Multi-view Neural Human Rendering

Minye Wu[1,3,4] Yuehao Wang[1] Qiang Hu[1] Jingyi Yu[1,2]
[1]ShanghaiTech University [2]DGene Inc. [3]University of Chinese Academy of Sciences
[4]Shanghai Institute of Microsystem and Information Technology
{wumy, wangyh3, huqiang, yujingyi}@shanghaitech.edu.cn

Abstract

We present an end-to-end Neural Human Renderer (NHR) for dynamic human captures under the multi-view setting. NHR adopts PointNet++ for feature extraction (FE) to enable robust 3D correspondence matching on low quality, dynamic 3D reconstructions. To render new views, we map 3D features onto the target camera as a 2D feature map and employ an anti-aliased CNN to handle holes and noises. Newly synthesized views from NHR can be further used to construct visual hulls to handle textureless and/or dark regions such as black clothing. Comprehensive experiments show NHR significantly outperforms the state-of-the-art neural and image-based rendering techniques, especially on hands, hair, nose, foot, etc.

1. Introduction

There have been tremendous demand for generating high quality 3D models of human in motion. Applications are numerous, ranging from producing ultra-realistic avatars in virtual and augmented reality [18, 40], to enabling holographic and immersive telecommunications [15] supported by latest data transmission networks. By far, most existing approaches have relied on conventional modeling and rendering pipelines: the 3D geometry of a performer is first captured using either active (e.g., depth cameras such as Microsoft Kinect) [31, 40] or passive (e.g., a multi-camera dome) [22, 28] systems and stored in the form of a 3D point cloud; the point cloud is then triangulated, texture mapped, compressed, streamed, and rendered at the viewing device. To achieve high fidelity reconstruction, dome-based systems require a large number of densely sampled cameras to handle occlusions [42], textureless regions [29], and detailed geometry (e.g., hands). Depth-camera based solutions such as Holoportation [33] are still restricted by the limited resolution. Often, a lot of manual work is needed to produce commercial quality results.

Image-based modeling and rendering (IBMR) [11, 32] attempts to interpolate new views (rays) from the sampled ones, guided by low quality reconstruction. Earlier techniques such as lumigraph [17] use coarse geometry proxy such as planes or visual hulls to select and then blend the sampled rays (images). The quality, however, relies heavily on the accuracy of the proxy. Image-based visual hull [29] bypasses 3D proxy generation using image-space ray ordering. In reality, previous IBMR methods are vulnerable to occlusions and cannot preserve fine details. To improve proxy geometry, it is also possible to fit an adjustable 3D model [5, 19]. The seminal work of Skinned Multi-Person Linear (SMPL) model [27] pre-scans 1786 human shapes and then learn a human model from them. It then estimates the shape parameters of the recovered point cloud. SMPL, however, assumes the "bare skin" model and cannot directly handle clothing or strong shape variations under complex poses. Shape deformation [31, 40] can partially mitigate the problem but are sensitive to reconstruction noise and holes.

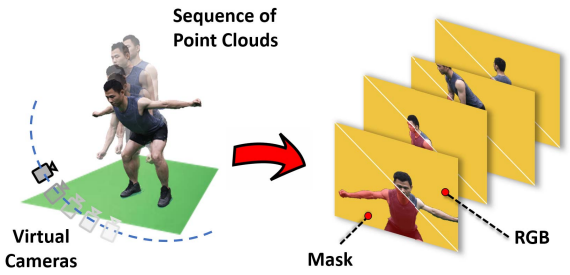

Figure 1. Our neural human renderer (NHR) produces photo-realistic free-view-video (FVV) from multi-view dynamic human captures.

In this paper, we resort to neural rendering (NR) to improve IBMR. Previous NR explores deep networks to "fix" the visual artifacts. [30, 3] exploits semantic information embedded in the captured imagery data to improve rendering. However, existing methods require using a large volume of training data, i.e., densely sampled input views. It is also possible to apply NR at the geometry stage of the classical graphics rendering pipeline, e.g., by directly refining the input 3D and texture data. [37] proposed a Neural Texture technique to handle noisy 3D geometry. Yet, it cannot handle severe defects such as holes caused by occlusions.

2575-7075/20/$31.00 ©2020 IEEE
DOI 10.1109/CVPR42600.2020.00175

In addition, nearly all existing NR techniques aim to handle static rather than dynamic models. Brute-force approaches that separately train at individual time instances are neither efficient nor practical.

