Computational Visual Media
DOI 10.1007/s41095-xxx-xxxx-x Vol. x, No. x, month year, xx–xx

Research Article

HDR-Net-Fusion: Real-time 3D Dynamic Scene Reconstruction with a Hierarchical Deep Reinforcement Network

Haoxuan Song[1], **Jiahui Huang**[1], **Yan-Pei Cao**[2] **and Tai-Jiang Mu**[1] (✉)

Abstract Reconstructing dynamic scenes with commodity depth cameras has many applications in computer graphics, computer vision, and robotics. However, due to the presence of noise and erroneous observations from data capturing devices and the inherently ill-posed nature of non-rigid registration with insufficient information, traditional approaches often produce low-quality geometry with holes, bumps, and misalignments. We propose a novel 3D dynamic reconstruction system, named HDR-Net-Fusion, which learns to simultaneously reconstruct and refine the geometry with a sparse embedded deformation graph of surfels on the fly with a hierarchical deep reinforcement(HDR) network, which consists of two parts: A global HDR-Net rapidly detecting local regions with large geometric errors, and a local HDR-Net serving as a local patch refinement operator to promptly complete and enhance such regions. To train the global HDR-Net, we formulate it as a novel reinforcement learning problem to implicitly learn the region selection strategy with the goal of improving the overall reconstruction quality. The applicability and efficiency of our approach are demonstrated using a large-scale dynamic reconstruction dataset. Our method can reconstruct the geometry with higher quality than traditional ones.

Keywords Dynamic Reconstruction; Deep Reinforcement Learning; Point Cloud Completion; Deep Neural Networks.

1 Introduction

3D reconstruction is a key technique in computer graphics with various applications in virtual and augmented reality and animation. In recent years, many advances have been made in both reconstruction quality and speed. Since the early success of KinectFusion [33], scanning with a commodity RGB-D camera and building the captured geometry in an online fashion has become commonplace in modern reconstruction systems. Subsequent work has either improved system scalability to support larger scenes and finer details by introducing new persistent data structures [28, 34, 54], or has focused on enhancing reconstruction quality through accurate frame-to-model registration [2, 7, 8].

While research on reconstructing and modeling static indoor scenes [4] has matured in the past few years, reconstructing dynamic objects (e.g. humans, animals and other freely moving objects) still remains an open problem in both the graphics and robotics communities (referred to as *dynamic SLAM* [9, 20, 21, 40, 41]). Given an input sequence recording a non-rigid deforming object, the goal of dynamic reconstruction is to recover the moving object's underlying shape in a canonical pose as well as the deformation field for each frame so that the geometry at each instant of time can be recovered. The seminal work of DynamicFusion [32] described a general pipeline adopted by many other algorithms: by parametrizing the per-frame deformation as a warp field defined on a sparse set of transformation nodes skinned from the full geometry, the underlying shape can be registered to the depth observations at a particular time via solving a non-rigid iterative closest point (NR-ICP) problem [3], yielding the transformation for every node. To further improve the robustness, many industrial and academic solutions use dedicated hardware [16, 35], or exploit a common deformable template [58, 59] as a prior to regularize the final result. On the other hand, multi-view reconstruction systems like Fusion4D [8] and FusionMLS [29] leverage

1 BNRist, Department of Computer Science and Technology, Tsinghua University, Beiing, 100084, China. E-mail: H. Song, H. Huang, {songhx17,huang-jh18}@mails.tsinghua.edu.cn; T.-J. Mu, taijiang@tsinghua.edu.cn (✉).

2 Kuaishou Technology Co., Ltd, Beijing, 100085, China. E-mail: caoyanpei@gmail.com.

Manuscript received: 2014-12-31; accepted: 2015-01-30.

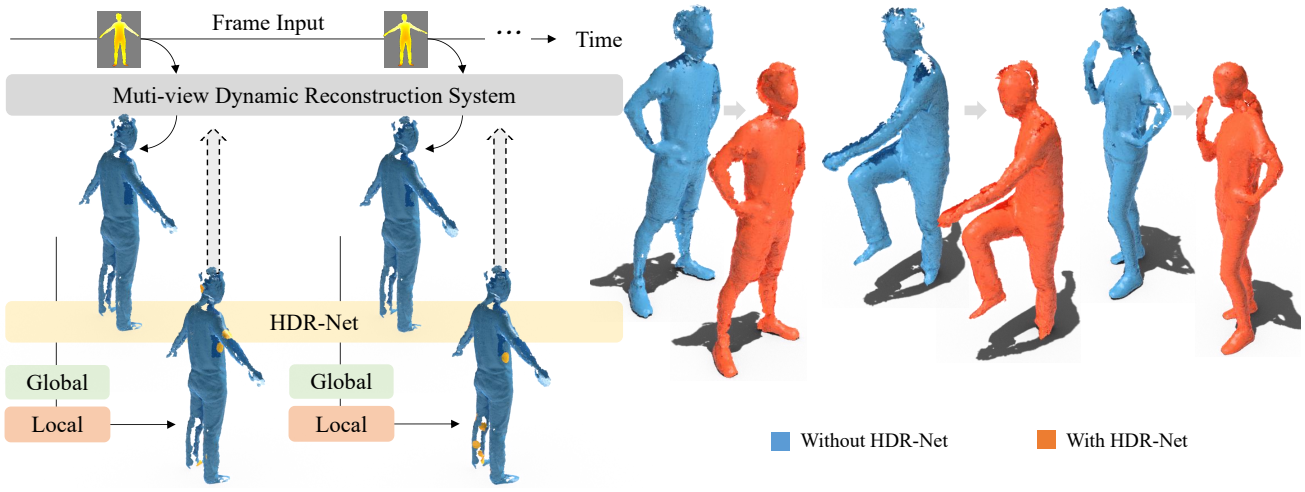

Fig. 1 HDR-Net-Fusion simultaneously reconstructs and performs patch-based geometric refinement in an online fashion. Compared to traditional reconstruction methods, our method can automatically complete missing regions and refine noisy patches, improving the overall reconstruction quality. In this figure, the yellow regions indicate refined patches over the geometry.

more complete observations from a large number of cameras to reconstruct the geometry with higher quality.

However, reconstructing scene dynamics is inherently an ill-posed problem because the solution space for occluded regions which are not observed by any camera can be infinitely large [11]. Various regularization terms, such as as-rigid-as-possible constraints, are used to tackle this problem to some extent, but they are not always met in real-world scenarios. Another challenge is the low-quality output provided by the capture device, which tends to contain noise and erroneous depth observations, resulting in artifacts in the final reconstructed model such as holes and bumps. To address the above challenges, we pursue a data-driven framework based on state-of-the-art deep learning techniques which can be easily integrated into an existing dynamic reconstruction pipeline to enhance the fusion quality.

Deep neural networks (DNN) have shown their applicability in a wide range of graphics applications such as shape completion [36, 60], geometric registration [13, 50], flow/correspondence estimation [15, 27], etc. Recently, 3D deep learning has also gained ever more attention in reconstruction applications [56]. However, there are few attempts to embed deep models directly into reconstruction systems, primarily due to efficiency and generalization considerations. Furthermore, in online systems where succeeding frames rely directly on previous fusion results, deep models directly operating over the already fused geometry [25, 60] cannot utilize intermediate fusion results, and it is impossible to recover from any catastrophic tracking failure in the reconstruction system.

In order to maintain the system efficiency while exploiting the power of deep learning models, we present HDR-Net-Fusion, a highly-efficient surfel [24]-based dynamic reconstruction system for simultaneous reconstructing and refining 3D dynamic scenes using a hierarchical deep reinforcement network, i.e., HDR-Net. The core of HDR-Net consists of two parts: a Global-HDR-Net and a Local-HDR-Net. The global net first considers the overall geometric structure of the current model, and determines those local patches which may be of poor quality, potentially leading to bad registration results for future frames. Then, the local net fixes such detected regions, performing patch-based geometry refinement using a data-driven neural network. We formulate the training of the global HDR-Net as a reinforcement learning problem: the optimal region selection strategy is implicitly learned to minimize the overall reconstruction error, considering both short-term and long-term loss during the fusion process. Our system is empirically proved to be accurate, robust and efficient. As far as we know, this is the first work to integrate deep neural networks into reconstruction using a reinforcement learning approach.

Briefly speaking, this paper makes the following contributions:

- the first efficient hierarchical deep reinforcement network integrated with a real-time dynamic multi-view 3D reconstruction.
- a reinforcement learning model for efficient, incremental to-fixed region selection.
- a deep neural network for high quality local reconstruction refinement.

