

OcclusionFusion: Occlusion-aware Motion Estimation for Real-time Dynamic 3D Reconstruction

Wenbin Lin

Chengwei Zheng

Jun-Hai Yong*

Feng Xu*

School of Software and BNRist, Tsinghua University

lwb20@mails.tsinghua.edu.cn, {zhengcw18, xufeng2003}@gmail.com, yongjh@tsinghua.edu.cn

Abstract

RGBD-based real-time dynamic 3D reconstruction suffers from inaccurate inter-frame motion estimation as errors may accumulate with online tracking. This problem is even more severe for single-view-based systems due to strong occlusions. Based on these observations, we propose OcclusionFusion, a novel method to calculate occlusion-aware 3D motion to guide the reconstruction. In our technique, the motion of visible regions is first estimated and combined with temporal information to infer the motion of the occluded regions through an LSTM-involved graph neural network. Furthermore, our method computes the confidence of the estimated motion by modeling the network output with a probabilistic model, which alleviates untrustworthy motions and enables robust tracking. Experimental results on public datasets and our own recorded data show that our technique outperforms existing single-view-based real-time methods by a large margin. With the reduction of the motion errors, the proposed technique can handle long and challenging motion sequences. Please check out the project page for sequence results: <https://wenbin-lin.github.io/OcclusionFusion>.

1. Introduction

Dynamic 3D reconstruction has been attracting more and more attention with the development of sensing and computing techniques. It aims to reconstruct the shape, motion, and appearance of the recorded objects, and thus enables users to record, edit, animate, and play with real 3D targets for various applications, including 3D design, video games, telecommunications, virtual reality, and augmented reality.

In dynamic 3D reconstruction with RGB-D sensors,

*This work was supported by Beijing Natural Science Foundation (JQ19015), the NSFC (No.61727808, 62021002), the National Key R&D Program of China (2018YFA0704000, 2019YFB1405703) and TC190A4DA/3. This work was supported by THUIBCS, Tsinghua University and BLBCI, Beijing Municipal Education Commission. Jun-Hai Yong and Feng Xu are corresponding authors.



Figure 1. Our method can reconstruct dynamic objects in real time.

fusion-based works [5, 6, 16, 19, 30] have achieved impressive results in recent years and have become a new technological trend with several important features. Firstly, these techniques do not require geometry templates of the target objects but fuse the geometries online with the recording. Therefore, they can handle various targets, including humans, objects, and even 3D scenes [8, 44]. Secondly, they can handle both rigid and nonrigid motions without requiring class-specific motion priors. This is also important to increase the generalization capability of 3D reconstruction. Thirdly, they can be achieved in real time and with a single consumer sensor, which makes these techniques be easily used by end-users.

In single view-based solutions, there is a significant quality gap between online and offline methods. On the online side, existing methods use iterative geometry fitting [30], sparse image feature matching [19], or photometric constraints [16] to estimate object motions. However, these real-time techniques cannot give very reliable temporal correspondences, and thus the accuracy of motion estimation is limited. With the error accumulation, these techniques tend to fail to track long sequences or challenging motions. On the other hand, offline methods can build much more accurate temporal correspondences without considering the computation complexity [5, 6], which leads to much better

reconstruction results.

We propose OcclusionFusion, which fills the quality gap between online and offline methods with real-time performance (Fig. 1) and the state-of-the-art accuracy even compared to offline methods. This is accomplished by better and efficiently exploring the spatial and temporal motion priors. For single view-based 3D dynamic reconstruction, occlusion is one key obstacle as the occluded regions need to be reconstructed without motion observations. On the other hand, we know that the occluded regions do not move arbitrarily. The motion of the visible regions and the historical motion information give strong prior knowledge to constrain the motion estimation of the occluded regions. Based on this observation, we propose to train a neural network to estimate full 3D motions of whole objects including the occluded surfaces using the motions of the visible regions as well as the historical information. With the obtained full object motion between consecutive frames, we do not require either exhaust correspondence computation like DeepDeform [6] or long range correspondences between multiple frames like Bozic et al. [5]. Either of them involves heavy computation costs that hinder real-time performance.

To estimate the motion of the occluded regions, we propose to train a light-weight graph neural network. The graph neural network integrates both the motion of the visible region by the graph structure and the historical information by involving a long short-term memory (LSTM) [17] module. Recent work 4DComplete [23] predicted the motion beyond the observable regions by a 3D convolution-based neural network. However, the 3D convolution module requires high computation and memory costs, which prevents their method from achieving real-time performance.

Furthermore, we model the per graph node motion using a Gaussian distribution, which not only improves the accuracy of motion prediction but also provides a confidence to aid the reconstruction module. With the confidence, we down-weight untrustworthy motion and improve the robustness of the reconstruction system.

In summary, the contributions lie in three aspects:

- We proposed a robust real-time dynamic 3D reconstruction system with a light-weight graph neural network for full 3D motion estimation. Various results including the one on public benchmark show that our real-time system outperforms the state-of-the-art offline methods.
- The graph neural network involves LSTM structure to leverage both the spatial and temporal information to predict the full object motion accurately and efficiently.
- Per node motion confidence is estimated by modeling the predicted motion using a Gaussian distribution, which gives more information for the reconstruction system to achieve high robustness.

2. Related Work

2.1. RGB-D Based Dynamic Reconstruction

Reconstruction of non-rigid deforming objects using a single RGB-D camera is an important research area in computer vision and graphics. Free-form capture [5, 6, 16, 19, 30, 35, 36] does not assume any geometric prior and can reconstruct general non-rigid scenes.

DynamicFusion [30] is the pioneering work that ables to capture non-rigid scenes in real-time by a hierarchical node graph structure and an efficient GPU solver. However, DynamicFusion did not utilize any visual cues from color images and cannot track the target robustly. VolumeDeform [19] introduced a grid structure to represent the object deformation and used sparse SIFT [27] features to improve the registration quality. Guo et al. [16] proposed a joint geometry, albedo, and motion field optimization pipeline to obtain a higher quality. KillingFusion [35] and SobolevFusion [36] are able to handle topology changes but cannot recover space-time correspondences along the whole input sequence. DeepDeform [6] and Bozic et al. [5] leveraged learning-based correspondences to improve the ability to handle fast motions. However, the exhaust correspondences prediction process of DeepDeform, and the multi-frame matching strategy of Bozic et al. [5] are time-consuming and hinder the reconstruction system from real-time. Other methods use a multi-camera setup like Fusion4D [11] or high frame-rate depth cameras like Motion2Fusion [10] to improve the reconstruction results. However, the equipment is difficult to access by average consumers. We present a novel real-time dynamic reconstruction method that can handle challenging motion with occlusion using only a single consumer depth sensor.

2.2. Learning-based Motion Estimation

In recent years, learning-based motion estimation methods have shown great superiority compared with hand-craft image features like SIFT [27] and SURF [2] as well as hand-crafted geometry descriptors [14, 20, 32, 33, 40]. Optical flow [9, 18, 25, 38, 39] and scene flow [3, 26, 28, 42] methods achieve promising results in dense motion estimation. DeepDeform [6] trained a delicate network to estimate sparse temporal correspondences for pixels by using the corresponding patch information. Bozic et al. [5] imposed dense optical flow to the reconstruction system, which is much more efficient than DeepDeform. However, the methods mentioned above only predict the motion of the visible part. Recently, 4DComplete [23] created a synthetic animation dataset called DeformingThings4D and proposed the first approach that predicts motion beyond the observable surface. However, its 3D convolution module is computationally intensive and cannot be applied to real-time scenarios like our method.