We present an end-to-end Neural Human Renderer (NHR) that produces high quality rendering from low fidelity 3D point cloud of dynamic human models. NHR trains on multi-view videos and is composed of three modules: feature extraction (FE), projection and rasterization (PR), and rendering (RE). FE adopts PointNet++ [34] to extract features from the reconstructed models over time even under strong topology/reconstruction inconsistencies based on structure and semantics. More importantly, The extracted features eliminate the dense view sample requirement by exploiting temporal coherence. The PR module maps 3D features onto the target camera to form a 2D feature map where back-propagation of the gradient on the 2D map can be directly conducted on the 3D point cloud. Finally, RE renders the final image from the feature map at the new viewpoint. Specifically, RE aims to handle incomplete and noisy geometry by employing an anti-aliased CNN [41] with a gated convolution layer [39] to enhance translation equivalence.

Newly synthesized views from NHR can further improve 3D reconstruction. Specifically, we modify our pipeline to output an additional foreground human mask. Rendering a dense set of new views enables high fidelity visual hull reconstruction. In particular, the constructed visual hull from NHR supplements the MVS point cloud and efficiently tackles texture-less and/or dark regions such as black clothing. Comprehensive experiments show NHR significantly outperforms the state-of-the-art IBR techniques and can reliably handle hands, hair, nose, foot, etc that are traditionally difficult to reconstruct even under dense capture setups.

2. Related Work

Rapid advances in 3D scanning and reconstruction techniques over the past decade have laid the foundation for 3D modeling and most recently, rendering, of real humans.

Reconstruction. Passive human reconstruction schemes follow traditional reconstruction pipelines by using a large number of cameras facing towards the character. Structure-from-motion can be used to first estimate camera parameters and sparse point cloud. Multi-view stereo (MVS) can then be applied to extract a point cloud of the human subject. It is worth noting that the density and quality of the point cloud depend heavily on the availability of textures: rich textures often lead to much denser reconstruction whereas textureless or dark regions can lead to sparse and unreliable reconstruction. There are a few recent approaches that use a dome formed by stereo pairs so that each pair can obtain more reliable estimation via stereo matching to partially tackle the textureless problem. The point cloud is then triangulated to form a mesh, e.g., via Poisson Surface completion, to enable efficient texture mapping. In reality, human body imposes additional challenges besides textureless: human body exhibits complex topology and hence occlusions. Consequently, the reconstruction contains holes where brute-force surface completion produces adhesive artifacts, e.g., arms are stitched to torso, fingers are "glued" as blobs, etc. By far, even commercial solutions (e.g., 8i [1], DGene [2]) fail to produce high quality 3D reconstruction even with hundreds of cameras.

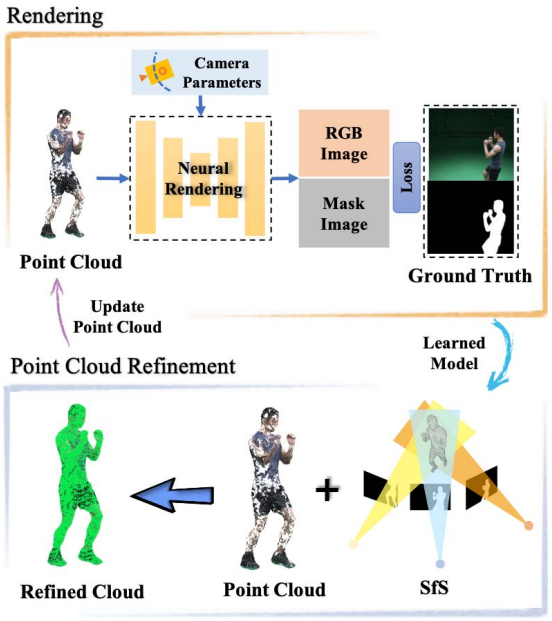

Figure 2. Our NHR pipeline employs neural rendering on spatio-temporal, low-quality 3D point clouds for multi-view rendering. § 4 introduces the rendering part. Our rendering results can then be used to further improve multi-view reconstruction by patching holes and textures. More details about refinement part are shown in § 5.