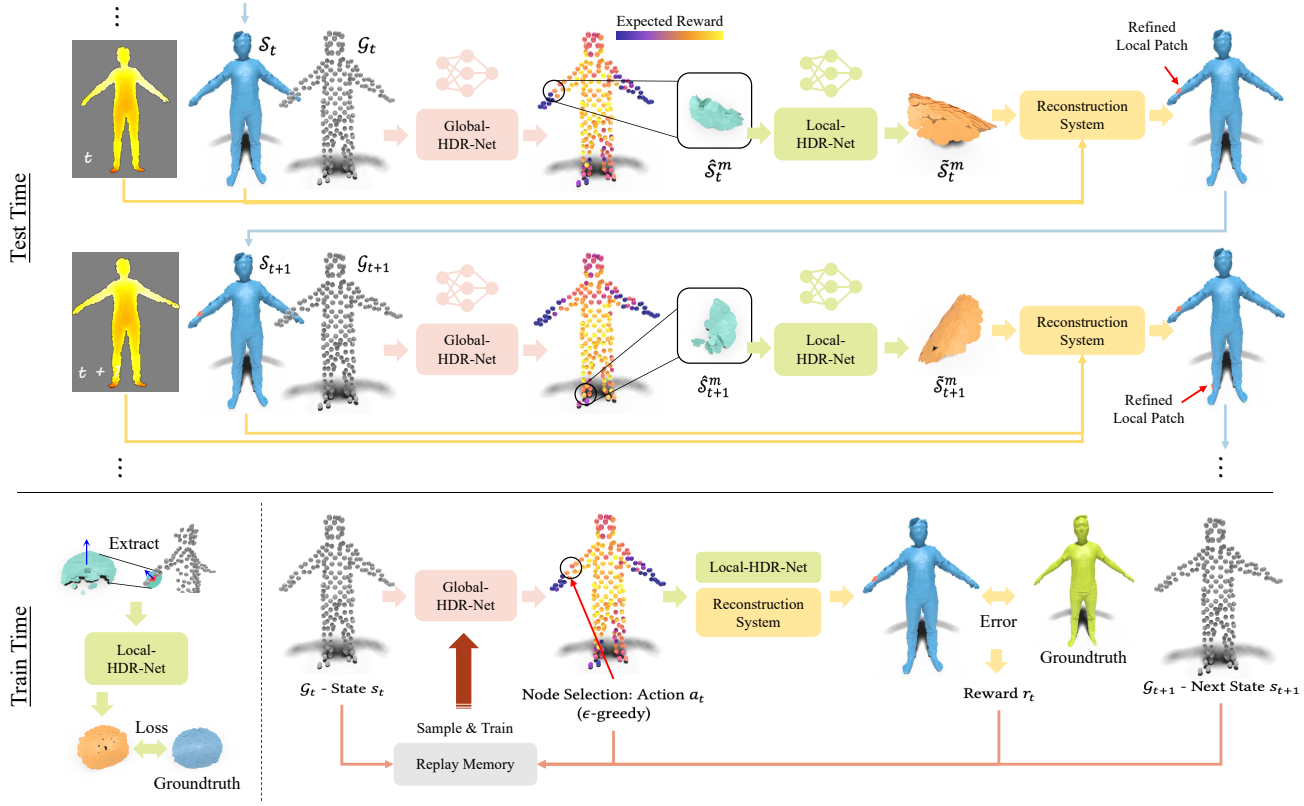

Fig. 2 An overview of our whole reconstruction system. At test time, the deformation nodes $\mathcal{G}_t$ of the live geometry $\mathcal{S}_t$ is fed into Global-HDR-Net and the local patch $\hat{\mathcal{S}}_t^m$ with the highest expected reward is selected and fed into Local-HDR-Net for refinement. The refined patch $\tilde{\mathcal{S}}_t^m$ replaces the original geometry and is fused into the whole model, which will be used for registering the next incoming frame. In order to train the global/local hierarchical networks, we first supervise the Local-HDR-Net with groundtruth full patches; then we fix its weight and train the Global-HDR-Net represented as a point-set-based DQN [31].

2 Related Works

Dynamic Reconstruction Inferring dynamic scene geometry remains an open research topic. Some work [5, 42, 45, 47, 58] adopts strong semantic scene priors (e.g. a human body or hand template) to facilitate accurate correspondence and registration. Other methods [12, 18, 22, 32], instead, choose to aggregate and denoise geometry in a canonical static space, and only track the per-frame deformation field over time, without the knowledge of the reconstructed scene beforehand. To tackle the inherent ambiguity of the deformation field, and achieve better reconstruction fidelity, [7, 8, 18] introduce sparse image feature tracking, silhouette constraints and albedo inference into the non-linear optimization to make tracking more robust, while [43, 44] bypass the correspondence estimation stage by imposing divergence constraints over the entire deformation vector field, and [16, 17] give dedicated hardware designs for obtaining cleaner and more complete depth and texture information. Readers are referred to [62] for a comprehensive literature review. Our approach introduce deep neural models to efficiently learn the geometric priors from dataset for higher fusion quality.

Point Set Deep Networks For our surfel-based representation of the reconstructed geometry, we apply deep networks which directly consume point clouds. PointNet [38] and its variants [39, 51, 55] are a standard choice for encoding point set features while providing a good description of multi-scale details. The work in [10] is the first point set decoder combining fully-connected and deconvolution layers. In order to enforce a uniform structure onto the generated point set, FoldingNet [57] and AtlasNet [14] used one or more uniform grids to condition the shape descriptor for shape generation. The designs of various deep point set networks support a variety of applications in both graphics and vision, such as point upsampling [25, 52] and shape completion [46, 60].

Deep Reinforcement Learning Traditional reinforcement learning aims to learn from past experiences and make better decisions in a principled way. The successful combination of deep neural networks and reinforcement learning algorithms has shown the capability of dealing

with higher-dimensional state and action spaces which were previously intractable [1]. Deep reinforcement learning has various applications in video games [31], generating animation [37] and indoor navigation [61]. A prominent approach is provided by the Deep Q-Network (DQN) [31] and its variants [48, 53], which approximate value functions with off-policy learning. Another line of approaches is based on policy gradients or actor-critics, where the model directly learns a stochastic policy [26, 30]. Our work formulates dynamic reconstruction as a Markov decision process and applies DQN to learn how to achieve minimum reconstruction error. We believe this is the first application of deep reinforcement learning in dynamic reconstruction pipelines.

3 Overview

Our HDR-Net-Fusion takes sequential depth maps captured using several commodity RGBD cameras as input and progressively reconstructs the geometry of the dynamic scene for every frame. As shown in Fig. 2, during the test phase of our algorithm, for each incoming frame, the warping field which best aligns the current depth observations and the reference geometry is first found, then a traditional fusion process will be operated by our basic reconstruction system (Sec. 5). After that, the Global-HDR-Net is applied to the embedded deformation nodes to compute an expected reward for each node (Sec. 6.2). The node with the highest expected reward is then selected and the local surfel patch surrounding that deformation node is fed into the Local-HDR-Net, which locally refines the patch geometry and completes missing areas (Sec. 6.1). The refined patch is then integrated into the reference geometry maintained by the reconstruction system to improve the quality of reconstruction and assist future tracking and registration.

4 Notations and Scene Representation

The dynamically reconstructed scene is represented by a set of deformation nodes $\mathcal{G} = \{\boldsymbol{g}^m \in \mathbb{R}^3\}$ and a set of surfels with neighborhoods $\mathcal{S} = \{s^i, N^i\}$. $\mathcal{S}$ is a dense reconstruction of the entire scene, while the nodes in $\mathcal{G}$ are scattered sparsely over the surface represented by the surfels. Each surfel $s^i = (\boldsymbol{p}^i, \boldsymbol{n}^i, r^i)$ is composed of its center $\boldsymbol{p}^i \in \mathbb{R}^3$, normal $\boldsymbol{n}^i \in \mathbb{R}^3$ and radius $r^i \in \mathbb{R}$. A neighbourhood set $N^i \subset \mathcal{G}$ is attached to each surfel, initialized as the nearest K neighbours of s^i among all deformation nodes $\boldsymbol{g}^m$ in $\mathcal{G}$. Similarly, a neighbourhood set $\mathcal{N}^m \subset \mathcal{G}$ can be built for all the deformation nodes to establish their spatial relationships.

For each frame t we compute a warping field $\mathcal{W}_t = \{q_t^m \in \mathbb{SE}(3)\}$ defined at each node in $\mathcal{G}$, where q_t^m is the transformation applied to $\boldsymbol{g}^m$ in frame t; it is represented using dual quaternions [23]. Let $\mathcal{G}^t = \{\boldsymbol{g}_t^m \in \mathbb{R}^3\}$ be the transformed version of $\mathcal{G}$ where $\boldsymbol{g}_t^m = q_t^m \cdot \boldsymbol{g}^m$. The surfels skin the deformation nodes and the transformation for each surfel is found by interpolating nearby node transformations as $\hat{q}_t^i = \sum_{m \in N_i} w_{im} q_t^m$, where $w_{im} = \exp(-\|\boldsymbol{p}^i - \boldsymbol{g}^m\|_2^2/\sigma^2)$, σ representing the node sampling distance [12]. We denote the transformed version of the surfels at frame t after applying $\hat{q}_t^i$ as $\mathcal{S}_t = \{s_t^i, N^i\}$, $s_t^i = (\boldsymbol{p}_t^i, \boldsymbol{n}_t^i, r^i)$, where $\boldsymbol{p}_t^i = \hat{q}_t^i \cdot \boldsymbol{p}^i$, and $\boldsymbol{n}_t^i$ is the normal $\boldsymbol{n}^i$ transformed by the rotation part of $\hat{q}_t^i$ only.