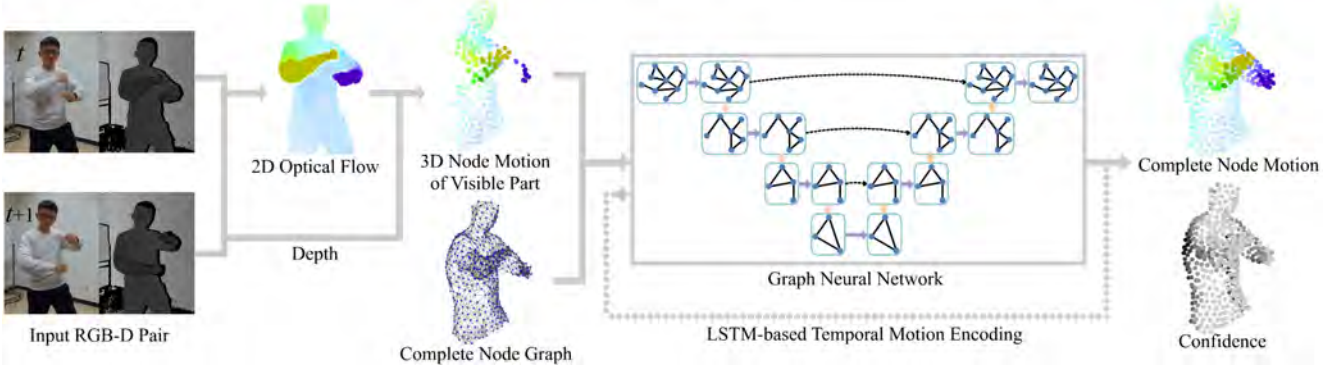


Figure 2. Pipeline of complete node motion estimation. Given input RGB-D images at frame t and $t + 1$ as well as the complete node graph at frame t , we first use a neural network to estimate the 2D optical flow between the input image pair. Then, we combine the 2D optical flow and the depth images to compute the 3D motion of visible nodes. The graph neural network uses both the motion of the visible nodes and the complete node graph as input. Meanwhile, the graph neural network integrates the historical node motion by involving an LSTM module to estimate complete node motion and per node confidence. The color encoding of optical flow is from [1].

2.3. Graph Neural Networks

Graph Neural Networks (GNNs) can model non-Euclidean data structure and have been applied to various scenarios like social networks [43], protein-protein interaction prediction [13], 3D pointcloud analysis [41], etc. Among existing works, graph convolutional networks (GCN) [21] performed spectral convolutions on graphs to propagate information among nodes. Graph U-Net [15] proposed graph pooling and unpooling operations based on trainable similarity measurements. Li et al. [22] leveraged residual connections to train deeper GNN and solve the problem of vanishing gradient and over-smoothing. Shi et al. [34] adopted multi-head attention to the message passing of graph learning. We adopt the skip connection in [22] and graph transformer module in [34] in our network. Similar to Graph U-Net, our method constructs a graph pyramid, while the feature upsampling and downsampling process of our approach is precomputed and untrainable.

3. Method

We build a single-view RGB-D based dynamic 3D reconstruction system which reconstructs a volumetric model and solves non-rigid surface motion frame-by-frame. The output motion is parameterized by the deformation of a node graph. To handle the motion tracking of the occluded nodes, we design an occlusion-aware motion estimation network (Sec. 3.1). The network infers the motion and the motion confidence of the whole node graph based on spatial and temporal motion observation. The pipeline of complete node motion estimation is demonstrated in Fig. 2. Given an input RGB-D image and the node graph of the current reconstructed model, we first estimate the 2D optical flow from the previous image to the current input image with

a neural network. Then, with the 2D optical flow and the depth images, we compute the 3D motion of the visible nodes. Both the 3D motion of the visible nodes and the complete node graph are fed to the graph neural network. Meanwhile, we involve an LSTM module to integrate the historical motion of the nodes, which carries temporal information to the graph neural network. Finally, the graph neural network predicts full node motion with per node confidence. With the predicted node motion and the confidence, we further optimize the deformation parameters of graph nodes and build a robust non-rigid reconstruction system (Sec. 3.2).

3.1. Occlusion-aware Motion Estimation Network

We use the graph-based representation proposed in [37] to parameterize object motion with sparse graph nodes. The graph nodes are uniformly sampled on the object surface. Points on the object surface are driven by the deformation of the graph nodes. However, with a single-view RGB-D camera, the motion of graph nodes is hard to solve when occlusions occur. To solve this problem, we propose an occlusion-aware motion estimator using a graph neural network.

Network Architecture. The graph neural network conducts message passing among nodes based on the node connectivity of the input node graph. As shown in Fig. 2, the goal of the graph neural network is to predict the motion of the whole node graph with per node confidence. The network’s input contains the complete graph structure, the motion of the visible nodes, and the motion of the historical frames.

Specifically, the input node feature includes the node’s 3D position and the 3D motion if the node is visible. For the occluded nodes, we assign their motion as zeros. Be-

sides, we add an extra dimension to indicate the node’s visibility. The value will be 1 for visible and 0 for occluded. In addition, we extract the rigid motion from the total motion by computing a rigid transformation in $SE(3)$ based on the motion of the visible nodes. Therefore, the neural network only needs to predict the non-rigid motion.

As the observation of a single frame is not strong enough to constrain the occluded motion, we combine the historical motion and the observation of the current frame to compute the features of the graph nodes. To integrate historical motion, we maintain a motion state for each node by involving an LSTM module. For each node, the LSTM uses the estimated historical motion and confidence to predict the node motion and confidence at the current frame.

To model 3D node motion with confidence, we use a Gaussian distribution with diagonal covariance $\mathcal{N}(\boldsymbol{\mu}, \sigma^2 \mathbf{I})$ to represent the motion of each node, where $\boldsymbol{\mu}$ is the predicted 3D motion, σ is the standard deviation used to reflect the confidence.

For message passing among nodes, we construct a multi-scale graph pyramid $\{\mathcal{G}_1, \mathcal{G}_2, \mathcal{G}_3, \mathcal{G}_4\}$ based on the input node graph. The network contains node feature downsampling, upsampling and skip connection between the same level of the pyramid like Graph U-Net [15]. The basic graph convolution module is the graph transformer proposed by Shi et al. [34]. Besides, we add a residual connection to the graph convolution module like Li et al. [22]. More details of the network can be found in the supplemental document. **Network Training.** To supervise the network training, we model the target node motion with the predicted Gaussian distribution and use a standard log-likelihood loss:

$$\mathcal{L}_{\text{out}}(\Theta_{\text{lstm}}, \Theta_{\text{graph}}) = -\frac{1}{N} \sum_{i=1}^N \log(\mathcal{N}(\mathbf{y}_i | \boldsymbol{\mu}_i, \sigma_i^2 \mathbf{I})), \quad (1)$$

where Θ_{lstm} and Θ_{graph} are the parameters of the LSTM module and the graph neural network, $\boldsymbol{\mu}_i$ and σ_i are the final outputs of the network, N is the number of nodes in the node graph, \mathbf{y}_i is the ground truth motion of the i th node.

For temporal motion encoding of the LSTM module, we impose another log-likelihood loss to the outputs of the LSTM module:

$$\mathcal{L}_{\text{temp}}(\Theta_{\text{lstm}}) = -\frac{1}{N} \sum_{i=1}^N \log(\mathcal{N}(\mathbf{y}_i | \boldsymbol{\mu}'_i, \sigma_i'^2 \mathbf{I})), \quad (2)$$

where $\boldsymbol{\mu}'_i$ and σ'_i are the outputs of the LSTM module. Besides, we find that truncating the minimum value of σ_i and σ'_i to 0.1 leads to slightly better performance.