Parametric modeling. Alternative modeling schemes attempt to fit a parametric model to the acquired image or point cloud. A number of such models [5, 19, 27] exploit priors in shape, pose and appearance to estimate the optimal human geometry. A strong assumption there is the "bare-skin" model: since human clothing has strong variations that cannot be easily reduced to simple parametric models, these models unanimously require the subject to wear tight clothing. For example, the notable work of SMPL [4]. The results are reasonable, even using a video or a single image. However, the clothing restriction significantly limits the applicability of the parametric models: often it is desirable that the subject wears fancy clothing.

Rendering. One can potentially bypass the 3D reconstruction process if the goal is to render the subject as real as possible at new viewpoints, e.g., via image-based modeling and rendering (IBMR). Methods in these categories exploit coarse geometry obtained either through multi-view stereo or by even simpler methods such as shape-from-silhouette for interpolating new views from the sampled views. Geometry proxy can be as simple as a plane or as complex as parametric human shapes [12, 29, 6, 20, 7, 43, 45] and view interpolation can be efficiently implemented via view-dependent texture mapping or by an unstructured lumigraph shader [6]. In earlier days where the resolution of the display was relative low, the rendering artifacts can be "hidden" through blurs or ghosting. More recent rendering techniques based optical flow [8, 13] can partially enhance the rendering quality but still produce noticeable visual artifacts near occlusion boundaries.

Our method exploits neural rendering [3, 30] that has shown promising results on image synthesis. Different from IBMR, NR attempts to learn from sampled images to mitigate visual artifacts. Methods based on GAN [16] learn the distribution of images and produce impressive results on several image generation tasks such as denoising, deblurring [24, 35], super-resolution, etc. We set out to use NR to bridge the gap between low quality 3D reconstruction and high quality image synthesis for dynamic 3D humans. For static scenes, NR can also be used in conjunction with classical IBR to achieve view-dependent rendering [10, 38, 37], image-based relighting [30], mesh denoising [37], and correspondence matching at both the voxel level [36] and the point level [3].

Closely to our approach is the recent Generative CNN models that aim to synthesize body appearance [14, 26], body articulation [9], or both [44, 25]. Their techniques can fix artifacts in captured 3D performances [28] and enhance low quality 3D face reconstruction [23]. A major difference of our technique is that we tackle dynamic models by exploiting temporal shape variations to compensate for sparsity in viewpoint sampling: rich body shape and appearance variations over time compensates for the lack of sufficient viewpoints using shared weights training. We also demonstrate how to use the rendered results to further improve reconstruction.

3. Approach Overview

Before proceeding, we explain our notations. We assume the multi-view stereo (MVS) input, although active 3D sensing can also fit naturally into the pipeline by bypassing the reconstruction process. The inputs to our NHR pipeline consists of a synchronized, multi-view video sequence $\mathcal{I}_t = \{I_t^c\}_{c=1,t=1}^{n_c,n_t}$ towards a performer, where c is the camera index, n_c is the total number of cameras, t is the frame index, n_t is the total number of frames. We assume the intrinsics and extrinsics at each camera c are known as $\{K^c\}_{c=1}^{n_c}$ and $\{T^c\}_{c=1}^{n_c}$. We also extract the human foreground mask $\mathcal{M}_t = \{M_t^c\}_{c=1,t=1}^{n_c,n_t}$ at all frames to facilitate training. Under the MVS setting, we can construct a point cloud at each frame $\mathcal{P} = \{P_t\}$. We also assume each point in the point cloud has color, computed through reprojection on the input views' image as $\mathcal{Y} = \{Y_t\}$.

The first task in NHR is to synthesize high quality new views through the rendering process (§ 4). In addition to RGB color rendering, we also produce a foreground mask that later facilitates model refinement (§ 5). Specifically, the initial point cloud sequence $\mathcal{P}$ is not only noisy but also contains many holes due to occlusions. The model refinement process can effectively fill in these holes in the synthesized new views that can be used to further improve rendering. Fig. 2 introduces the complete pipeline of our iterative render-model technique, with both rendering and geometry refinement modules where the former is illustrated in details in Fig. 3 and the latter in § 5.

4. NHR Rendering

4.1. The Rendering Process

The NHR rendering process consists of three modules: Feature Extraction (FE), projection and rasterization (PR), and Rendering (RE).