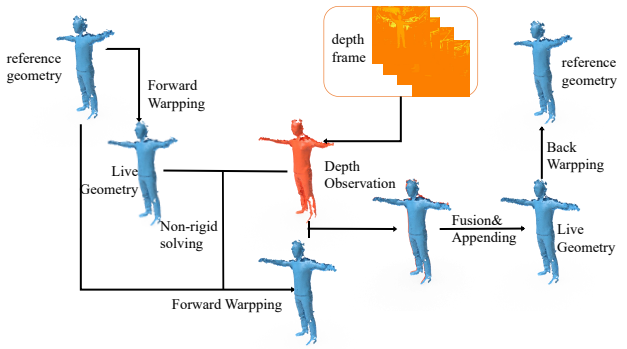

Fig. 3 Pipeline of the basic reconstruction system. For each incoming frame, we first forward warp the reference geometry into live geometry according to the current warp field, then a new deformation field is solved to align the reference geometry and depth observation. After that, the depth observation is fused with the new live geometry. Finally, we warp the live geometry back to the reference geometry.

5 Basic Reconstruction System

The design of our basic reconstruction system is inspired by [12] and illustrated in Fig. 3. The initial surfels $\mathcal{S}$ are called the *reference geometry* and the up-to-date surfels $\mathcal{S}_t$ are called the *live geometry*. For each frame, $\mathcal{W}_t$ is determined and the new surfels introduced in the current frame are appended to the live geometry; matched surfels are updated according to the running mean integration protocol [24]. The live geometry is then warped back to the reference geometry which provides a canonical shape representation.

Energy Function A key step in the dynamic reconstruction is to find the per-frame warping field $\mathcal{W}_t$, which is solved by minimizing the following energy, consisting of a data term, a correspondence term and a regularization term:

$$E(\mathcal{W}_t) := E_{\text{data}}(\mathcal{W}_t) + \lambda_{\text{c}} E_{\text{cor}}(\mathcal{W}_t) + \lambda_{\text{r}} E_{\text{reg}}(\mathcal{W}_t), \quad (1)$$

where λ_{c} and λ_{r} are balancing weights. The data term,

$$E_{\text{data}}(\mathcal{W}_t) := \sum_{v=1}^{V} \sum_{i \in \mathcal{V}_t^v} \|\boldsymbol{n}_{d_t^{i,v}}^{\top} (\boldsymbol{p}_t^i - \boldsymbol{p}_{d_t^{i,v}})\|^2, \quad (2)$$

is a depth-to-plane ICP error summing across all V input views per frame, where $\mathcal{V}_t^v$ is the visible surfel index set for the current $\mathcal{W}_t$ from the v-th view, and $d_t^{i,v}$ is the corresponding depth observation of the i-th surfel found by re-projecting s_t^i into the v-th camera view, transformed by the camera extrinsic. The correspondence term is:

$$E_{\text{cor}}(\mathcal{W}_t) \coloneqq \sum_{v=1}^{V} \sum_{(s_t^i, u_t^i) \in \mathcal{C}_t^v} \|\boldsymbol{p}_{u_t^i}^v - \boldsymbol{p}_t^i\|^2, \tag{3}$$

which is a distance between two sparsely related points found by global patch collider [49] and $\mathcal{C}_t^v$ is the correspondence set containing tuples of matched surfels s_t^i and pixels u_t^i for the v-th view. $\boldsymbol{p}_{u_t^i}^v$ is the 3D point re-projected from pixel u_t^i at view v. The regularization term,

$$E_{\text{reg}}(\mathcal{W}_t) \coloneqq \sum_{\boldsymbol{g}^m \in \mathcal{G}} \sum_{n \in \mathcal{N}^m} \|\boldsymbol{g}_t^m - q_t^n \cdot \boldsymbol{g}^m\|_2^2, \tag{4}$$

is an as-rigid-as-possible constraint encouraging nearby nodes to share the same transformation.

Challenges Generally, our basic reconstruction system can work well on simple datasets and special cases like topology changes and tracking failure can be fixed by re-initialization [12]. However, without any prior knowledge about the dynamic scene structure, it is still very challenging to track fast motions and a lot of information will be lost during re-initialization. The situation will become worse when there are erroneous depth observations or when parts of the dynamic structure are occluded from any views.

6 HDR-Net: Hierarchical Deep Reinforcement Network for Dynamic Reconstruction

To address the challenges faced by the traditional reconstruction system discussed in Sec. 5, a repairing to the erroneous and occluded parts is necessary, while the efficiency of the reconstruction system should also be guaranteed. We thus propose a hierarchical reinforcement network (HDR-Net) that first find the to-fixed regions (Global-HDR-Net) efficiently using reinforcement learning algorithms and then fix these regions by exploiting the power of deep neural network (Local-HDR-Net).

6.1 Local-HDR-Net

Network Architecture For each frame, given a selected deformation node $\boldsymbol{g}_t^m$ from Global-HDR-Net, we gather all surfels influenced by that node as $\hat{\mathcal{S}}_t^m \coloneqq \{s_t^i | \boldsymbol{g}^m \in N^i\}$ and feed that local patch into the Local-HDR-Net. Local-HDR-Net's job is to generate $\tilde{\mathcal{S}}_t^m$, a completed and denoised version of $\hat{\mathcal{S}}_t^m$. Two design requirements exist for our model: (i) as the network is applied to the reconstructed geometry on a per-frame basis, the model should be lightweight, requiring minor additional computation, and

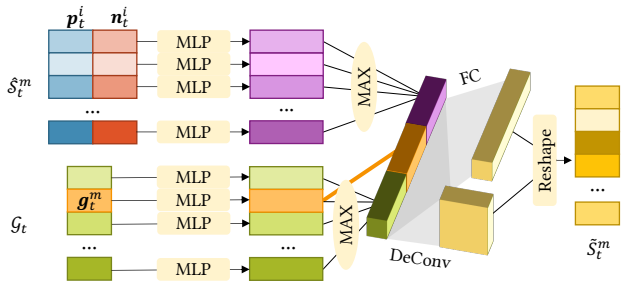

Fig. 4 Local-HDR-Net Model. The network takes the local surfel patch and all deformation nodes as input. The concatenated latent feature vector is decoded using fully-connected and deconvolution layers.

(ii) in order to resolve the inherent ambiguity of point set completion, knowledge of the entire scene geometry should be taken into consideration.

We therefore propose a hybrid encoder-decoder structure using the ordering-agnostic PointNet [38] as the backbone, as shown in Fig. 4. To integrate global geometric knowledge, we use $\mathcal{G}_t$ as a summary of the current coarse shape: we find that this gives a good global shape approximation which can effectively summarize the overall scene structure to the network.

In the encoder part of our model, $\mathcal{G}_t$ and $\hat{\mathcal{S}}_t^m$ are first encoded separately, extracting features with respective point-shared MLPs. As the encoded feature of each point in $\mathcal{G}_t$, we take its globally aggregated feature vector as well as the point feature vector for $\boldsymbol{g}_t^m$. These two feature vectors are then concatenated with the aggregated per-point feature of $\hat{\mathcal{S}}_t^m$. The overall aggregated latent representation of the local region now contains information summarizing the patch geometry in its global context.

In the decoder, we find that using a classic fully connected and deconvolution combination [10] generates the best results while still allowing real-time processing. Deformation-based decoders [57] easily lead to over-smoothed surfaces lacking detail, while implicit-function-based decoders [36] involve heavy sampling computations during inferencing. The direct output of our model is simply the 3D surfel center position's offset to the selected node $\tilde{\boldsymbol{o}}_t^i$, which is easier to learn compared than the surfel normal, given its spatial continuity.

Loss Function We use the earth-mover distance (EMD) as the loss function to train the network:

$$\mathcal{L} \coloneqq \frac{1}{|\tilde{\mathcal{S}}_t^m|} \min_{\phi: \tilde{\boldsymbol{p}} \to \bar{\boldsymbol{p}}} \sum_{s_t^i \in \tilde{\mathcal{S}}_t^m} \|\bar{\boldsymbol{p}}_t^i - \phi(\tilde{\boldsymbol{p}}_t^i)\|_2^2 \tag{5}$$

where $\tilde{\boldsymbol{p}}_t^i = \tilde{\boldsymbol{o}}_t^i + \boldsymbol{g}_t^m$ is the center position of each output surfel in the world coordinate. ϕ is a bijection; the best linear assignment expressed by the $\min$ operator can be computed

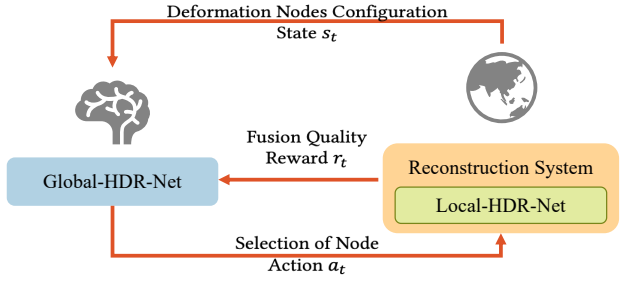

Fig. 5 Reinforcement learning for training Global-HDR-Net. The reconstruction system as well as Local-HDR-Net serve as the environment while Global-HDR-Net is the agent whose task is to select a deformation node in $\mathcal{G}_t$ to be refined by the Local-HDR-Net for each frame.

efficiently using the approach in [10]. $\bar{\boldsymbol{p}}$ are the ground truth surfel center.