The total loss is a weighted combination of the above two terms:

$$\mathcal{L}_{\text{total}} = \mathcal{L}_{\text{out}} + 0.1\mathcal{L}_{\text{temp}}. \quad (3)$$

3.2. Confidence Guided Non-Rigid Reconstruction

For non-rigid RGB-D reconstruction, we follow the classic volumetric pipeline that uses a truncated signed distance field (TSDF) to store the canonical model and a motion field to warp the canonical model to align it with the input image sequence.

We use the graph-based representation proposed in [37] to parameterize the non-rigid motion. The motion field \mathcal{W} can be and represented by a deformation graph $\mathcal{G} = \{\mathbf{p}_i, \mathbf{T}_i\}$, where $\mathbf{p}_i \in \mathbb{R}^3$ is the position of the i th node, $\mathbf{T}_i \in SE(3)$ is the transformation of the i th node. The motion field can be computed by convex combinations of neighboring nodes’ transformations.

Given the reconstructed model \mathcal{M}^{t-1} and the input RGB-D image pair $\{\mathcal{C}^{t-1}, \mathcal{D}^{t-1}\}$ and $\{\mathcal{C}^t, \mathcal{D}^t\}$, we optimize the following energy to solve \mathcal{W}^t .

$$E_{\text{total}}(\mathcal{W}^t) = \lambda_{\text{depth}} E_{\text{depth}} + \lambda_{\text{motion}} E_{\text{motion}} + \lambda_{2D} E_{2D} + \lambda_{\text{reg}} E_{\text{reg}}. \quad (4)$$

For the depth term E_{depth} , we employ dense point-to-plane alignment between depth image \mathcal{D}^t and the warped model:

$$E_{\text{depth}} = \sum_{(\mathbf{v}, \mathbf{u}^t) \in \mathcal{P}} (\mathbf{n}_{\mathbf{u}^t}^T (\mathbf{v}' - \mathbf{u}^t))^2, \quad (5)$$

where \mathbf{v} is a vertex on canonical model \mathcal{M} , \mathbf{v}' is the warped vertex given by \mathcal{W}^t . \mathbf{u}^t is a 3D point projected from the depth image \mathcal{D}^t , its normal is represented as $\mathbf{n}_{\mathbf{u}^t}$. \mathcal{P} is the vertex pair set. We first render the warped model \mathcal{M}^{t-1} to camera view at $t-1$ and get the projected 2D coordinate of all visible vertices. Let Π be the projection function, for $\Pi(\mathbf{v}')$, we use the computed 2D optical flow to find the corresponding pixel where $\Pi(\mathbf{u}^t)$ locates.

Based on the output motion and confidence of the neural network, we constrain the motion field to be close to the network’s prediction and construct a motion energy term as follow:

$$E_{\text{motion}} = \sum_{i \in \mathcal{G}} w_i \|\mathbf{T}_i^t \mathbf{p}_i - (\mathbf{T}_i^{t-1} \mathbf{p}_i + \boldsymbol{\mu}_i)\|_2^2, \quad (6)$$

where $\boldsymbol{\mu}_i$ denotes the motion predicted by the neural network at the i th node, w_i is the weight computed through σ_i and $\boldsymbol{\mu}_i$:

$$w_i = \exp\left(-\frac{k\sigma_i^2}{(\|\boldsymbol{\mu}_i\|_2 + \epsilon)^2}\right). \quad (7)$$

Given the dense correspondences between consecutive images, the 2D term constrains the projection of warped vertices to be consistent with the correspondences of the 2D optical flow:

$$E_{2D} = \sum_{(\mathbf{v}, \mathbf{u}^t) \in \mathcal{P}} \|\Pi(\mathbf{v}') - \Pi(\mathbf{u}^t)\|_2^2. \quad (8)$$

E_{reg} is the as-rigid-as-possible regularizer,

$$E_{\text{reg}} = \sum_{j \in \mathcal{G}} \sum_{i \in N_j} \|T_j^t p_i - T_i^t p_j\|_2^2, \quad (9)$$

where N_j denotes the neighboring nodes of the j th node.

After solving the motion field \mathcal{W}^t , we update the canonical TSDF volume. If there is a region beyond the coverage of the current node graph, new nodes will be inserted. For details on geometry fusion and node graph extension, please refer to DynamicFusion [30].

3.3. Implementation Details

Multi-scale Graph Pyramid Construction. Given a node graph sampled on the object surface, we construct a multi-scale graph pyramid by subsampling the nodes with different node intervals. The nodes in the $(l + 1)$ th level are a subset of the nodes in the l th level. The higher the level, the larger the interval. As for the edges between nodes, we compute k -nearest neighbors of each node in the Euclidean space at the first level. Then we use the breadth-first search on the graph to compute neighbors for higher levels. More details can be found in the supplemental document.

Data Generation. Network training relies on the supervision of the full 3D motion of objects. We leverage the DeformingThings4D [23] dataset, which contains 1,972 animations spanning 31 categories of humanoids and animals.

To simulate the 3D reconstruction scenario, we first uniformly sample graph nodes on the object’s surface. Then, we introduce a virtual camera that keeps the moving object in the center of the rendered depth image. Once a graph node is visible to the camera, the node will be added to an observed node set \mathcal{N} . Therefore, the number of the observed nodes will increase over time. For frame t we build a graph pyramid of the current observed node set \mathcal{N}^t . Based on the nodes’ visibility, \mathcal{N}^t can be divided into $\mathcal{N}_{\text{vis}}^t$ and $\mathcal{N}_{\text{occ}}^t$. For each node, we compute the 3D displacement between temporally adjacent frames as its motion, and only the motion of $\mathcal{N}_{\text{vis}}^t$ is fed to the network. In addition, we randomly resize the objects into a 3D bounding box ranging from $1\text{m} \times 1\text{m} \times 1\text{m}$ to $2\text{m} \times 2\text{m} \times 2\text{m}$, which is a common volume for dynamic reconstruction systems.

3D Reconstruction System. In the 3D reconstruction system, we leverage state-of-the-art 2D optical flow prediction method RAFT [39] to estimate the optical flow between the input image pairs. As the input images contain depth information, we retrain the RAFT network with RGB-D four-channel input on synthetic optical flow dataset Sintel [7], FlyingThings3D [29], and Monkaa [29]. The extra depth channel helps to achieve higher accuracy, especially when motion blur occurs in the color images. The choice of 2D optical flow methods is discussed in the supplemental document. Then, with the help of depth images, we compute the visibility of graph nodes and backproject the 2D image

Method	EPE(mm)
Rigid Fitting	15.47
ARAP Deformation [37]	4.39
Ours w/o temporal motion	4.02
Ours w/o log-likelihood loss	4.07
Ours	3.75
Ours + post-processing	3.28

Table 1. Motion estimation results of the occluded nodes.

	Method	Geo. error	Def. error
Online	DynamicFusion [30]	1.078	6.179
	VolumeDeform [19]	7.485	20.841
	Ours w/o motion term	0.406	3.154
	Ours	0.386	2.800
Offline	DeepDeform [6]	0.416	3.152
	Bozic et al. [5]	0.403	2.872

Table 2. Results on the DeepDeform [6] non-rigid reconstruction benchmark. The geometry error evaluates the difference between the reconstructed model and the input depth image. The deformation error evaluates the consistency between the motion tracking results and manually annotated correspondences. All values are in cm.

coordinate to 3D space to obtain the 3D motion of the visible nodes. For the 3D motion of the historical frames, we use the output of the 3D reconstruction system rather than the network’s output, as it integrates more constraints and is usually more accurate.