Feature Extraction. Previous neural rendering on point cloud requires learning a feature descriptor at each 3D point beyond its original RGB color. Different from a static 3D model, we observe under our dynamic human capture setting, the recovered point cloud at each time instance is different in point number and density, as the reconstruction is determined by the MVS technique. As a result, such inconsistency introduces additional challenges: learning a feature descriptor at each point at each time instance is computationally expensive and requires a significant amount of storage. In addition, the number of view cameras is relatively small and therefore there are limited samples for learning the descriptor. We instead set out to use all images at all time instances. In particular, we exploit semantic features in human shape and their coherence over time. These features are learned from end-to-end supervision.

Specifically, PointNet++ can effectively serve as a feature extractor. We observe that under the multi-view setting, the appearance at different viewpoints can exhibit variations due to lighting directions, cloth materials, skin reflectance, etc. Therefore, we also consider view direction in the FE process to mitigate the view dependency effects. At the same time, we impose the recovered color of the 3D points as prior. Eqn 1 shows the FE process:

$$D_t = \psi_{\text{fe}}(\varphi_{norm}(P_t), \{Y_t, V\}) \tag{1}$$

where ψ_{fe} corresponds to PointNet++. In our implementa-

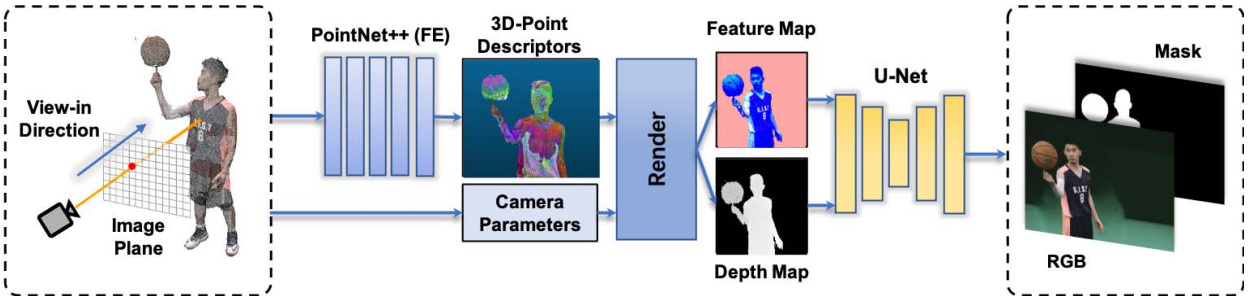

Figure 3. Our NHR consists of three modules: Feature Extraction (FE) based on PointNet++ to process spatio-temporal point cloud sequences, Projection and Rasterization (PR) for feature projection, and U-Net based renderer for feature decoding. All together they form a differentiable renderer.

tion, we remove the classifier branch in the original network to keep only the segmentation branch as the FE branch. It takes the point cloud and its features at each moment as input to obtain a feature descriptor D_t. $V = \{v^i\}$ represents the (normalized) view direction towards a point as $v^i = \frac{p_t^i - o}{\|p_t^i - o\|_2}$, where o is the center of projection (CoP) of the target view camera. $\{\cdot\}$ represents concatenation that concatenates the color and the normalized view direction of the point as the initial point attributes (or features) feeding into ψ_{fe}. Point coordinates are standard normalized by using $\varphi_{norm}(\cdot)$.

Projection and Rasterization. Once we obtain the feature descriptor D of the point cloud, we set out to synthesize new views. Given a target camera with intrinsic and extrinsic $\hat{K}$ and $\hat{T}$, we project the point cloud onto the camera and splat points into pixel coordinates on image plane linearly. This step rasterizes points into pixel squares. We use the Z-buffer to maintain correct depth ordering and hence occlusions. This produces a projected 2D feature map S: $S_{x,y} = d_t^i$, where d_t^i is feature descriptor of the $i's$ point p_t^i in P_t, preserved after z-buffer depth ordering onto pixel (x, y). We assign a learnable default feature vector θ_d for each background pixel. The complete PR process ψ_{pr} for producing the 2D feature map S can be described as:

$$S, E = \psi_{\text{pr}}(P_t, D_t, \hat{K}, \hat{T}, \theta_d) \tag{2}$$

where E is the depth map in current view.

Rendering. The feature map S produced above provides an encoding of the target new view. In the final rendering stage (RE), we use convolutional neural networks (CNN) to decode S into its corresponding RGB image and a foreground mask.