6.2 Global-HDR-Net

Problem Formulation For each frame t, Global-HDR-Net aims to select the node $\boldsymbol{g}_t^m$ for the subsequent local patch refinement operation described in Sec. 6.1. One can definitely choose to refine more than one node, or in the extreme case, all the nodes in one single frame. However, too many passes of network inference will drastically affect the system's real-time performance. In fact, as the output reward is not supposed to vary largely given subtle changes in the input node positions, all nodes with high reward will be eventually picked up in time based on the high capturing frame rate. By instead performing inference on one node at a time, we distribute the computation across the entire session so that the running speed is guaranteed while still not preventing any nodes from being chosen.

One simple strategy for this module could be to always select $\mathcal{S}_t^m$ with the worst geometric quality. However, a greedy algorithm will not necessarily lead to a globally optimal result as it does not consider possible future registration error and the empirical performance of Local-HDR-Net. We instead pursue an algorithm that is aware of both short-term and long-term reconstruction quality and takes the properties of both the underlying dynamic reconstruction system and Local-HDR-Net into account.

We propose to solve this problem using ideas from reinforcement learning (RL) which implicitly model the environment using existing experience gained through trial and error. A natural analogy can be made between Global-HDR-Net and a reinforcement learning (RL) agent. The dynamic reconstruction system and the local net serve as the environment, which receives an *action* (a deformation node $\boldsymbol{g}_t^m$) from the network, performs internal fusion and local patch refinement, and emits the reconstructed result as the new observation. The rewards for the action performed can be modeled by the score of reconstruction quality. By choosing different actions at each timestamp, the Global-HDR-Net agent influences internal state of the system and all the succeeding reconstruction steps.

The target of RL is to learn an optimal policy which can be later executed during inferencing. The optimal policy maximizes the expected return along the state transition path, which, in our case, effectively minimizes reconstruction error over all time steps.

From a theoretical point of view, two key propositions have to be met for the above formulation to be meaningful. Firstly, the reconstruction system should obey the Markov property, where the state of the current step is solely dependent on the previous step's state. Secondly, an appropriate choice of the deformation node can be made solely from the configuration of $\mathcal{G}_t$. The first proposition is naturally satisfied because for each frame, the depth observation is integrated only with the fused geometry from the previous frame. Also, we have found that regions with poor reconstruction quality often have highly complicated or mostly occluded parts, which to a certain degree justifies us in assuming the second proposition to be true.

Learning Algorithm and Network Architecture We employ DQN [31], which uses an efficient off-policy value-function based approach, as our reinforcement learning algorithm. DQN aims to learn the Q-function (expected reward given state and action) through past experiences, and approximates $Q(s_t, \cdot)$ using a deep neural network (i.e. the Q-Network) to model the high-dimensional state and action space. Here we use s_t and a_t to denote the state and action for frame t. Specifically, s_t represents the positions of global nodes $\mathcal{G}_t$ up to frame t and a_t is the integer index m of the selected deformation node in $\mathcal{G}_t$. By enforcing Bellman equality and minimizing temporal difference error δ_t, the Markov process of the environment can be precisely modeled by the Q-function and our final policy can be greedily selected as $\pi^*(s_t) := \operatorname{argmax}_a Q(s_t, a)$ so that in each frame t we maximize $\sum_{t'=t}^{T} \gamma^{t'-t} r_{t'}$, where $\gamma > 0$ is the discount factor, r_t is the reward for frame t and T is the number of total frames.

Following [31] we define the temporal difference error as the following:

$$\delta_t(\Theta) := Q(s_t, a_t; \Theta) - (r_t + \gamma \max_a Q(s_{t+1}, a; \Phi)), \quad (6)$$

where Θ is the parameter of the policy deep network and Φ is the parameter of the target deep network. During training, we execute the reconstruction system with Local-HDR-Net several times and gather the (s_t, a_t, r_t, s_{t+1}) tuples. Here, the actions a_t are chosen using an ϵ-greedy policy which interpolates between the currently found best policy π^*

and a completely random policy with factor ϵ. It can be proved that this strategy can converge to an optimal policy, balancing exploration and exploitation in state space. We store multiple state-action-reward tuples across different episodes in a common replay memory. Mini-batches are then sampled from the replay memory to train the policy network parameter Θ using back propagation, so that $\delta_t^2(\Theta)$ is minimized. Φ is usually fixed and updated to Θ only every few episodes to guarantee a stable training.

We choose PointNet++ [39] as our Q-network. It takes in point set $\mathcal{G}_t$ at frame t and the numbers of local surfel patch $\mathcal{S}_t^m$(m$\in \mathcal{G}_t$) as input and predicts the Q-value, i.e. the expected reward for each point (possible action).

The reward r_t is taken as the negative chamfer distance $\mathcal{D}$, defined as:

$$\begin{aligned}\mathcal{D} :=& \frac{1}{|\mathcal{S}_{\mathrm{p}}|} \sum_{\boldsymbol{p}^i \in \mathcal{S}_{\mathrm{p}}} \min_{\boldsymbol{p}^j \in \mathcal{S}_{\mathrm{g}}} \|\boldsymbol{p}^i - \boldsymbol{p}^j\|_2 \\ &+ \frac{1}{|\mathcal{S}_{\mathrm{g}}|} \sum_{\boldsymbol{p}^j \in \mathcal{S}_{\mathrm{g}}} \min_{\boldsymbol{p}^i \in \mathcal{S}_{\mathrm{p}}} \|\boldsymbol{p}^i - \boldsymbol{p}^j\|_2,\end{aligned} \tag{7}$$

where $\mathcal{S}_{\mathrm{p}}$ are the surfel positions of the current geometry using the reconstruction system and $\mathcal{S}_{\mathrm{g}}$ is the ground-truth reference geometry.

7 Results and Discussions

In this section, we introduce the experimental setup for implementing our system, show the results and comparisons, and validate the design of our method.

7.1 Experimental Setup

Dataset Our experiments used sequences from the Human10 dataset [28] to test our algorithm. This dataset contains 10 long sequences of several human actors performing various actions, of which 9 are publicly usable. In Human10, each sequence was recorded with 4 fixed-position 512×512 resolution RGB-D cameras distributed uniformly around a 360° viewing circle. The frustum of each depth camera covers a partial view of the entire human body. The limited sensor quality, leading to severe depth error and loss, many fast large motions as well as topology changes present very challenging data to the reconstruction system, resulting in very frequent tracking loss and re-initialization. To measure reconstruction quality, the dataset provides a ground truth 3D mesh reconstructed using a free-viewpoint-video [6] capture system.

To verify the generalization of the network, we split the 9 sequences into training sequences (human1/2/4/6/9) of HDR-Net and testing sequences (human0/3/7/8). Weights for both Local-HDR-Net and Global-HDR-Net were learned and cross-validated solely from the training frames. In order to train Local-HDR-Net, patch-level surfel and deformation node data are generated. We first generate surfels and nodes from depth observations every single frame without warping, to simulate the artifacts caused by re-initialization and cameras' quality. On the other side, we use Poisson disk sampling to sample equally-spaced surfels over the ground truth mesh. We then gather surfels from both the reconstruction results and the ground truth surrounding each node to form a complete patch, using a ball query, forming a local patch training pair for the supervised learning of patch completion. Additionally, we balance the distribution of training pairs by their *completeness* score, defined as the portion of ground truth surfels closer than a certain threshold to its nearest neighbour in the partial surfel patch (extracted from the input depth map). Empirically we find better overall system performance can be achieved with this balanced dataset, most of whose training pairs would otherwise be almost complete.

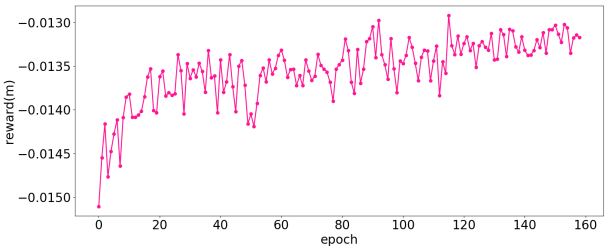

Fig. 6 The variation of the average reward per epoch during the training process of global-HDR-Net.

Training Protocol We use a common supervised training strategy to optimize the Local-HDR-Net using the dataset described above: an AdaGrad optimizer is used with an learning rate of 10^{-3}. The training of Global-HDR-Net is based on 200 randomly selected consecutive frames from each episode. For each frame, we randomly sample a state-action-reward tuple batch from the replay memory and optimize Θ with the RMSprop optimizer. The network weight Φ is updated to Θ every 3 episodes. During execution of the ϵ-greedy policy, we start with 90% probability of selecting random nodes and decreased the probability exponentially to 5% with a 200 frames decay rate. The discount factor is set to $\gamma = 0.999$. In total, we train for around 160 episodes to get a fairly convergent result. The loss curve is shown in Fig. 6.