4. Experiments

In this section, we first present the performance and parameters of our system (Sec. 4.1). Then, we evaluate the accuracy of the occlusion-aware motion prediction network (Sec. 4.2). Next, we evaluate the dynamic reconstruction system quantitatively and qualitatively (Sec. 4.3). Finally, in our accompanying video, we show sequence results to fully demonstrate the power of our technique.

4.1. Performance and Parameters

Our system is implemented on a computer with an Intel i7-10700 CPU, 32 GB RAM, and two NVIDIA RTX GeForce 2080Ti graphics cards. We use an Azure Kinect DK to record RGB-D sequences at 30 fps. The depth and color images are aligned and resized to 640×480 . The full pipeline runs in real time at about 36ms per frame, where the neural network-based motion prediction takes 13ms, warp field solving takes 16ms, and 7ms for model update, marching cubes, and anything else. The 2D optical flow estimation step takes about 26ms on another graphics card.



Figure 3. Qualitative comparisons with DynamicFusion [30] and Guo et al. [16] on the DeepDeform dataset.

Since we have made the two steps in a pipeline, our system achieves real-time performance but with a one-frame delay.

The weights in Eq. 4 are empirically set as $\lambda_{\text{depth}} = 1$, $\lambda_{\text{motion}} = 2$, $\lambda_{2D} = 1e - 6$, $\lambda_{\text{reg}} = 5$. k and ϵ in Eq. 7 are set to 4 and 1cm.

4.2. Experiments of Motion Prediction

We evaluate the accuracy of motion prediction on the DeformingThings4D [23] dataset. Given a node set \mathcal{N} , its subset of visible nodes of a given camera viewpoint, and the 3D motion of the visible nodes, the goal is to estimate the motion of the occluded nodes. Following 4DComplete [23], we use rigid fitting and as-rigid-as-possible (ARAP) deformation as baselines to compare with ours. The rigid fitting approach estimates the occluded motion by a global rigid transformation estimated by the visible motion. The ARAP deformation approach leverages local rigidity and the motion of visible nodes to optimize the occluded motion. Here we use a graph-based ARAP deformation proposed by [37]. For evaluation, we randomly choose 100 animations from the animal motion subset, which are never used during training, and we apply the average 3D end-point-error (EPE) over all 100 sequences as the metric.

Comparisons. The results are reported in Tab. 1. Rigid fitting gives the worst result because there are large non-



Figure 4. Qualitative comparisons with DeepDeform [6] and Bozic et al. [5] on the DeepDeform dataset. Results of DeepDeform and Bozic et al. [5] come from the video of Bozic et al. [5].

rigid motions in the test sequences. ARAP deformation cannot give good results either because it only considers local rigidity priors. On the other hand, we achieve the best performance because we explore multi-scale motion priors through the graph pyramid structure. Like 4DComplete, we find that employing optimization-based post-processing with ARAP prior further improves the results.

Evaluations. Furthermore, we train the network without the temporal motion integrated by the LSTM, and the error increases, which verifies the effectiveness of the temporal motion. In addition, we use mean-squared error (MSE) loss instead of log-likelihood loss based on per node Gaussian distribution to train the network, and the error is also larger. In this case, the MSE loss is equivalent to the log-likelihood loss with a fixed σ for every node. As the motion uncertainties of the visible nodes and the occluded nodes are obviously different, we believe using per node Gaussian distribution to model the node motion is more appropriate. Note that the network is trained only with humanoid motions. It still performs well on animal sequences, which shows the strong generalization ability of the network.

Since the data preprocessing and network training of 4DComplete [23] are not yet publicly available, we are not able to make a direct comparison with their method.

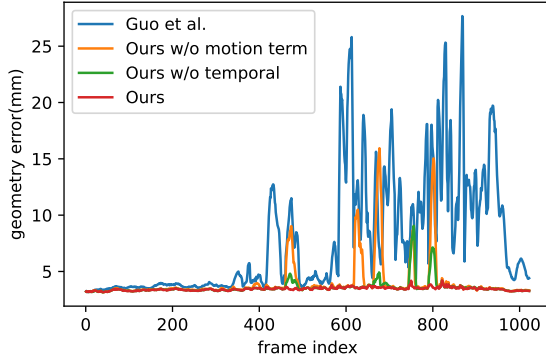


Figure 5. Geometry errors on the top sequence of Fig. 6. The average geometry errors over the whole sequence are 7.68mm, 4.01mm, 3.58mm, and 3.45mm from Guo et al. [16] to Ours.

According to the results in Tab. 1, our approach outperforms ARAP deformation by a large margin, which is similar to 4DComplete. However, our light-weight neural network takes only 13ms for motion and confidence prediction, while 4DComplete takes approximate 3s and cannot predict confidence, which is important guidance for 3D dynamic reconstruction as indicated in our experiments (Sec 4.3.2).

4.3. Experiments of Dynamic Reconstruction

We evaluate our method on the public non-rigid reconstruction benchmark of the DeepDeform [6] dataset. We first compare our method with the state-of-the-art of both online and offline solutions. Then we perform ablation studies to evaluate our key components. Finally, we show more reconstruction results of our system.

4.3.1 Comparisons

The results are shown in Tab. 2. We can see that there is a big accuracy gap between the existing online [19, 30] and offline [5, 6] methods, indicating it is difficult to improve the accuracy in the online scenario. With the help of the light-weight full motion estimation and confidence prediction, our method fills this gap and even outperforms the state-of-the-art offline methods [5, 6] with real-time performance.

As Guo et al. [16] did not evaluate errors on the DeepDeform [6] non-rigid reconstruction benchmark, we conduct a numerical comparison on a sequence from the DeepDeform dataset on our own. The geometry error of all the frames in the sequence is shown in Fig. 5. Our approach keeps the geometry error low throughout the sequence, while Guo et al. [16] has difficulty tracking the fast motion in the second half of the sequence.

For qualitative evaluation, we show comparisons with online methods DynamicFusion [30] and Guo et al. [16] in Fig. 3 and offline methods DeepDeform [6] and Bozic et

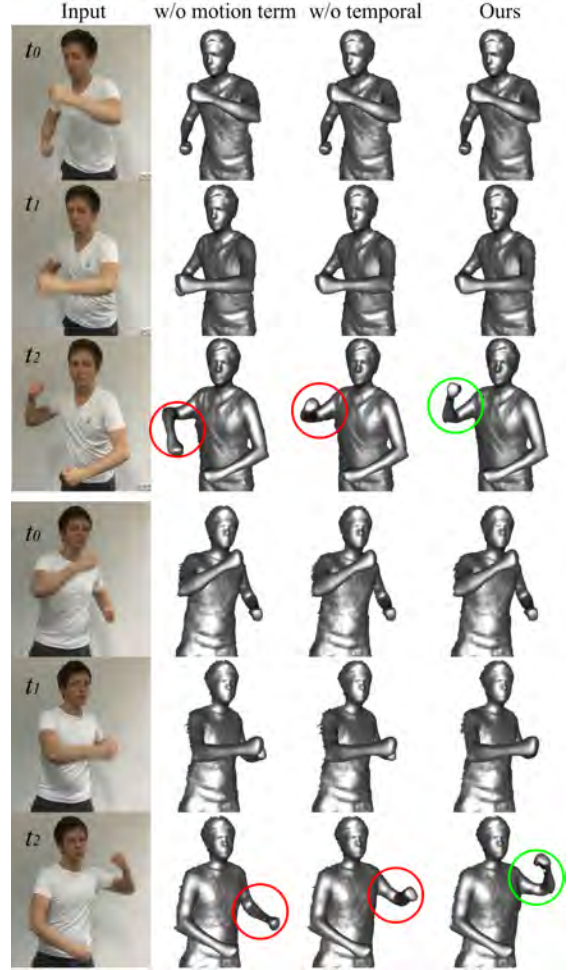


Figure 6. Ablation study on E_{motion} and temporal information. Both sequences are from the DeepDeform dataset.

al. [5] in Fig. 4. Our method handles faster motion much better than the online methods and aligns better with the input images than the offline methods.

