G}$ . This equation is further solved by the iterative PCG solver.

### 3.4. Back-Propagation Through the Two Solvers

The learning pipeline is shown in Fig. 4. We integrate all energy optimization steps into an end-to-end training pipeline. To this end, we need to make both Gauss-Newton and PCG differentiable. In the Gauss-Newton case, the update steps stop when a specified threshold is reached. Such if-else based termination criteria prevents error back-propagation. We apply the same solution as in [19, 25, 12], *i.e.*, we fix the number of Gauss-Newton iterations. In this

project, we set this number to a small digit. There are two reasons behind this: 1) For the recursive nature of the Gauss-Newton layer, large iterations number will induce instability to the network training, 2) By limiting the available step, the feature extractor is pushed to produce the features that allow Gauss-Newton solver to make bigger jumps toward the solution. Thus we can achieve faster convergence and robust solving.

Back-propagation through PCG can be done in a different fashion as described in [1]. Equation (8) need to be solved in every Gauss-Newton iteration. Let's represent  $\mathbf{J}^T \mathbf{J}$  by  $\mathbf{A}$ ,  $\Delta \mathcal{G}$  by  $\mathbf{x}$ , and  $\mathbf{J}^T \mathbf{r}$  by  $\mathbf{b}$ , then we get the following iconic equation:

$$\mathbf{A} \mathbf{x} = \mathbf{b} \quad (9)$$

Suppose that we have already got the gradient of loss  $\mathbb{L}$  w.r.t to the solution  $\mathbf{x}$  as  $\partial \mathbb{L} / \partial \mathbf{x}$ . We want to back propagate that quantity onto  $\mathbf{A}$  and  $\mathbf{b}$ :

$$\frac{\partial \mathbb{L}}{\partial \mathbf{b}} = \mathbf{A}^{-1} \frac{\partial \mathbb{L}}{\partial \mathbf{x}} \quad (10)$$

$$\frac{\partial \mathbb{L}}{\partial \mathbf{A}} = (-\mathbf{A}^{-1} \frac{\partial \mathbb{L}}{\partial \mathbf{x}}) (\mathbf{A}^{-1} \mathbf{b})^T = -\frac{\partial \mathbb{L}}{\partial \mathbf{b}} \mathbf{x}^T \quad (11)$$

which means that back-propagating through the linear system only need another PCG solve for Equation (10).

### 3.5. Training Objective & Data Acquisition

The method outputs the final deformation graph after a few Gauss-Newton iterations. We apply the L1 flow loss on all the translation vectors  $\mathbf{t}_i \in \mathcal{G}$  in the deformation graph

$$\mathbb{L}_{flow} = \sum_{\mathbf{t}_i \in \mathcal{G}} |\mathbf{t}_i - \mathbf{t}_{i,gt}| \quad (12)$$

where  $\mathbf{t}_{i,gt} \in \mathbb{R}^3$  is node- $i$ 's ground truth 3D translation vector, *i.e.*, the scene flow.



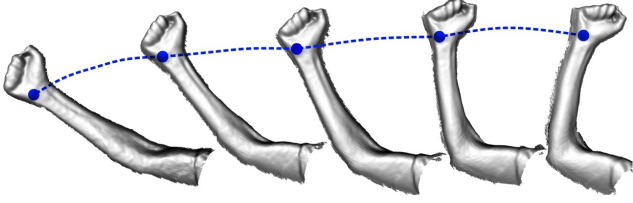


Figure 5. Using our non-rigid tracking and reconstruction method to obtain point-point correspondence. This method can generate accurate correspondence when the motion is small. The long term correspondence between distant frames can be obtained by accumulating small inter-frame motions through time and space.