To evaluate the system performance, all input frames are never seen by neither networks, allowing fair evaluation of our pipeline. Specifically, among the only four sequences (human0/3/8/9) which contain RGB information and can achieve a good result in multi-view sequences' tracking, we choose human9 for the training of Global-HDR-Net.

Fig. 7 Selected frames demonstrating the overall reconstruction quality of HDR-Net-Fusion (Rows from top to down correspond to human0/3/8 in Human10, respectively). In each pair of meshes in each row, we show the reconstructed result without HDR-Net in blue and the result with HDR-Net in red. Our algorithm successfully identifies the missing or noisy regions and refines them reasonably.

Considering the dependency of Global-HDR-Net's training on Local-HDR-Net's performance, Global-HDR-Net would learn non-generalizable policies if Local-HDR-Net is too familiar with the sequence we train Global-HDR-Net on. Therefore, we only take human1/2/4/6 for Local-HDR-Net's training and evaluating, excluding the human9.

Implementation Details We implemented our multi-view reconstruction system in C++/CUDA. The training code for both Local-HDR-Net and Global-HDR-Net is written using PyTorch. We interface the reconstruction system and the deep network part so that both training and inferencing are tightly coupled and trained effectively end-to-end. The entire algorithm is tested on a workstation with an Nvidia Titan RTX graphics card, running the reconstruction pipeline, Local-HDR-Net and Global-HDR-Net simultaneously.

In practice, we find that Local-HDR-Net does not guarantee a perfectly smoothed output, which may degrade the overall system performance. Hence we separately adopt a post-processing step to directly reject bad local net output $\tilde{\mathcal{S}}_m^t$. This post-rejection step finds all nearest neighbour pairs in $\tilde{\mathcal{S}}_m^t$ and computes the dot product of normal vectors of the pair. This generated surfel patch is rejected if the mean value of all dot products is less than ε, which can guarantee the smoothness of the surfels we add.

In addition, the number of fixed nodes each frame can be adjusted in practice to achieve a better quality. Since the Global-HDR-Net can output the expected reward of all nodes, we can select several nodes within an acceptable range instead of the best one and fix them one by one so that we can refine a necessary number of nodes every frame while still enabling the real-time performance.

In the presence of fast motion, the tracking module may fail, we detect this abnormality by checking the residuals of the registration solver and perform re-initialization operation [8, 12].

Parameter Selection The parameter chosen for the reconstruction system is dependent on the applications. For effectively tracking and recovering human geometry, we empirically set the reconstruction parameters in Sec. 5

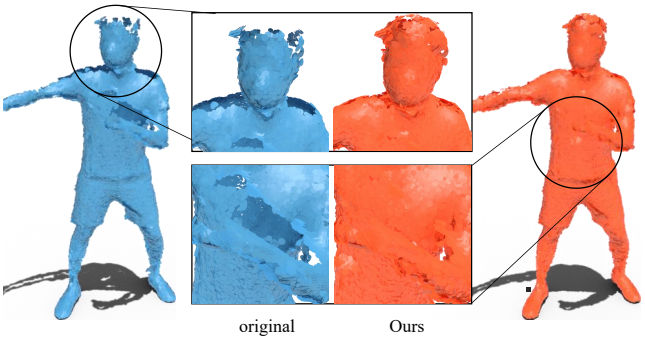

Fig. 8 Close-up comparison of the geometry reconstructed by basic method and our method. HDR-Net can solve both the artifacts caused by erroneous depth observation and the incompleteness caused by re-initialization.

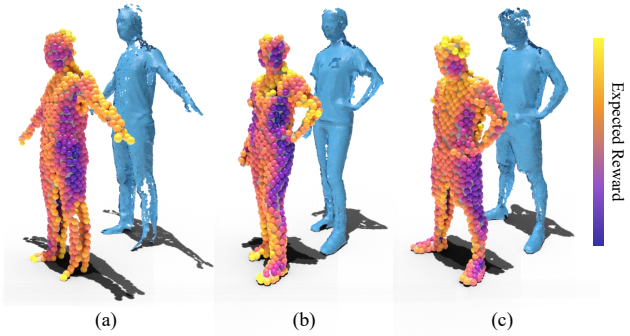

Fig. 9 Expected reward for the deformation nodes computed by the Global-HDR-Net (superimposed on the surfels). Since the Q-value's interval of each sequence is not uniform, we visualize the relative values by coloring the nodes where blue means relatively lower value and yellow means the higher.

Tab. 1 Times of re-initialization during the reconstruction of our framework without (w/o) or with (w/) HDR-Net. Our method can provide a reduction of re-initialization especially when the re-initialization is frequent.

Sequence	w/o HDR-Net	w/ HDR-Net
Human0	20	13
Human3	15	11
Human8	7	6

as $\lambda_{\mathrm{r}} = 2.3$ and $\lambda_{\mathrm{c}} = 1.3$ while the node sampling distance [12] $\sigma = 0.04$cm. The post-processing rejection threshold ε is set to 0.9.

In terms of network structure, our Local-HDR-Net encodes surfels in $\hat{\mathcal{S}}_t^m$ using a shared MLP with sequentially 32, 64, and 256 channels, encodes each point in $\mathcal{G}_t$ with 32, 64, 256 channels and transforms the concatenated latent feature into two patches of 256 3D points with FC and DeConv layers separately. The Global-HDR-Net downsamples input deformation nodes into 256, 64 and 16 points sequentially with set abstraction layers and the local feature is interpolated using feature propagation layer. The input nodes $\boldsymbol{G}_t$ are padded for a minimum size of 400.

7.2 Overall Performance

We present some qualitative results of our entire HDR-Net-Fusion framework in Fig. 7. Compared to the results without HDR-Net, the combination of our hierarchical network can effectively complete and refine missing or inaccurate regions over the fused model. Leveraging the geometric prior of the underlying scenes using our carefully-designed deep network, a plausible completion can be generated, filling in the holes of occluded regions (e.g. the body part partially hidden by moving arms) or wrongly-observed regions (e.g. the region with dark hair whose depth cannot be accurately measured by the sensor). Meanwhile, during the re-initialization caused by large registration error, most part of the fused model is deleted, which can be quickly fixed by our model and subsequent registration artifacts can be minimized. Tab. 1 shows the comparison of the number of re-initialization required between the systems with and without HDR-Net. When there are frequent re-initializations caused by large motions, our repair can prevent more subsequent registration artifacts and make a significant reduction in the number of re-initialization.

A close-up of the reconstructed geometry is shown in Fig. 8. The regions of the actor's head and shoulders contain holes as a result of the erroneous observation depth, while the region behind his left arm is empty because of a recent re-initialization. Both of the artifacts can be effectively fixed by our method.

The behaviour of our algorithm can be further analyzed by visualizing the expected reward computed by the global net, answering the question that what has the Global-HDR-Net learned and why it is useful in our setting. As shown in Fig. 9, Global-HDR-Net can find the place with holes and bumps efficiently and accurately. In addition, it tends to repair regions of the model's boundaries such as shoulders and feet. This is valid in the sense that the parts are more likely to move fast later and need to be refined to make the tracking more reliably. Otherwise, it will be harder to track a broken arm which we can see in Fig. 7 and lead to frequent re-initialization shown in Tab. 1. Presumably, as the input to our global net only contains the deformation nodes at the current frame, the model implicitly learns to predict the potential node motions given the static pose and jointly considers both the spatial and temporal cues when making decisions. Again, the policy is implicitly learned for lower reconstruction error, which is hard for a hand-crafted heuristic to imitate as demonstrated in Sec. 7.4.

Timing Our reconstruction system takes ∼25ms per

Fig. 10 Selected frames from the test set demonstrating the single-view performance of our framework (red) compared with [12] (blue). Our method provides results with richer features and fewer artifacts.

frame for a single-view sequence of Human10, and more time for the image pre-processing only as to the multi-view sequences, which can be paralleled if there are multiple processors. The average inference time is ∼2ms and ∼4.5ms for Local-HDR-Net and Global-HDR-Net, respectively, adding little overhead to the underlying reconstruction. This is benefited from the lightweight design of our deep models and the scalable surfel representation. In conclusion, our system can reach ∼25Hz with 5 nodes fixed per frame. Taking the parallel running of Local-HDR-Net into consideration, the process can be even faster.

7.3 Comparison with the Traditional Method

To demonstrate the advancement of our framework over traditional reconstruction method, we also reconstruct the test data with SurfelWarp [12] for each single perspective to make a comparison with our method. Fig. 10 presents several selected frames from the sequences reconstructed by both SurfelWarp and our system. The result shows our method will refine the holes and bumps effectively, which is impossible for traditional methods without the correct input. Therefore, our system can give a reconstruction result with higher-quality than traditional methods when there are lots of noise and erroneous depth observations. In addition, the broken or disconnected legs and arms will be quickly completed, which confirms our conclusion in Sec. 7.2.

7.4 Global-HDR-Net Comparison

To test the performance of Global-HDR-Net and to show that our network actually learns effective information from its training experience, we set up two competing agents executing different polices.