4.3.2 Ablation Studies

We evaluate three key components of our technique regarding their contributions to the 3D dynamic reconstruction.

Complete Node Motion. We remove the complete node motion constrain E_{motion} and show qualitative reconstruction results in Fig. 6. In each group of the results, the reconstructed models at t_0 are almost the same. After a few frames, the reconstruction results with the E_{motion} term are significantly better than the unused ones. Specifically, the body parts occluded at t_1 fail to be tracked by the method without E_{motion} term. Moreover, the geometry error curve shown in Fig. 5 indicates that without the E_{motion} term, the geometry error rises frequently, which is majorly caused by occlusions. For more comparison, the results in Tab. 2 show

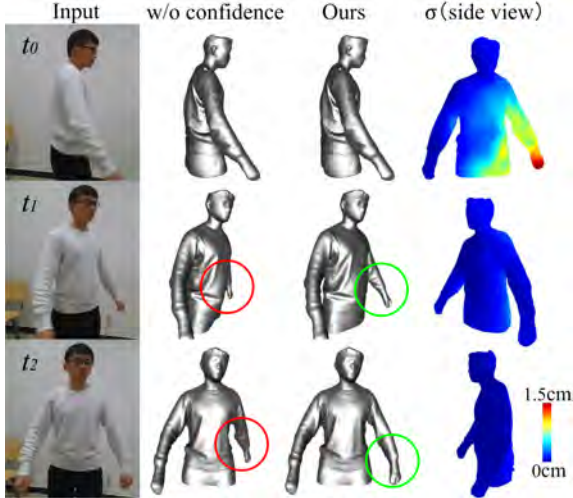


Figure 7. Ablation study on confidence weights.

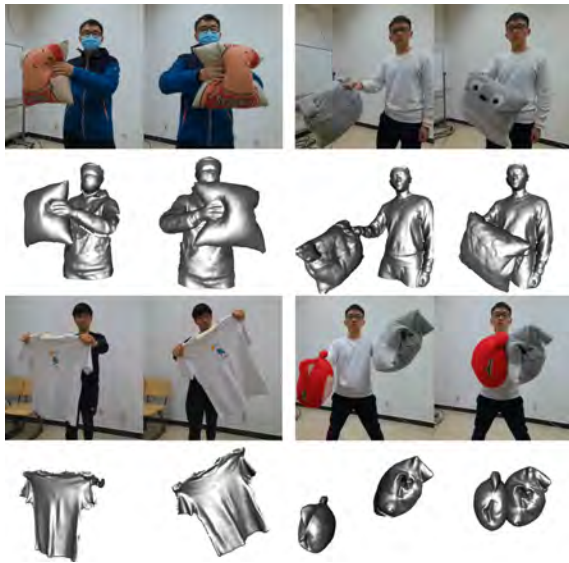


Figure 8. More reconstruction results of our method.

that removing E_{motion} term leads to higher reconstruction error. These comparisons verify that the 3D full motion helps the reconstruction system track the occluded regions.

Temporal Information. We investigate the effectiveness of the temporal information used in the graph neural network, and show the comparisons in Fig. 6. The body parts occluded at t_1 are not well tracked without the temporal information. When a moving arm is largely occluded, with only the motion of the visible upper arm, it is still hard to estimate the motion accurately. Involving the historical motion helps to improve the motion prediction. Besides, results in Fig. 5 show that removing the temporal information also brings higher geometry error.

Per Node Confidence. To show the importance of the per

node confidence, we visualize the predicted standard deviation σ on the reconstructed object’s surface in the side view (from right to left) and the corresponding results in Fig. 7. At t_0 , the left side of the body is occluded, and the predicted σ of this region is significantly higher than that of the visible region, which indicates the corresponding motion is more uncertain. As the left side of the body keeps occluded in a few frames, the inaccurate motion causes error to accumulate over time in the results without considering the confidence. The method with per node confidence down-weights the untrustworthy motion and brings better reconstruction results.

4.3.3 More Results

We show more reconstruction results of our method in Fig. 8. Ours reconstruction system is able to handle various types of targets because the neural network is trained on a dataset that includes a rich set of non-rigid objects.

5. Limitations

Although our method improves the quality of motion tracking by introducing a graph-based full motion prediction network, there are still some failure cases. First, our method cannot handle topology change, which is an open problem for node-graph-based reconstruction systems. Incorrect connections in the deformation graph can lead to tracking failures. Possible solutions could be leveraging the advantage of tracking-free reconstruction methods like POSEFusion [24], or using neural networks to predict the positions and connections of the graph nodes like Bozic et al. [4]. Besides, if a part of the object is occluded for a long time and has large non-rigid motion, it is hard for our method to re-track it when it becomes visible again.

6. Conclusion

We propose OcclusionFusion, a single-view RGB-D based real-time dynamic 3D reconstruction method that outperforms the current online techniques by a large margin. We use a complete 3D motion field to guide the object reconstruction and tracking, where the motion of the occluded regions is estimated online by a pre-trained lightweight graph neural network. The graph neural network combines the motion of visible regions and temporal information by involving an LSTM structure to accurately and efficiently predict the complete object motion. Moreover, the graph neural network predicts confidence together with the motion by modeling the network output as a Gaussian distribution, which effectively enhances the robustness of the reconstruction system. Experimental results show that long and challenging sequences can be tracked well in real time using our technique with a single view input.