Collecting  $\mathbf{t}_{i,gt}$  is a non-trivial task. Inspired by Zeng *et al.* [29] and Schmidt *et al.* [22], we realize that the 3D correspondence ground truth can be achieved by running the state-of-the-art tracking and reconstruction methods such as BundleFusion [8], for rigid scenes, or DynamicFusion [20]/VolumeDeform [13], for non-rigidly deforming scenes. For the rigid training set, we turn to the ScanNet, which contains a large number of indoor sequences with BundleFusion based camera trajectory. For the non-rigid training dataset, as shown in Fig. 5, we run our geometry based non-rigid reconstruction method (which is similar to DynamicFusion [20]) on the collected non-rigid sequences. We argue that non-rigid feature learning could benefit from rigid scenes. Since the rigid scenes can be considered as a subset of the non-rigid ones, the domain gap is not that huge when we approximate the rigid object surface from a deformable perspective. Eventually, the feature learning pipeline is pre-trained on ScanNet and fine-tuned on our non-rigid dataset.

#### 4. Data-Driven Preconditioner

Preconditioner  $\mathbf{M}^{-1}$  modifies the system  $\mathbf{Ax} = \mathbf{b}$  to

$$\mathbf{M}^{-1}\mathbf{Ax} = \mathbf{M}^{-1}\mathbf{b} \quad (13)$$

which is easier to solve. From the iterative optimization perspective, solving (13) is equal to finding the  $\mathbf{x}$  that minimizes the quadratic form

$$\min_{\mathbf{x}} \|\mathbf{M}^{-1}\mathbf{Ax} - \mathbf{M}^{-1}\mathbf{b}\|^2 \quad (14)$$

Here, we propose the ConditionNet  $\mathcal{C}(\cdot)$  based on neural networks with an encoder and decoder structure to do the mapping:

$$\mathcal{C}(\cdot) : \mathbb{R}^{n \times n} \rightarrow \mathbb{R}^{n \times n} : \mathbf{A} \rightarrow \mathbf{M}^{-1}$$

A good preconditioner should be a symmetric positive definite (SPD) matrix, otherwise, the PCG can not guarantee to converge. To this end, the ConditionNet first generates the lower triangle matrix  $\mathbf{L}$ . Then the preconditioner  $\mathbf{M}^{-1}$  is computed as

$$\mathbf{M}^{-1} = \mathbf{LL}^T \quad (15)$$

Empirically, we apply a hard positive threshold on  $\mathbf{M}^{-1}$ 's diagonal entries to combat the situations that there exist zero singular values. By doing this,  $\mathbf{M}^{-1}$  is ensured to be an SPD matrix in our case.

The matrix density, *i.e.*, the ratio of non-zero entries, play an important role in preconditioning. On one end, a denser preconditioner has a higher potential to approximate  $\mathbf{A}^{-1}$ , which is the perfect conditioner, but the matrix inverse itself is time-consuming. On the other end, a sparser matrix is cheaper to achieve while leading to a poor preconditioning effect. To examine the trade-off between efficiency and effectiveness, we propose the following three ConditionNet variants. They use the same network structure but generate preconditioners with different density, from dense to sparse.

**ConditionNet-Dense.** As shown in Fig. 6, this one uses full matrix  $\mathbf{A}$  as input and generate the dense preconditioner, in which all entries can be non-zero. Intuitively, this model is trying to approximate the perfect conditioner  $\mathbf{A}^{-1}$ .

**ConditionNet-Sparse.** This one inputs full matrix  $\mathbf{A}$ . For the output, a binary mask is applied such that any entry in  $\mathbf{L}$  is set to zero if the corresponding entry in  $\mathbf{A}$  is also zero.

**ConditionNet-Diagonal.** The input and output are the block diagonals of the matrices. There are  $w \times h$  diagonal blocks and each block is  $6 \times 6$ . Since each block is directly related to a feature in the 2D mesh grid, we reshape the input block diagonal entries to a  $[w, h, 36]$  volume to leverage such 2D spatial correlations. The output volume is  $[w, h, 21]$  for the lower triangle matrix  $\mathbf{L}$ . This model generates the sparsest preconditioner.