- *Random*: nodes are uniformly sampled from $\mathcal{G}_t$.
- *Heuristic*: we first imperially remove all the candidate nodes not satisfying the following criteria:
  - The number of surfels related to the node should be greater than 20;
  - The mean Euclidean distance from each surfel to the node should be smaller than 0.08;
  - The surfel confidence maintained within the reconstruction system, representing the point's stability and reliability, should be greater than 2.0.

  The above criteria ensure Local-HDR-Net is getting enough information for inference. Then the eligible node with fewest surfels is selected by this policy since the patch with fewer surfels should be given a higher priority for refinement.

The performance of the policies are compared using the quality of the reconstructed model computed using a two-way chamfer distance as defined in Equ. 7. Fig. 11 shows both qualitative and quantitative comparisons over two of the Human10 sequences. Results show that the manually designed heuristic policy leads to better reconstruction quality than the random policy most of the time, but the effect is not obvious and stable enough. Interestingly, we find that the random policy can sometimes lead to worse results than the simple reconstruction system without Local-HDR-Net. This is because that some randomly selected nodes may contain too much noise or have a low completeness score, whose corresponding complete geometry is too challenging for Local-HDR-Net to recover, generating many noisy outliers. In the contrast,our policy provides an effective refinement to the geometry. Clearly, choosing the correct node is as important as the geometry refinement process, which needs careful handling.

Compared to our policy, which outperforms all the baselines and is learned with the direct goal of minimizing reconstruction error, heuristic policy is the closest competitor but is unaware of the behaviour of Local-HDR-Net and the underlying reconstruction system. It is non-trivial to manually build a spatial-temporal aware criterion as analyzed in Sec. 7.2.

7.5 Local-HDR-Net Comparison

Local-HDR-Net mainly focuses on completing and refining local patch geometries. We compare it with three baselines:

- FoldingNet [57], whose decoder is designed by concatenating sampled points on a uniform grid with the global feature vector. The network learns how to deform such a uniform grid to the desired shape; surface smoothness is guaranteed.
- The Point Completion Network [60], employing a coarse-to-fine completion strategy and aggregating multiple deformable grids to assemble the final

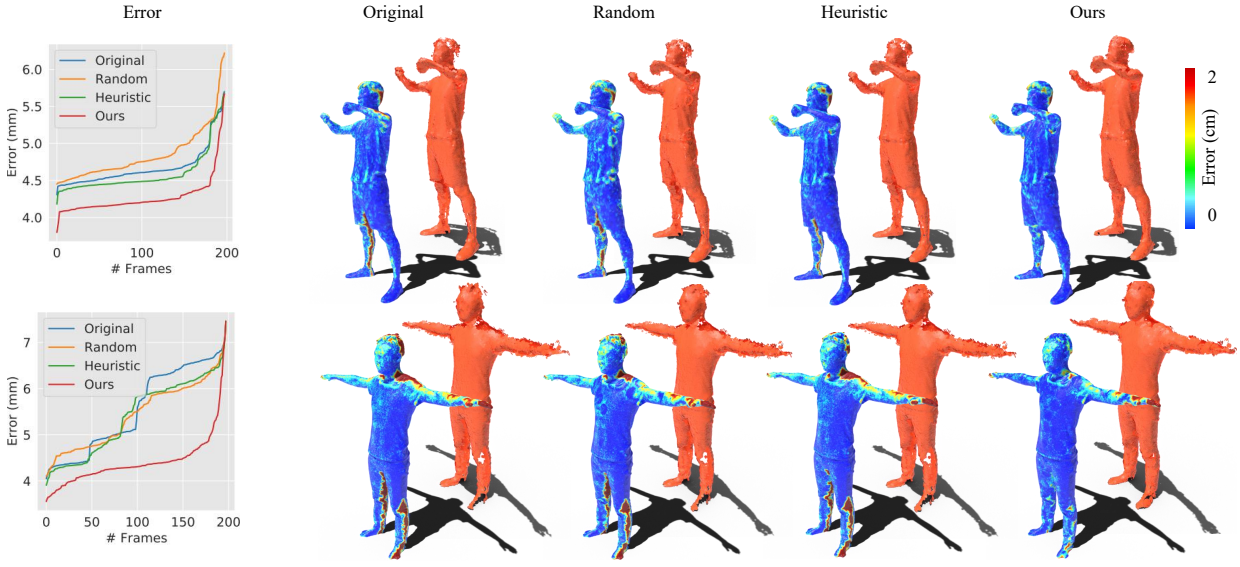

Fig. 11 Comparisons of different policies of node selection on the human0/3 from Human10. The leftmost figures plot sorted errors across the entire test sequence (i.e. the smaller the better). The error map based on the ground truth model is placed on the front of each geometry reconstructed by different policies.

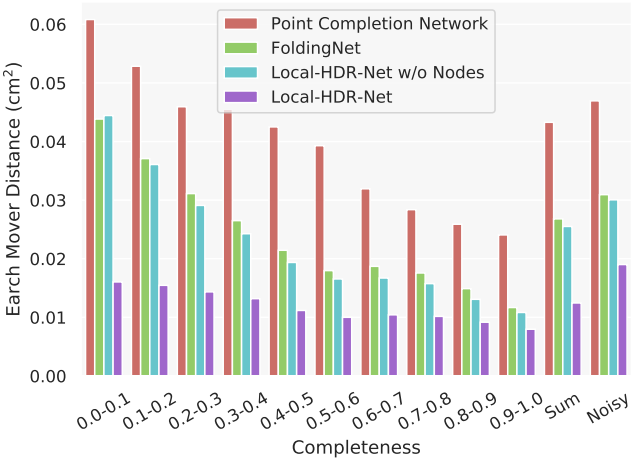

Fig. 12 Comparison with different Local-HDR-Net baselines for patches with different completeness scores. The last two columns show the average EMD loss of all the patches without and with artificial noise, respectively.

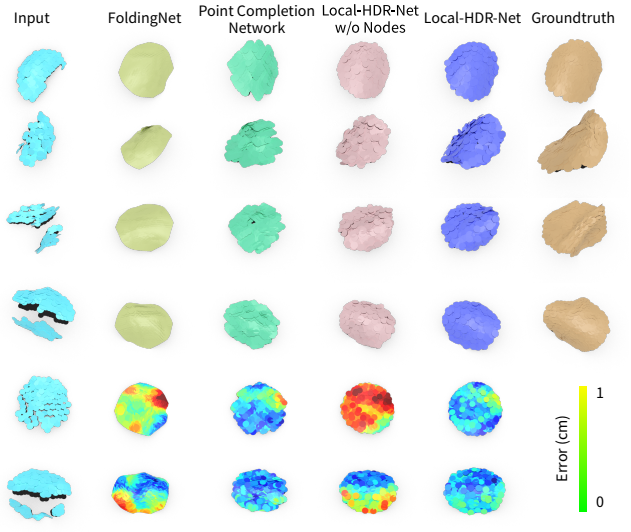

Fig. 13 Qualitative comparison to Local-HDR-Net baselines using randomly selected patches from our dataset. Error maps on the last two rows are overlaid on the predicted patch $\tilde{\mathcal{S}}_t^m$.

completed shape.

- A variant of Local-HDR-Net, lacking the branch taking in the scene deformation nodes $\mathcal{G}_t$. This variant is used to test the utility of the scene structural guidance.

All baseline models and Local-HDR-Net are trained for 200 epochs. Network hyper-parameters including model architecture and learning strategies are separately tuned for each model to give best cross-validation performance.

For evaluation we use the earth mover distance (Equ. 5) between prediction and groundtruth surfel positions as our metric. Distance error is plotted for different ranges of patch completeness (from 0.0 to 1.0) in Fig. 12. Considering that there will be a lot of noise and errors in the actual reconstruction, we also test the performance of the networks when a Gaussian noise with a standard deviation of 5mm is added to the normal direction of each surfel of the input. For patches with low completeness (< 0.1), most methods find it most challenging to infer missing fine details due to loss of information. In addition, patches with too much noise will also lead to a bad result. This also justifies our strategy of rejecting patches with too few surfels and returning a negative reward: this explicitly discourages the global net from choosing overly challenging patches for the local net.

Fig. 14 Reconstruction results of sequences from DeepDeform dataset [2] and Human10 dataset [28] (the bottom-left two). Compared with traditional method [12] (blue), ours (red) can complete objects other than human body and real scenes including interaction of people and things like bag and ball.

Compared to the baselines, our Local-HDR-Net yields much smaller distance errors in most cases and exhibits better stability in the case of noisy inputs. Fig. 13 renders results qualitatively in the form of surfels. FoldingNet [57] can generate smooth surfaces, but its uniform grid parametrization leads to bent boundaries and the deformation in complex areas is unnatural. Point Completion Network [60] assembles the surface from many smaller patch grids, resulting in overlaps and uneven distribution of surfels. It is unsuitable for small-scale geometry refinement. Our baseline without nodes completely discards the scene structural guidance, i.e. the global context, which is proved very important when the patches' information is very limited or contains much noise.