References

- [1] Simon Baker, Daniel Scharstein, J. P. Lewis, Stefan Roth, Michael J. Black, and Richard Szeliski. A database and evaluation methodology for optical flow. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 1–8, 2007. 3
- [2] Herbert Bay, Andreas Ess, Tinne Tuytelaars, and Luc Van Gool. Speeded-up robust features (SURF). *Comput. Vis. Image Underst.*, 110(3):346–359, 2008. 2
- [3] Aseem Behl, Despoina Paschalidou, Simon Donné, and Andreas Geiger. Pointflownet: Learning representations for rigid motion estimation from point clouds. In *Proceedings of the IEEE Conference on computer Vision and Pattern Recognition (CVPR)*, pages 7962–7971, 2019. 2
- [4] Aljaz Bozic, Pablo Palafox, Michael Zollhofer, Justus Thies, Angela Dai, and Matthias Niessner. Neural deformation graphs for globally-consistent non-rigid reconstruction. In *Proceedings of the IEEE Conference on computer Vision and Pattern Recognition (CVPR)*, pages 1450–1459, June 2021. 8
- [5] Aljaz Bozic, Pablo R. Palafox, Michael Zollhöfer, Angela Dai, Justus Thies, and Matthias Nießner. Neural non-rigid tracking. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2020. 1, 2, 5, 6, 7
- [6] Aljaz Bozic, Michael Zollhöfer, Christian Theobalt, and Matthias Nießner. Deepdeform: Learning non-rigid RGB-D reconstruction with semi-supervised data. In *Proceedings of the IEEE Conference on computer Vision and Pattern Recognition (CVPR)*, pages 7000–7010, 2020. 1, 2, 5, 6, 7
- [7] D. J. Butler, J. Wulff, G. B. Stanley, and M. J. Black. A naturalistic open source movie for optical flow evaluation. In *European Conference on Computer Vision (ECCV)*, pages 611–625, Oct. 2012. 5
- [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 Trans. Graph.*, 36(3):24:1–24:18, 2017. 1
- [9] Alexey Dosovitskiy, Philipp Fischer, Eddy Ilg, Philip Häusser, Caner Hazirbas, Vladimir Golkov, Patrick van der Smagt, Daniel Cremers, and Thomas Brox. FlowNet: Learning optical flow with convolutional networks. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 2758–2766, 2015. 2
- [10] Mingsong Dou, Philip L. Davidson, Sean Ryan Fanello, Sameh Khamis, Adarsh Kowdle, Christoph Rhemann, Vladimir Tankovich, and Shahram Izadi. Motion2fusion: real-time volumetric performance capture. *ACM Trans. Graph.*, 36(6):246:1–246:16, 2017. 2
- [11] Mingsong Dou, Sameh Khamis, Yury Degtyarev, Philip L. Davidson, Sean Ryan Fanello, Adarsh Kowdle, Sergio Orts-Escolano, Christoph Rhemann, David Kim, Jonathan Taylor, Pushmeet Kohli, Vladimir Tankovich, and Shahram Izadi. Fusion4d: real-time performance capture of challenging scenes. *ACM Trans. Graph.*, 35(4):114:1–114:13, 2016. 2
- [12] Matthias Fey and Jan E. Lenssen. Fast graph representation learning with PyTorch Geometric. In *ICLR Workshop on Representation Learning on Graphs and Manifolds*, 2019. 11
- [13] Alex M Fout. *Protein interface prediction using graph convolutional networks*. PhD thesis, Colorado State University, 2017. 3
- [14] Andrea Frome, Daniel Huber, Ravi Kolluri, Thomas Bülow, and Jitendra Malik. Recognizing objects in range data using regional point descriptors. In *European Conference on Computer Vision (ECCV)*, volume 3023, pages 224–237, 2004. 2
- [15] Hongyang Gao and Shuiwang Ji. Graph u-nets. In *Proceedings of the 36th International Conference on Machine Learning*, volume 97, pages 2083–2092, 2019. 3, 4
- [16] 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 Trans. Graph.*, 36(3):32:1–32:13, 2017. 1, 2, 6, 7
- [17] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9:1735–80, 12 1997. 2
- [18] Eddy Ilg, Nikolaus Mayer, Tonmoy Saikia, Margret Keuper, Alexey Dosovitskiy, and Thomas Brox. FlowNet 2.0: Evolution of optical flow estimation with deep networks. In *Proceedings of the IEEE Conference on computer Vision and Pattern Recognition (CVPR)*, pages 1647–1655, 2017. 2
- [19] Matthias Innmann, Michael Zollhöfer, Matthias Nießner, Christian Theobalt, and Marc Stamminger. Volumedeform: Real-time volumetric non-rigid reconstruction. In *European Conference on Computer Vision (ECCV)*, volume 9912, pages 362–379, 2016. 1, 2, 5, 7
- [20] Andrew Edie Johnson and Martial Hebert. Using spin images for efficient object recognition in cluttered 3d scenes. *IEEE Trans. Pattern Anal. Mach. Intell.*, 21(5):433–449, 1999. 2
- [21] Thomas N. Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. In *International Conference on Learning Representations, ICLR*, 2017. 3
- [22] Guohao Li, Matthias Mueller, Guocheng Qian, Itzel Carolina Delgado Perez, Abdullellah Abualshour, Ali Kassem Thabet, and Bernard Ghanem. Deepgcns: Making gcns go as deep as cnns. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pages 1–1, 2021. 3, 4
- [23] Yang Li, Hikari Takehara, Takafumi Taketomi, Bo Zheng, and Matthias Nießner. 4dcomplete: Non-rigid motion estimation beyond the observable surface. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 12706–12716, October 2021. 2, 5, 6
- [24] Zhe Li, Tao Yu, Zerong Zheng, Kaiwen Guo, and Yebin Liu. Posefusion: Pose-guided selective fusion for single-view human volumetric capture. In *Proceedings of the IEEE Conference on computer Vision and Pattern Recognition (CVPR)*, pages 14162–14172, June 2021. 8
- [25] Pengpeng Liu, Michael R. Lyu, Irwin King, and Jia Xu. Self-low: Self-supervised learning of optical flow. In *Proceedings of the IEEE Conference on computer Vision and Pattern Recognition (CVPR)*, pages 4571–4580, 2019. 2

- [26] Xingyu Liu, Charles R. Qi, and Leonidas J. Guibas. FlowNet3d: Learning scene flow in 3d point clouds. In *Proceedings of the IEEE Conference on computer Vision and Pattern Recognition (CVPR)*, pages 529–537, 2019. [2](#)
- [27] David G. Lowe. Distinctive image features from scale-invariant keypoints. *Int. J. Comput. Vis.*, 60(2):91–110, 2004. [2](#)
- [28] Wei-Chiu Ma, Shenlong Wang, Rui Hu, Yuwen Xiong, and Raquel Urtasun. Deep rigid instance scene flow. In *Proceedings of the IEEE Conference on computer Vision and Pattern Recognition (CVPR)*, pages 3614–3622, 2019. [2](#)
- [29] Nikolaus Mayer, Eddy Ilg, Philip Häusser, Philipp Fischer, Daniel Cremers, Alexey Dosovitskiy, and Thomas Brox. A large dataset to train convolutional networks for disparity, optical flow, and scene flow estimation. In *Proceedings of the IEEE Conference on computer Vision and Pattern Recognition (CVPR)*, pages 4040–4048, 2016. [5](#)
- [30] Richard A. Newcombe, Dieter Fox, and Steven M. Seitz. Dynamicfusion: Reconstruction and tracking of non-rigid scenes in real-time. In *Proceedings of the IEEE Conference on computer Vision and Pattern Recognition (CVPR)*, pages 343–352, 2015. [1](#), [2](#), [5](#), [6](#), [7](#)
- [31] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Köpf, Edward Z. Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems (NeurIPS)*, pages 8024–8035, 2019. [11](#)
- [32] Radu Bogdan Rusu, Nico Blodow, and Michael Beetz. Fast point feature histograms (FPFH) for 3d registration. In *IEEE International Conference on Robotics and Automation, ICRA*, pages 3212–3217, 2009. [2](#)
- [33] Radu Bogdan Rusu, Nico Blodow, Zoltan Csaba Marton, and Michael Beetz. Aligning point cloud views using persistent feature histograms. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 3384–3391, 2008. [2](#)
- [34] Yunsheng Shi, Zhengjie Huang, Shikun Feng, Hui Zhong, Wenjing Wang, and Yu Sun. Masked label prediction: Unified message passing model for semi-supervised classification. In *Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence (IJCAI-21)*, pages 1548–1554, 2021. [3](#), [4](#), [11](#)
- [35] Miroslava Slavcheva, Maximilian Baust, Daniel Cremers, and Slobodan Ilic. Killingfusion: Non-rigid 3d reconstruction without correspondences. In *Proceedings of the IEEE Conference on computer Vision and Pattern Recognition (CVPR)*, pages 5474–5483, 2017. [2](#)
- [36] Miroslava Slavcheva, Maximilian Baust, and Slobodan Ilic. Sobolevfusion: 3d reconstruction of scenes undergoing free non-rigid motion. In *Proceedings of the IEEE Conference on computer Vision and Pattern Recognition (CVPR)*, pages 2646–2655, 2018. [2](#)
- [37] Robert W. Sumner, Johannes Schmid, and Mark Pauly. Embedded deformation for shape manipulation. *ACM Trans. Graph.*, 26(3):80, 2007. [3](#), [4](#), [5](#), [6](#)
- [38] Deqing Sun, Xiaodong Yang, Xiaodong Yang, Ming-Yu Liu, and Jan Kautz. Pwc-net: Cnns for optical flow using pyramid, warping, and cost volume. In *Proceedings of the IEEE Conference on computer Vision and Pattern Recognition (CVPR)*, pages 8934–8943, 2018. [2](#), [12](#)
- [39] Zachary Teed and Jia Deng. RAFT: recurrent all-pairs field transforms for optical flow. In *European Conference on Computer Vision (ECCV)*, volume 12347, pages 402–419, 2020. [2](#), [5](#), [11](#)
- [40] Federico Tombari, Samuele Salti, and Luigi di Stefano. Unique signatures of histograms for local surface description. In *European Conference on Computer Vision (ECCV)*, volume 6313, pages 356–369, 2010. [2](#)
- [41] Yue Wang, Yongbin Sun, Ziwei Liu, Sanjay E. Sarma, Michael M. Bronstein, and Justin M. Solomon. Dynamic graph CNN for learning on point clouds. *ACM Trans. Graph.*, 38(5):146:1–146:12, 2019. [3](#)
- [42] Zirui Wang, Shuda Li, Henry Howard-Jenkins, Victor Adrian Prisacariu, and Min Chen. FlowNet3d++: Geometric losses for deep scene flow estimation. In *IEEE Winter Conference on Applications of Computer Vision*, pages 91–98, 2020. [2](#)
- [43] Yongji Wu, Defu Lian, Yiheng Xu, Le Wu, and Enhong Chen. Graph convolutional networks with markov random field reasoning for social spammer detection. In *Proceedings of the AAAI Conference on Artificial Intelligence*, pages 1054–1061, 2020. [3](#)
- [44] Hao Zhang and Feng Xu. Mixedfusion: Real-time reconstruction of an indoor scene with dynamic objects. *IEEE Trans. Vis. Comput. Graph.*, 24(12):3137–3146, 2018. [1](#)