#### 4.1. Self-Supervised Training

The straight forward way to train the ConditionNet is to minimize the condition number  $\kappa(\mathbf{M}^{-1}\mathbf{A}) = \lambda_{\max}/\lambda_{\min}$  *i.e.*, the ratio of the maximum and minimum singular value in  $\mathbf{M}^{-1}\mathbf{A}$ . However, the time consuming singular value decomposition (SVD) makes large scale training impractical.

Instead, we propose the PCG-Loss for training. As shown in Fig. 6 the learned preconditioner  $\mathbf{M}^{-1}$  is fed to a PCG layer to minimize (14) and output the solution  $\mathbf{x}$ . Training data generation for the ConditionNet is fully automatic; *i.e.*, no annotations are needed to find the ground truth solution  $\mathbf{x}_{gt}$  to equation  $\mathbf{Ax} = \mathbf{b}$ , which we do by running a standard PCG solver. To obtain  $\mathbf{x}_{gt}$ , the standard PCG is executed as many iterations as possible till convergence. Then the L1 PCG-Loss is applied on the predicted solution

$$\mathbb{L}_{pcg} = |\mathbf{x} - \mathbf{x}_{gt}| \quad (16)$$

The training samples, *i.e.*, the  $[\mathbf{A}, \mathbf{b}]$  pairs, are collected from the Gauss-Newton update step in Eqn. (8).

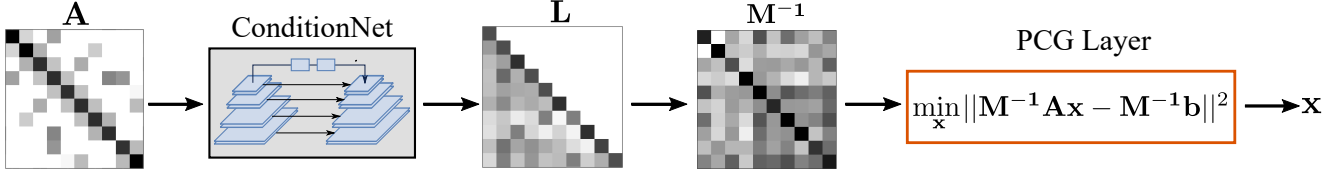


Figure 6. Overview of ConditionNet-Dense. The output  $L$  is the lower triangle matrix of the preconditioner. After a few iterations in the PCG layer, the solution  $x$  is then penalized by the L1 loss (16). The whole pipeline can be trained end-to-end.

Method	Energy Terms					ScanNet (SN)				Non-Rigid Dataset (NR)			
	ARAP	P2Plane	P2Point	Color	Feature	3D EPE (cm) on $\downarrow$ validation/test on frame jump				3D EPE (cm) on $\downarrow$ validation/test on frame jump			
						0 $\rightarrow$ 2	0 $\rightarrow$ 4	0 $\rightarrow$ 8	0 $\rightarrow$ 16	0 $\rightarrow$ 2	0 $\rightarrow$ 4	0 $\rightarrow$ 8	0 $\rightarrow$ 16
N-ICP-0	✓			✓		3.21/3.65	5.03/5.68	8.17/9.66	15.35/18.65	2.29/2.3	4.08/3.3	7.71/6.3	13.63/12.72
N-ICP-1 [20]	✓	✓	✓			2.43/2.75	4.50/5.38	8.11/9.09	14.62/17.10	<b>1.49/1.53</b>	2.98/2.71	6.61/6.52	11.14/12.07
N-ICP-2 [31]	✓	✓	✓	✓		<b>2.04/2.71</b>	3.58/4.47	6.07/7.89	10.36/14.96	1.68/1.70	3.41/2.60	5.50/5.20	11.80/10.59
Ours (SN)	✓	✓			✓	<b>2.10/2.60</b>	<b>3.55/4.39</b>	<b>5.28/6.98</b>	<b>7.34/10.59</b>	1.73/1.60	2.77/2.63	4.99/5.08	7.09/8.32
Ours (SN+NR)	✓	✓			✓	–	–	–	–	<b>1.55/1.34</b>	<b>2.25/2.23</b>	<b>4.16/4.50</b>	<b>6.47/7.59</b>