7.6 Generalization

To demonstrate the generalization of our system on real world scenes, we also select some sequences from the DeepDeform dataset [2] and the sequence human7 from Human10 dataset [28] to test our method. The results are shown in Fig. 14. It turns out that our method can also fix the artifacts and achieve better results than the traditional method [12]. Furthermore, in addition to the human body, our system can fix the artifacts of other objects as well.

7.7 Limitations

There are two typical limitations of our method: Firstly, as shown in Fig. 15(a), although our Local-HDR-Net can provide a nice refinement for most of the model parts, it still remains challenging to learn features of complicated and subtle structures like hands. A possible reason is the network is hard to present all the features of such a large batch. Narrowing the range of a single patch when the structure is complicated may gain a better result.

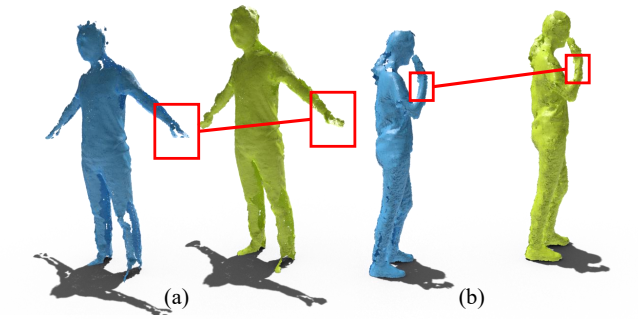

Fig. 15 Limitations of our method. In each pair we show the original reconstructed model in blue and ours in yellow.

Besides, a more powerful deep neural network with self-attention mechanisms [19] can be adopted to learn more discriminative features for point cloud completion.

Secondly, as shown in Fig. 15(b), selecting constant number of nodes per frame can lead to problems since the model's completeness is continuously changing during the reconstruction. For some models already completed enough, Global-HDR-Net will select some completed patches to 'refine', resulting in computational inefficiency and even leading to worse results. A more adaptive node selection strategy can be applied in the future by rejecting previous already chosen nodes or selecting nodes by considering a predicted score about the completeness of the nodes.

8 Conclusion

This paper has presented HDR-Net-Fusion, a novel dynamic reconstruction system using a hierarchical deep reinforcement network to improve reconstruction quality. Its applicability and effectiveness have been experimentally demonstrated using a large-scale dynamic fusion dataset. Our approach formulates the global selection of a local geometric patch for refinement in terms of reinforcement learning and uses a point-based neural network to complete and improve the local geometry. We hope this work can inspire future work pursuing better dynamic reconstruction quality using powerful deep learning and reinforcement learning algorithms.

Acknowledgements

We thank all the anonymous reviewers for their helpful comments on this paper. This work was supported by the Natural Science Foundation of China (Grant No.: 61902210, 61521002).

References

[1] K. Arulkumaran, M. P. Deisenroth, M. Brundage, and A. A. Bharath. A brief survey of deep reinforcement learning. *arXiv preprint arXiv:1708.05866*, 2017.

[2] A. Bozic, M. Zollhöfer, C. Theobalt, and M. Nießner. Deepdeform: Learning non-rigid RGB-D reconstruction with semi-supervised data. In *CVPR*, pages 7000–7010, 2020.

[3] B. J. Brown and S. Rusinkiewicz. Global non-rigid alignment of 3-d scans. *ACM Trans. Graph.*, 26(3):21, 2007.

[4] K. Chen, Y. Lai, and S. Hu. 3d indoor scene modeling from RGB-D data: a survey. *Comput. Vis. Media*, 1(4):267–278, 2015.

[5] C. G. Cifuentes, J. Issac, M. Wüthrich, S. Schaal, and J. Bohg. Probabilistic articulated real-time tracking for robot manipulation. *IEEE Robotics Autom. Lett.*, 2(2):577–584, 2017.

[6] A. Collet, M. Chuang, P. Sweeney, D. Gillett, D. Evseev, D. Calabrese, H. Hoppe, A. G. Kirk, and S. Sullivan. High-quality streamable free-viewpoint video. *ACM Trans. Graph.*, 34(4):69:1–69:13, 2015.

[7] M. Dou, P. L. Davidson, S. R. Fanello, S. Khamis, A. Kowdle, C. Rhemann, V. Tankovich, and S. Izadi. Motion2fusion: real-time volumetric performance capture. *ACM Trans. Graph.*, 36(6):246:1–246:16, 2017.

[8] M. Dou, S. Khamis, Y. Degtyarev, P. L. Davidson, S. R. Fanello, A. Kowdle, S. Orts-Escolano, C. Rhemann, D. Kim, J. Taylor, P. Kohli, V. Tankovich, and S. Izadi. Fusion4d: real-time performance capture of challenging scenes. *ACM Trans. Graph.*, 35(4):114:1–114:13, 2016.

[9] Z.-J. Du, S.-S. Huang, T.-J. Mu, Q. Zhao, R. R. Martin, and K. Xu. Accurate dynamic slam using crf-based long-term consistency. *IEEE Trans. Vis. Comput. Graph.*, pages 1–1, 2020.

[10] H. Fan, H. Su, and L. J. Guibas. A point set generation network for 3d object reconstruction from a single image. In *CVPR*, pages 2463–2471, 2017.

[11] K. Fujiwara, K. Nishino, J. Takamatsu, B. Zheng, and K. Ikeuchi. Locally rigid globally non-rigid surface registration. In *ICCV*, pages 1527–1534, 2011.

[12] W. Gao and R. Tedrake. Surfelwarp: Efficient non-volumetric single view dynamic reconstruction. In *Robotics: Science and Systems*, 2018.

[13] Z. Gojcic, C. Zhou, J. D. Wegner, L. J. Guibas, and T. Birdal. Learning multiview 3d point cloud registration. In *CVPR*, pages 1756–1766, 2020.

[14] T. Groueix, M. Fisher, V. G. Kim, B. C. Russell, and M. Aubry. A papier-mâché approach to learning 3d surface generation. In *CVPR*, pages 216–224, 2018.

[15] X. Gu, Y. Wang, C. Wu, Y. J. Lee, and P. Wang. Hplflownet: Hierarchical permutohedral lattice flownet for scene flow estimation on large-scale point clouds. In *CVPR*, pages 3254–3263, 2019.

[16] K. Guo, P. Lincoln, P. L. Davidson, J. Busch, X. Yu, M. Whalen, G. Harvey, S. Orts-Escolano, R. Pandey, J. Dourgarian, D. Tang, A. Tkach, A. Kowdle, E. Cooper, M. Dou, S. R. Fanello, G. Fyffe, C. Rhemann, J. Taylor, P. E. Debevec, and S. Izadi. The relightables: volumetric performance capture of humans with realistic relighting. *ACM Trans. Graph.*, 38(6):217:1–217:19, 2019.

[17] K. Guo, J. Taylor, S. R. Fanello, A. Tagliasacchi, M. Dou, P. L. Davidson, A. Kowdle, and S. Izadi. Twinfusion: High framerate non-rigid fusion through fast correspondence tracking. In *3DV*, pages 596–605, 2018.

[18] K. Guo, F. Xu, T. Yu, X. Liu, Q. Dai, and Y. Liu. Real-time geometry, albedo, and motion reconstruction using a single RGB-D camera. *ACM Trans. Graph.*, 36(3):32:1–32:13, 2017.

[19] M. Guo, J. Cai, Z. Liu, T. Mu, R. R. Martin, and S. Hu. PCT: point cloud transformer. *arXiv preprint arXiv: 2012.09688*, 2020.

[20] J. Huang, S. Yang, T. Mu, and S. Hu. Clustervo: Clustering moving instances and estimating visual odometry for self and surroundings. In *CVPR*, pages 2165–2174, 2020.

[21] J. Huang, S. Yang, Z. Zhao, Y. Lai, and S. Hu. Clusterslam: A SLAM backend for simultaneous rigid body clustering and motion estimation. In *ICCV*, pages 5874–5883, 2019.

[22] M. Innmann, M. Zollhöfer, M. Nießner, C. Theobalt, and M. Stamminger. Volumedeform: Real-time volumetric non-rigid reconstruction. In *ECCV (8)*, pages 362–379, 2016.

[23] L. Kavan, S. Collins, J. Zára, and C. O'Sullivan. Geometric skinning with approximate dual quaternion blending. *ACM Trans. Graph.*, 27(4):105:1–105:23, 2008.

[24] M. Keller, D. Lefloch, M. Lambers, S. Izadi, T. Weyrich, and A. Kolb. Real-time 3d reconstruction in dynamic scenes using point-based fusion. In *3DV*, pages 1–8, 2013.

[25] R. Li, X. Li, C. Fu, D. Cohen-Or, and P. Heng. PU-GAN: A point cloud upsampling adversarial network. In *ICCV*, pages 7202–7211, 2019.

[26] T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra. Continuous control with deep reinforcement learning. In *ICLR (Poster)*, 2016.

[27] X. Liu, C. R. Qi, and L. J. Guibas. Flownet3d: Learning scene flow in 3d point clouds. In *CVPR*, pages 529–537, 2019.

[28] Z. Liu, Y. Cao, Z. Kuang, L. Kobbelt, and S. Hu. High-quality textured 3d shape reconstruction with cascaded fully convolutional networks. *IEEE Trans. Vis. Comput. Graph.*, 27(1):83–97, 2021.