A. Network Details

We first show the details in the network architecture, including the LSTM-based temporal motion encoding and the graph neural network. Then, we show the details in the network training. The neural network is implemented using the PyTorch [31] and PyTorch Geometric [12] libraries.

A.1. Architecture Details

LSTM-based Temporal Motion Encoding. The input and output motion of the LSTM module are represented using Gaussian distribution with diagonal covariance $\mathcal{N}(\boldsymbol{\mu}, \sigma^2 \mathbf{I})$, where $\boldsymbol{\mu} \in \mathbb{R}^3, \sigma \in \mathbb{R}$. The LSTM module is a standard two-layer LSTM with a hidden feature dimension of 32. Then we use a fully connected layer to predict the $\boldsymbol{\mu}'$ and σ' based on the output feature of the LSTM.

Graph Neural Network. We show the architecture of the graph neural network in Fig. 9 and the architectures of the graph pyramid convolution and the GConv block in Fig. 10 and Fig. 11. The graph transformer is proposed by Shi et al. [34]. For all the dropout blocks, the drop probability is 0.1. N_i denotes the number of nodes in the i th level of the graph pyramid. The dimensionality of the input node feature is 11. It contains three dimensions of node position, three dimensions of node motion of the current frame, one dimension of the visibility, and four dimensions of the output $\boldsymbol{\mu}'$ and σ' from the LSTM module. Then the output motion vectors ($\boldsymbol{\mu}$ and σ) of the graph neural network will be used as the input historical motion vectors for future frames.

A.2. Training Details

Loss Function. The log-likelihood loss based on per node Gaussian distribution can be transformed as follow:

$$\begin{aligned} \mathcal{L} &= -\frac{1}{N} \sum_{i=1}^N \log(\mathcal{N}(\mathbf{y}_i | \boldsymbol{\mu}_i, \sigma_i^2 \mathbf{I})) \\ &= \frac{1}{2N} \sum_{i=1}^N \left(\log 2\pi + \log \sigma_i + \frac{\|\mathbf{y}_i - \boldsymbol{\mu}_i\|_2^2}{\sigma_i^2} \right) \quad (10) \\ &= C_1 \sum_{i=1}^N \left(\log \sigma_i + \frac{\|\mathbf{y}_i - \boldsymbol{\mu}_i\|_2^2}{\sigma_i^2} \right) + C_2, \end{aligned}$$

where C_1, C_2 are constants. Therefore the loss function can be simplified as:

$$\mathcal{L} = \frac{1}{N} \sum_{i=1}^N \left(\log \sigma_i + \frac{\|\mathbf{y}_i - \boldsymbol{\mu}_i\|_2^2}{\sigma_i^2} \right). \quad (11)$$

Thus, if the σ_i is a fixed value, the loss is equivalent to the mean-squared error (MSE) loss.

Training Procedure. The whole network is trained end-to-end. The LSTM module takes the historically predicted

motion as input, while the predicted motion is not accurate enough at the beginning. So we use the ground truth historical motion with $\sigma = 0$ to warm up the network training. We first train the network using ground truth historical motion for 200 epochs. Then, we switch the input historical motion to the network’s output and train for another 1200 epochs. Besides, the input historical motion vectors are treated as undifferentiable constants and detached from the computation graph. Thus the gradients of the LSTM module do not flow back to the graph neural network. We train the network using the Adam optimizer with a learning rate of 0.001 and a batch size of 64. The whole training process takes about four days using an NVIDIA RTX GeForce 2080Ti GPU.

To quantitatively evaluate the generalization ability of our network, we train the model on the humanoids subset and test it on the animal subset for motion estimation in Sec. 4.2. For other experiments, we use both subsets for training to achieve better reconstruction results.

In addition, as the estimated 3D motion of the visible part in the real world is not as perfect as the synthetic dataset, we add random Gaussian noise to the visible motion during network training. The noise is represented as $\mathcal{N}(\boldsymbol{\mu}, \sigma^2 \mathbf{I})$, where $\boldsymbol{\mu} = [0, 0, 0]$ and σ is sampled from a uniform distribution $\mathcal{U}(-0.4, 0.4)$. However, in the motion prediction evaluation in Sec. 4.2 we do not add any noise.

B. Details in Multi-scale Graph Pyramid Construction

We construct a 4-level multi-scale graph for message passing among nodes. Considering that the node connectivity of the first level is computed based on the Euclidean distance, which may lead to misconnections between unrelated parts, we discard edges by temporal consistency. More precisely, if the distance between two nodes changes more than a threshold, the edge between them will be discarded.

The node features downsampling between adjacent levels is performed directly by copying the node features to the higher level, as the nodes in the $(l + 1)$ th level are a subset of the nodes in the l th level. In feature upsampling, the node features of the $(l + 1)$ th level are assigned using the feature of the nearest neighbor in the l th level.

The intervals between nodes from the first to the fourth level are set to {4cm, 8cm, 16cm, 32cm} and the neighbor amounts are {8, 6, 4, 3}. Besides, the distance change threshold is set to 4cm.

C. RGB-D Based Optical Flow Prediction

We use the RAFT [39] network to estimate 2D optical flow. The original RAFT is trained with RGB images. We change the input dimensionality from 3 to 4 and retrain the network using RGB-D images as input. For the depth channel, we use the inverse depth (the reciprocal of depth) as

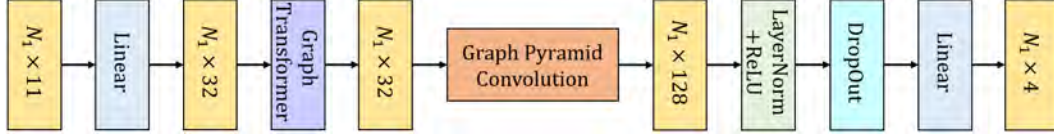


Figure 9. Architecture of the graph neural network.