Table 1. 3D End point Error (EPE) on ScanNet and our Non-Rigid dataset. The frame jumps shows the index of the indices of the source and target frame. The number of unknowns in the deformation graph is 1152 ( $16 \times 12 \times 6$ ). Ours (SN): trained on ScanNet [7]. Ours (SN + NR): pretrained on ScanNet and fine-tuned on the Non-Rigid dataset [3].

During the training phase, we limit the number of available iterations in the PCG layer. This is to encourage the ConditionNet to generate a better preconditioner that achieves the same solution while using fewer steps. At the early phase of the training, the PCG layer with limited iterations does not guarantee a good convergence. The back-propagation strategy described in Section 3.4 can not be applied here, because incomplete solving results in wrong gradient. Instead, we directly flow the gradient through all PCG iterations for ConditionNet training.

We train ConditionNet and the non-rigid feature extractor separately. They are used together at the testing phase.

## 5. Experiments

**Implementation Details:** The resolution of the deformation graph is  $16 \times 12$ . Empirically, the weighting factor  $[\lambda_f, \lambda_g, \lambda_r]$  in the energy function (7) are set to  $[1, 0.5, 40]$ . The number of Gauss-Newton iterations is 3 for non-rigid feature extractor training. The number of PCG iterations is 10 for ConditionNet Training. We implement our networks using the publicly available Pytorch framework and train it with Tesla P100 GPUs. We trained all the models from scratch for 30 epochs, with a mini-batch size of 4 using Adam [15] optimizer, where  $\beta_1 = 0.9$ ,  $\beta_2 = 0.999$ . We used an initial learning rate of 0.0001 and halve it every 1/5 of the total iterations.

### 5.1. Datasets

**ScanNet** ScanNet [7] is a large-scale RGBD video dataset containing 1,513 sequences in 706 different scenes. The sequences are captured by iPad Mounted RGBD sensors that provide calibrated depth-color pairs of VGA resolution. The 3D camera poses are based on BundleFusion [8]. The 3D dense motion ground truth on the ScanNet is obtained by projecting point cloud via depth and 6-Dof camera pose. We apply the following filtering process for training data. To narrow the domain gap with the non-rigid dataset, we filter out images if more than 50% of the pixels have the invalid depth or depth values larger than 2 meters. To avoid image pairs with large pose error, we filter image pairs with a large photo-consistency error. Finally, we remove the image pairs with less than 50% “covisibility”, *i.e.*, the percentage of the pixels that are visible from both images. Similarly, the sequences are subsampled using the intervals [2, 4, 8, 16]. We use 60k frame pairs in total and split train/valid/test as 8/1/1.

**Non-Rigid Dataset** We use the non-rigid dataset from Aljaž *et al.* [3] which consists of 400 non-rigidly deforming scenes, over 390,000 RGB-D frames. A variety of deformable objects are captured including adults, children, bags, clothes, and animals, etc. The distance of the objects to the camera center lies in the range [0.5m, 2.0m]. Depending on the complexity of the scene, the foreground object masks are either obtained by a simple depth threshold or depth map aided human annotation. We run our tracking and reconstruction method to obtain the ground truth

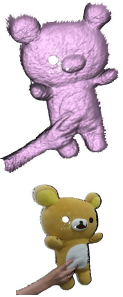

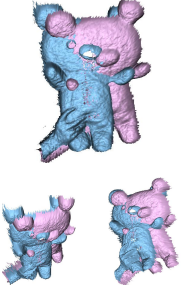
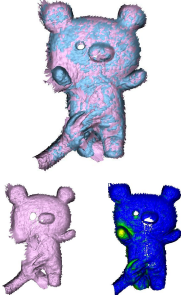