[29] S. Meerits, D. Thomas, V. Nozick, and H. Saito. Fusionmls: Highly dynamic 3d reconstruction with consumer-grade RGB-D cameras. *Comput. Vis. Media*, 4(4):287–303, 2018.

[30] V. Mnih, A. P. Badia, M. Mirza, A. Graves, T. P. Lillicrap, T. Harley, D. Silver, and K. Kavukcuoglu. Asynchronous methods for deep reinforcement learning. In *ICML*, volume 48, pages 1928–1937, 2016.

[31] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. A. Riedmiller, A. Fidjeland, G. Ostrovski, S. Petersen, C. Beattie, A. Sadik, I. Antonoglou, H. King, D. Kumaran, D. Wierstra, S. Legg, and D. Hassabis. Human-level control through deep reinforcement learning. *Nat.*, 518(7540):529–533, 2015.

[32] R. A. Newcombe, D. Fox, and S. M. Seitz. Dynamicfusion:

Reconstruction and tracking of non-rigid scenes in real-time. In *CVPR*, pages 343–352, 2015.

[33] R. A. Newcombe, S. Izadi, O. Hilliges, D. Molyneaux, D. Kim, A. J. Davison, P. Kohli, J. Shotton, S. Hodges, and A. W. Fitzgibbon. Kinectfusion: Real-time dense surface mapping and tracking. In *ISMAR*, pages 127–136, 2011.

[34] M. Nießner, M. Zollhöfer, S. Izadi, and M. Stamminger. Real-time 3d reconstruction at scale using voxel hashing. *ACM Trans. Graph.*, 32(6):169:1–169:11, 2013.

[35] S. Orts-Escolano, C. Rhemann, S. R. Fanello, W. Chang, A. Kowdle, Y. Degtyarev, D. Kim, P. L. Davidson, S. Khamis, M. Dou, V. Tankovich, C. T. Loop, Q. Cai, P. A. Chou, S. Mennicken, J. P. C. Valentin, V. Pradeep, S. Wang, S. B. Kang, P. Kohli, Y. Lutchyn, C. Keskin, and S. Izadi. Holoportation: Virtual 3d teleportation in real-time. In *UIST*, pages 741–754, 2016.

[36] J. J. Park, P. Florence, J. Straub, R. A. Newcombe, and S. Lovegrove. Deepsdf: Learning continuous signed distance functions for shape representation. In *CVPR*, pages 165–174, 2019.

[37] X. B. Peng, P. Abbeel, S. Levine, and M. van de Panne. Deepmimic: example-guided deep reinforcement learning of physics-based character skills. *ACM Trans. Graph.*, 37(4):143:1–143:14, 2018.

[38] C. R. Qi, H. Su, K. Mo, and L. J. Guibas. Pointnet: Deep learning on point sets for 3d classification and segmentation. In *CVPR*, pages 77–85, 2017.

[39] C. R. Qi, L. Yi, H. Su, and L. J. Guibas. Pointnet++: Deep hierarchical feature learning on point sets in a metric space. In *NIPS*, pages 5099–5108, 2017.

[40] M. Rünz and L. Agapito. Co-fusion: Real-time segmentation, tracking and fusion of multiple objects. In *ICRA*, pages 4471–4478, 2017.

[41] M. Rünz, M. Buffier, and L. Agapito. Maskfusion: Real-time recognition, tracking and reconstruction of multiple moving objects. In *ISMAR*, pages 10–20, 2018.

[42] T. Schmidt, R. A. Newcombe, and D. Fox. DART: dense articulated real-time tracking with consumer depth cameras. *Auton. Robots*, 39(3):239–258, 2015.

[43] M. Slavcheva, M. Baust, D. Cremers, and S. Ilic. Killingfusion: Non-rigid 3d reconstruction without correspondences. In *CVPR*, pages 5474–5483, 2017.

[44] M. Slavcheva, M. Baust, and S. Ilic. Sobolevfusion: 3d reconstruction of scenes undergoing free non-rigid motion. In *CVPR*, pages 2646–2655, 2018.

[45] J. Taylor, L. Bordeaux, T. J. Cashman, B. Corish, C. Keskin, T. Sharp, E. Soto, D. Sweeney, J. P. C. Valentin, B. Luff, A. Topalian, E. Wood, S. Khamis, P. Kohli, S. Izadi, R. Banks, A. W. Fitzgibbon, and J. Shotton. Efficient and precise interactive hand tracking through joint, continuous optimization of pose and correspondences. *ACM Trans. Graph.*, 35(4):143:1–143:12, 2016.

[46] L. P. Tchapmi, V. Kosaraju, H. Rezatofighi, I. D. Reid, and S. Savarese. Topnet: Structural point cloud decoder. In *CVPR*, pages 383–392, 2019.

[47] D. Tzionas and J. Gall. Reconstructing articulated rigged models from RGB-D videos. In *ECCV Workshops (3)*, pages 620–633, 2016.

[48] H. van Hasselt, A. Guez, and D. Silver. Deep reinforcement learning with double q-learning. In *AAAI*, pages 2094–2100, 2016.

[49] S. Wang, S. R. Fanello, C. Rhemann, S. Izadi, and P. Kohli. The global patch collider. In *CVPR*, pages 127–135, 2016.

[50] Y. Wang and J. Solomon. Deep closest point: Learning representations for point cloud registration. In *ICCV*, pages 3522–3531, 2019.

[51] Y. Wang, Y. Sun, Z. Liu, S. E. Sarma, M. M. Bronstein, and J. M. Solomon. Dynamic graph CNN for learning on point clouds. *ACM Trans. Graph.*, 38(5):146:1–146:12, 2019.

[52] Y. Wang, S. Wu, H. Huang, D. Cohen-Or, and O. Sorkine-Hornung. Patch-based progressive 3d point set upsampling. In *CVPR*, pages 5958–5967, 2019.

[53] Z. Wang, T. Schaul, M. Hessel, H. van Hasselt, M. Lanctot, and N. de Freitas. Dueling network architectures for deep reinforcement learning. In *ICML*, pages 1995–2003, 2016.

[54] T. Whelan, M. Kaess, M. Fallon, H. Johannsson, J. Leonard, and J. McDonald. Kintinuous: Spatially extended kinectfusion. In *RSS RGB-D Workshop*, 2012.

[55] W. Wu, Z. Qi, and F. Li. Pointconv: Deep convolutional networks on 3d point clouds. In *CVPR*, pages 9621–9630, 2019.

[56] Y. Xiao, Y. Lai, F. Zhang, C. Li, and L. Gao. A survey on deep geometry learning: From a representation perspective. *Comput. Vis. Media*, 6(2):113–133, 2020.

[57] Y. Yang, C. Feng, Y. Shen, and D. Tian. Foldingnet: Point cloud auto-encoder via deep grid deformation. In *CVPR*, pages 206–215, 2018.

[58] T. Yu, K. Guo, F. Xu, Y. Dong, Z. Su, J. Zhao, J. Li, Q. Dai, and Y. Liu. Bodyfusion: Real-time capture of human motion and surface geometry using a single depth camera. In *ICCV*, pages 910–919, 2017.

[59] T. Yu, J. Zhao, Z. Zheng, K. Guo, Q. Dai, H. Li, G. Pons-Moll, and Y. Liu. Doublefusion: Real-time capture of human performances with inner body shapes from a single depth sensor. *IEEE Trans. Pattern Anal. Mach. Intell.*, 42(10):2523–2539, 2020.

[60] W. Yuan, T. Khot, D. Held, C. Mertz, and M. Hebert. PCN: point completion network. In *3DV*, pages 728–737, 2018.

[61] Y. Zhu, R. Mottaghi, E. Kolve, J. J. Lim, A. Gupta, L. Fei-Fei, and A. Farhadi. Target-driven visual navigation in indoor scenes using deep reinforcement learning. In *ICRA*, pages 3357–3364, 2017.

[62] M. Zollhöfer, P. Stotko, A. Görlitz, C. Theobalt, M. Nießner, R. Klein, and A. Kolb. State of the art on 3d reconstruction with RGB-D cameras. *Comput. Graph. Forum*, 37(2):625–652, 2018.

Haoxuan Song is currently an undergraduate at the Department of Computer Science and Technology, Tsinghua University. His research interests include computer vision and computer graphics.

Jiahui Huang received his B.S. degree in computer science and technology from Tsinghua University in 2018. He is currently a Ph.D. candidate at the Departemnt of Computer Science and Technology, Tsinghua University. His research interests include computer vision, robotics and computer graphics.

Yan-Pei Cao received his B.S and Ph.D. degrees in computer science and technology from Tsinghua University in 2013 and 2018, respectively. He is currently a research engineer at Y-tech, Kuaishou Technology. His research interests include geometric modeling and processing, 3D reconstruction, and 3D computer vision.

Tai-Jiang Mu is currently an assistant researcher at Tsinghua University, where he received his B.S. and Ph.D. degrees in computer science and technology in 2011 and 2016, respectively. His research interests include computer vision, robotics and computer graphics.