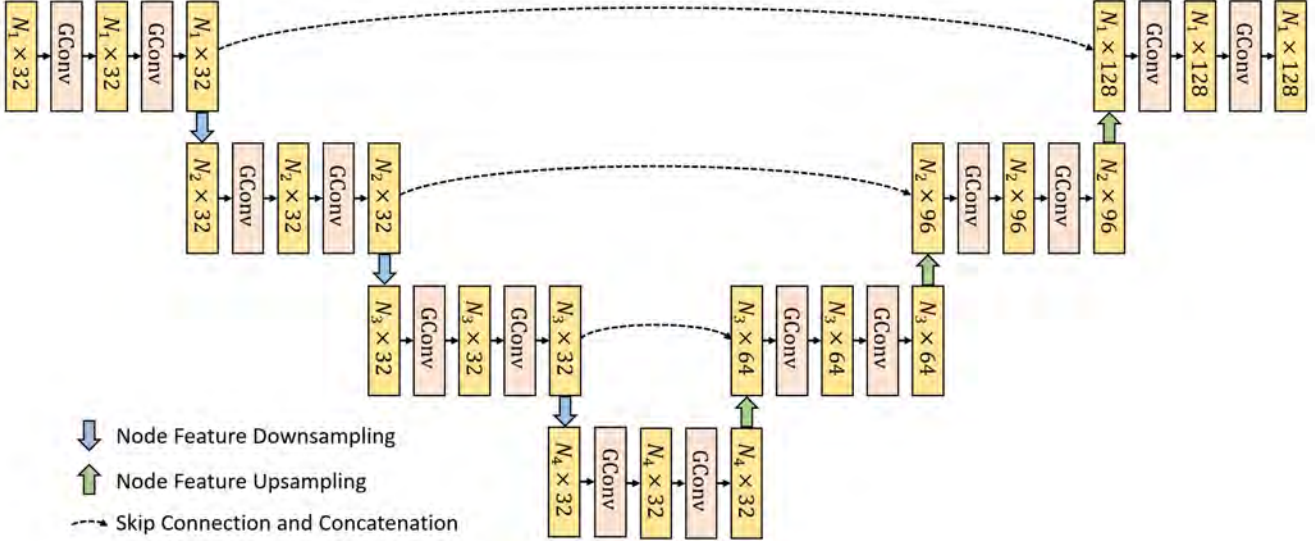


Figure 10. Architecture of the Graph Pyramid Convolution in Fig. 9.

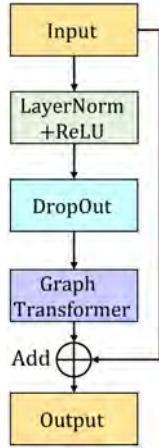


Figure 11. Architecture of the GConv block.

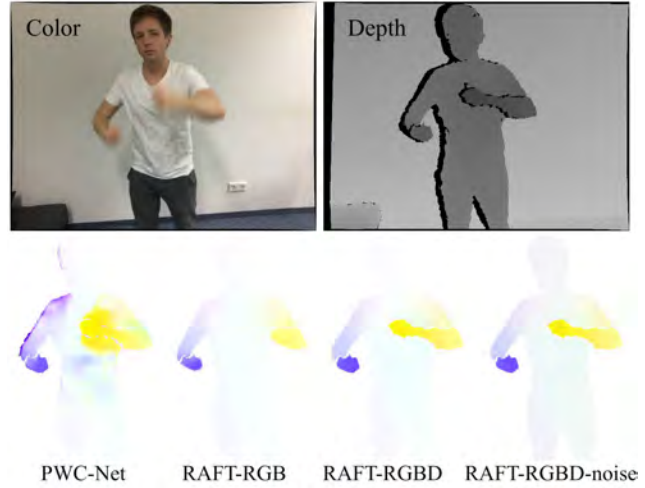


Figure 12. Results of different optical flow methods when motion blur occurs.

input.

To speed up the 2D optical flow estimation in the 3D reconstruction system, we resize the input images from 640×480 to 320×240 to compute the optical flow and upsample the optical flow to the original resolution by bi-linear interpolation.

D. Choice of Optical Flow

To test the robustness to flow estimation, we evaluate our system using different optical flow settings PWC-Net [38], RAFT-RGB and RAFT-RGBD. Besides, we further add Gaussian noise of $\mathcal{N}(0, 4)$ pixels on x and y axes to

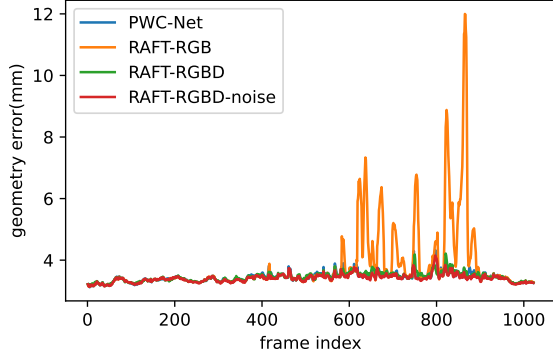


Figure 13. Geometry errors on the top sequence of Fig. 6. The average geometry errors over the whole sequence are 3.44mm, 3.79mm, 3.45mm, and 3.41mm from PWC-Net to RAFT-RGBD-noise.

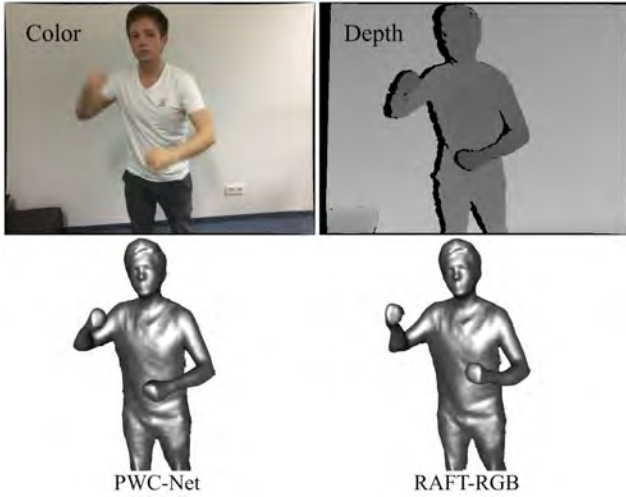


Figure 14. Reconstruction results based on PWC-Net and RAFT-RGB.

our RAFT-RGBD optical flow. We use the top sequence of Fig. 6 for evaluation. The optical flow results at the 833rd frame of the sequence are shown in Fig. 12. We can see severe motion blur occurs on the fast swinging arm in the color image, and significant errors appear in the optical flow of RGB-based methods (PWC-Net and RAFT-RGB). However, since depth images do not suffer the blur artifacts much and provide geometric information, RAFT-RGBD generates reliable optical flow. This indicates the benefit of involving depth in flow estimation.

For quantitative evaluation, we show the geometry errors of different optical flow methods over the whole sequence in Fig. 13. We can see that the geometry errors of our reconstruction method based on PWC-Net, RAFT-RGBD, and RAFT-RGBD-noise are all low and close to each other, which indicates that our method is robust to noise in the

optical flow. Only the reconstruction result based on the RAFT-RGB optical flow provides a large geometry error, which is caused by the tracking failure of the fast swinging arm. We show the reconstruction results using PWC-Net and RAFT-RGB at frame 837 (4 frames after the optical flow shown in Fig. 12) in Fig. 14. The reconstruction result of PWC-Net is better than RAFT-RGB, although they both provide a larger optical flow error. We believe the reason for this is that the optical flow errors of PWC-Net appear mainly on the torso, while those of RAFT-RGB appear on the anterior segment of the arm. In addition, errors on the torso are more easily corrected by the learned motion prior and the regularization term in the optimization, because there is more motion information around the torso region where the predicted optical flow is wrong.