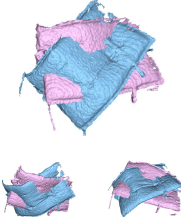
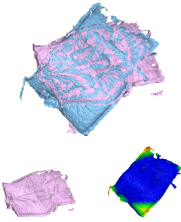
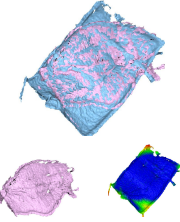
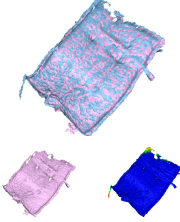



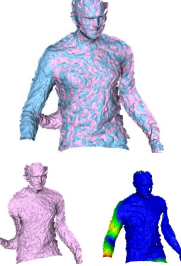
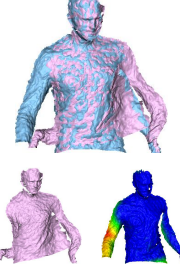
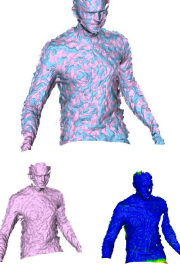


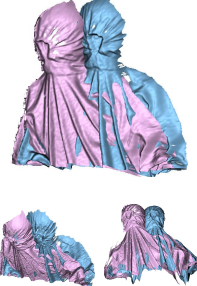
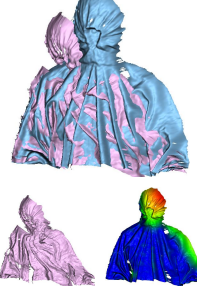


Scene	Mesh <sup>1</sup> and Masked Color Image		Initial Alignment <sup>2</sup>	Final Alignment, Transformed Source Mesh, and Alignment Error <sup>3</sup>		
	Source	Target		N-ICP-1 [20]	N-ICP-2 [31]	Ours
	<i>bear</i>					
<i>pillow</i>						
<i>adult</i>						
<i>clothes</i>						

Figure 7. Frame-Frame tracking results. <sup>1</sup> Meshes are constructed from depth images. Depth images are preprocessed by the bilateral filter to reduce observation noise. <sup>2</sup> Initial alignment is done by simply setting the camera poses of both frames to identity. <sup>3</sup> The alignment error (hotter means larger) measures the point to point distance between target mesh and the transformed source mesh.

non-rigid motions. We remove the drifted sequences by manually checking the tracking quality of the reconstructed model. The example of this dataset can be found in the paper [3]. Similarly to the rigid case, we sub-sample the sequences using the frame jumps [2, 4, 8, 16] to simulate the different magnitude of non-rigid deformation. For data-augmentation, we perform horizontal flips, random gamma, brightness, and color shifts for input frame pairs. Finally, we got 8.5k frame pairs in total and split train/valid/test as 8/1/1.

## 5.2. Non-Rigid Tracking Evaluation

**Baselines** We implement a few variants of the non-rigid ICP (N-ICP) methods. They apply different energy terms as shown in Tab. 1. Among them, N-ICP-1 is our implementation of the method DynamicFusion [20], and N-ICP-2 is our implementation for the method described in [31]. The original two papers are focusing on the model to frame tracking problems where the model is either reconstructed on-the-fly or pre-defined. Here all baselines are deployed



for the frame-frame tracking problem. Ours first optimizes the feature fitting term based objective (7) to get the coarse motion and then refine the graph with the classic point-to-plane constraints using the raw depth maps.

**Quantitative Results** The quantitative results on the ScanNet dataset and the non-rigid dataset can be found in Table 1. The estimated motions are evaluated using the 3D End-Point-Error (EPE) metric. On ScanNet, Ours(SN) achieves overall better performance than the other N-ICP baselines, especially when the motions are large (e.g., on 0→8 and 0→16 frame jump). Note that the ScanNet pre-trained model Ours(SN) even achieves better results than the classic N-ICPs on the non-rigid dataset, indicating a good generalization ability of the learned non-rigid feature, which makes sense considering that the learnable CNN model focuses only on the feature extraction part, and the using of classic optimizer disentangle the direct mapping from images to motion. It also proves the assumption that the rigid and non-rigid surfaces lie in quite close domains. The fine-tuned model Ours(SN+NR) on the non-rigid dataset further improved these numbers.

**Qualitative Results** Fig. 7 shows the frame-frame tracking results on the non-rigid frame pairs. We selected the frame pairs with relatively large non-rigid motions. N-ICP-1 and N-ICP-2 have trouble dealing with these motions and converged to bad local minimums. Our method manages to converge to the global solutions on these challenging cases. For instance, the *clothes* scene in Fig. 7 is an especially challenging case for classic non-rigid ICP methods because the point-to-plane term has no chance to slide over the zigzag clothes surface which contains multiple folds, and the color consistency term could also be easily confused by the repetitive camouflage textures of the *clothes*. The learned features show an advantage for capturing high order deformation on those cases.

### 5.3. Preconditioning Results

We randomly collected 10K [A, b] pairs from different iterations the Gauss-Newton step. We split them to train/valid/test according to the ratio of 8/1/1. We compare with 3 PCG baselines: w/o preconditioner, the standard block-diagonal preconditioner, and the Incomplete Cholesky factorization based preconditioner. We also show the ablation studies on three of the ConditionNet variants: Diagonal, Sparse and Dense. Fig. 1 shows the PCG steps using different preconditioners. The learned preconditioner outperforms the classic ones by a large margin. Tab. 2 shows PCG’s solving results using different preconditioners. All learned preconditioners significantly reduced the condition numbers. ConditionNet-Dense achieves the best convergence rate and the least overall solving time.

Preconditioner	<i>density</i>	$\kappa$	<i>iters</i>	<i>time (ms)</i>
None	–	3442.18	46	33.43
Block-Diagonal	0.46%	541.52	44	31.34
Incomplete Cholesky	1.52	379.82	37	28.42
ConditionNet-Diagonal (ours)	0.46%	93.55	21	12.38
ConditionNet-Sparse (ours)	1.52%	125.81	23	17.80
ConditionNet-Dense (ours)	100.%	<b>34.90</b>	<b>13</b>	<b>10.32</b>

Table 2. PCG solving results using different preconditioners (residue threshold of convergence:  $10^{-6}$ ). *density*: density of preconditioner.  $\kappa$ : condition number of the modified linear system. *iters*: total steps for convergence. *time(ms)*: time of solving. All numbers are obtained with Pytorch-GPU implementation.

## 6. Conclusion and Discussion

In this work, we present an end-to-end learning approach for non-rigid RGB-D tracking. Our core contribution is the learnable optimization approach which improves both robustness and convergence by a significant margin. The experimental results show that the learned non-rigid feature significantly improves the convergence of Gauss-Newton solver for the frame-frame non-rigid tracking. In addition, our method increases the PCG solver’s convergence rate by predicting a good preconditioner. Overall, the learned preconditioner requires 2 to 3 times fewer iterations until convergence.

While we believe this results are very promising and can lead to significant practical improvements in non-rigid tracking and reconstruction frameworks, there are several major challenges are yet to be addressed: 1) The proposed non-rigid feature extractor adopted plain 2D convolution kernels, which are potentially not the best option to handle 3D scene occlusions. One possible research avenue is to directly extract non-rigid features from 3D point clouds or mesh structures using the point-based architectures [21], or even graph convolutions [4]. 2) Collecting dense scene flow using DynamicFusion for real-world RGB-D video sequence is expensive (i.e., segmentation and outlier removal can become painful processes). The potential solution is learning on synthetic datasets. (e.g., using graphics simulations where the dense motion ground truth is available).

## Acknowledgements

This work was also supported by a TUM-IAS Rudolf Mößbauer Fellowship, the ERC Starting Grant *Scan2CAD* (804724), and the German Research Foundation (DFG) Grant *Making Machine Learning on Static and Dynamic 3D Data Practical*, as well as the JST CREST Grant Number JPMJCR1403, and partially supported by JSPS KAKENHI Grant Number JP19H01115. YL was supported by the Erasmus+ grant during his stay at TUM. We thank Christopher Choy, Atsuhiko Noguchi, Shunya Wakasugi, and Kohei Uehara for helpful discussion.



## References

- [1] Jonathan T Barron and Ben Poole. The fast bilateral solver. In *ECCV*, pages 617–632. Springer, 2016. 4
- [2] Åke Björck. *Numerical methods for least squares problems*. SIAM, 1996. 2
- [3] Aljaž Božič, Michael Zollhöfer, Christian Theobalt, and Matthias Nießner. Deepdeform: Learning non-rigid rgb-d reconstruction with semi-supervised data. In *CVPR*, 2020. 2, 6, 7
- [4] Michael M Bronstein, Joan Bruna, Yann LeCun, Arthur Szlam, and Pierre Vandergheynst. Geometric deep learning: going beyond euclidean data. *IEEE Signal Processing Magazine*, 34(4):18–42, 2017. 8
- [5] Jacobi Carl Gustav Jacob. Über eine neue auflösungsart der bei der methode der kleinsten quadrate vorkommenden linearen gleichungen. In *Nachrichten*, pages 22, 297, 1845. 3
- [6] Che-Han Chang, Chun-Nan Chou, and Edward Y Chang. Clkn: Cascaded lucas-kanade networks for image alignment. In *CVPR*, pages 2213–2221, 2017. 2
- [7] Angela Dai, Angel X Chang, Manolis Savva, Maciej Halber, Thomas Funkhouser, and Matthias Nießner. Scannet: Richly-annotated 3d reconstructions of indoor scenes. In *CVPR*, pages 5828–5839, 2017. 6
- [8] Angela Dai, Matthias Nießner, Michael Zollhöfer, Shahram Izadi, and Christian Theobalt. Bundlefusion: Real-time globally consistent 3d reconstruction using on-the-fly surface reintegration. *ACM Transactions on Graphics (TOG)*, 36(3):24, 2017. 5, 6
- [9] Alexey Dosovitskiy, Philipp Fischer, Eddy Ilg, Philip Hausser, Caner Hazirbas, Vladimir Golkov, Patrick Van Der Smagt, Daniel Cremers, and Thomas Brox. FlowNet: Learning optical flow with convolutional networks. In *ICCV*, pages 2758–2766, 2015. 2
- [10] Mingsong Dou, Sameh Khamis, Yury Degtyarev, Philip Davidson, Sean Ryan Fanello, Adarsh Kowdle, Sergio Orts Escolano, Christoph Rhemann, David Kim, Jonathan Taylor, et al. Fusion4d: Real-time performance capture of challenging scenes. *ACM Transactions on Graphics (TOG)*, 35(4):114, 2016. 2
- [11] Kaiwen Guo, Feng Xu, Tao Yu, Xiaoyang Liu, Qionghai Dai, and Yebin Liu. Real-time geometry, albedo, and motion reconstruction using a single rgb-d camera. *ACM Transactions on Graphics (TOG)*, 36(3):32, 2017. 2
- [12] Lei Han, Mengqi Ji, Lu Fang, and Matthias Nießner. Regnet: Learning the optimization of direct image-to-image pose registration. *arXiv preprint arXiv:1812.10212*, 2018. 2, 4
- [13] Matthias Innmann, Michael Zollhöfer, Matthias Nießner, Christian Theobalt, and Marc Stamminger. Volumedeform: Real-time volumetric non-rigid reconstruction. In *ECCV*, pages 362–379. Springer, 2016. 1, 2, 5
- [14] Max Jaderberg, Karen Simonyan, Andrew Zisserman, et al. Spatial transformer networks. In *Advances in neural information processing systems*, pages 2017–2025, 2015. 3
- [15] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014. 6
- [16] Hao Li, Robert W Sumner, and Mark Pauly. Global correspondence optimization for non-rigid registration of depth scans. In *Computer graphics forum*, volume 27, pages 1421–1430. Wiley Online Library, 2008. 1, 2
- [17] Xingyu Liu, Charles R Qi, and Leonidas J Guibas. Flownet3d: Learning scene flow in 3d point clouds. In *CVPR*, pages 529–537, 2019. 2
- [18] Jonathan Long, Evan Shelhamer, and Trevor Darrell. Fully convolutional networks for semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3431–3440, 2015. 3
- [19] Zhaoyang Lv, Frank Dellaert, James M Rehg, and Andreas Geiger. Taking a deeper look at the inverse compositional algorithm. In *CVPR*, pages 4581–4590, 2019. 2, 4
- [20] Richard A Newcombe, Dieter Fox, and Steven M Seitz. Dynamicfusion: Reconstruction and tracking of non-rigid scenes in real-time. In *CVPR*, pages 343–352, 2015. 1, 2, 5, 6, 7
- [21] Charles Ruizhongtai Qi, Li Yi, Hao Su, and Leonidas J Guibas. Pointnet++: Deep hierarchical feature learning on point sets in a metric space. In *Advances in neural information processing systems*, pages 5099–5108, 2017. 8
- [22] Tanner Schmidt, Richard Newcombe, and Dieter Fox. Self-supervised visual descriptor learning for dense correspondence. *IEEE Robotics and Automation Letters*, 2(2):420–427, 2016. 2, 5
- [23] Yifei Shi, Kai Xu, Matthias Nießner, Szymon Rusinkiewicz, and Thomas Funkhouser. Planematch: Patch coplanarity prediction for robust rgb-d reconstruction. In *ECCV*, pages 750–766, 2018. 2
- [24] Robert W Sumner, Johannes Schmid, and Mark Pauly. Embedded deformation for shape manipulation. *ACM Transactions on Graphics (TOG)*, 26(3):80, 2007. 1
- [25] Chengzhou Tang and Ping Tan. Ba-net: Dense bundle adjustment network. *arXiv preprint arXiv:1806.04807*, 2018. 2, 4
- [26] Osamu Tatebe. The multigrid preconditioned conjugate gradient method. 1993. 3
- [27] Henk A van der Vorst. High performance preconditioning. *SIAM Journal on Scientific and Statistical Computing*, 10(6):1174–1185, 1989. 3
- [28] Kwang Moo Yi, Eduard Trulls, Vincent Lepetit, and Pascal Fua. Lift: Learned invariant feature transform. In *ECCV*, pages 467–483. Springer, 2016. 2
- [29] Andy Zeng, Shuran Song, Matthias Nießner, Matthew Fisher, Jianxiong Xiao, and Thomas Funkhouser. 3dmatch: Learning local geometric descriptors from rgb-d reconstructions. In *CVPR*, pages 1802–1811, 2017. 2, 5
- [30] Huizhong Zhou, Benjamin Ummenhofer, and Thomas Brox. Deeptam: Deep tracking and mapping. In *ECCV*, pages 822–838, 2018. 2
- [31] Michael Zollhöfer, Matthias Nießner, Shahram Izadi, Christoph Rehmann, Christopher Zach, Matthew Fisher, Chenglei Wu, Andrew Fitzgibbon, Charles Loop, Christian Theobalt, et al. Real-time non-rigid reconstruction using an rgb-d camera. *ACM Transactions on Graphics (TOG)*, 33(4):156, 2014. 1, 2, 6, 7