We benefit from the recent U-Net architecture which has shown great success in image denoising, deblurring, and style transfer applications. In our application, we use U-Net to decode S. Notice that the point cloud from MVS is sparse and the projected feature maps contain holes and even exhibits see-throughs where the foreground points are missing. We treat these artifacts as semantic noise. When using NR, our goal is to remove these incorrect pixels and therefore we adopt the gated convolution [39] in place of the convolutional layer in U-Net. Specifically, our goal is to have the network identifies the location of these semantic noises through training and then exploit the attention mask mechanism to correct the feature maps at the convolution layer.

Recall that the depth map E generated from PR contains rich geometric information of the scene. In particular, abrupt changes in depth values is a strong indicator of semantic noise, especially for low depth values. Therefore, we use both S and standard normalizated depth map $\hat{E}$ as input to the RE network, to reduce semantic noises.

It is critical to note that our NHR aims to render the human subject from any view direction, i.e., the human subject can appear at any location within an image. This implies that the neural render should maintain translation equivalence for the feature map S. In our implementation, we use MaxBlurPool and ConvBlurPool [41] to replace the downsampling operations in the original U-Net (including the pooling layer and the convolutional layer with stride), to mitigate incoherence caused by translation of the target camera.

ψ_{render} represents our revised U-Net. The final layer of ψ_{render} outputs an image with four channels, the first three produces an RGB image I^* and last produces the foreground human mask M^* under Sigmoid.

$$I^*, M^* = \psi_{\text{render}}(S, \hat{E}) \tag{3}$$

4.2. Network Training

To acquire training data, we use a multi-camera dome system composed of up to 80 synchronized industrial high resolution cameras. We call these cameras sample cameras the same as traditional image-based rendering. The dome uses a green screen to facilitate easy foreground segmentation. All cameras face inwards towards the performer although most captures can only capture a part rather than the complete image of the performer, as discussed in § 6. All

cameras are pre-calibrated.

For training, we set one of the sample cameras as the target camera. This allows us to use the ground truth I_t^c and M_t^c to conduct supervised training. As described in § 4.1, our end-to-end networks updates the parameters by back-propagating the gradients of the loss function from 2D image to 3D point cloud. Recall our goal is to render the target view as photo-realistic as possible, we therefore adopt the perceptual loss [21] and L1 loss as the loss function as:

$$\begin{aligned}\mathcal{L}(\theta_{pn},\theta_{render},\theta_d) = \\ \sum_{i=1}^{n_b}[\lambda\cdot(\|I_i^*-I_i\|_1+\|M_i^*-M_i\|_1)+ \\ (1-\lambda)\cdot(\|\psi_{\text{vgg}}(I_i^*)-\psi_{\text{vgg}}(I_i)\|_2+ \\ \|\psi_{\text{vgg}}(M_i^*)-\psi_{\text{vgg}}(M_i)\|_2)]\end{aligned} \tag{4}$$

where n_b is the batch size; I_i^* and M_i^* are the ith rendered output image and mask in the mini-batch; $\psi_{\text{vgg}}(\cdot)$ extracts feature maps from the 2th and 4th layer of the VGG-19 network pretrained on the ImageNet dataset.

Since the dome system consists of rather limited sample cameras. To train the network for better adaptation at arbitrary viewpoints, we further augment the training data by 2D image transformations. Specifically, we adopt three types of transforms, random translation, random scaling, and random rotation, that can be easily achieved by modifying the camera intrinsic/extrinsic parameters and then re-rendering the 3D point cloud.

Conceptually, one can adopt two training methods, individual training and shared training, based on the type of input data. The former trains on each individual performer. Such an approach is suitable when only a small number of performers have been captured or when we need to fine-tune the network for a specific performer. The latter trains on a large number of performers there the training process shares the same network weight of ψ_{render} but produces separate weights in FE. This allows the FE module to learn the unique geometric and appearance characteristics of individual performers while maintaining a unified feature embedding space. The shared rendering module further decodes the feature descriptors onto target images.

In our implementation, we bootstrap our network using shared training and then fine-tune the network for each performer using individual training. For a new performer, we first fix ψ_{render}, and train FE from scratch using shared training. After 5 epochs, we conduct individual training. This strategy significantly accelerates the training process.

5. Geometry Refinement

Even dense MVS setups produce patches of holes on textureless or occluded regions. Our NHR can virtually patch these holes and produce satisfactory results at every time instance. However, we observe that when rendering a video sequence, the results produce flickering artifacts at the synthesized regions that originally correspond to holes, even though every individual frame produces reasonable results. Similar artifacts have been observed in previous NR techniques [3]. We mitigate the problem by reducing the holes via geometry refinement.

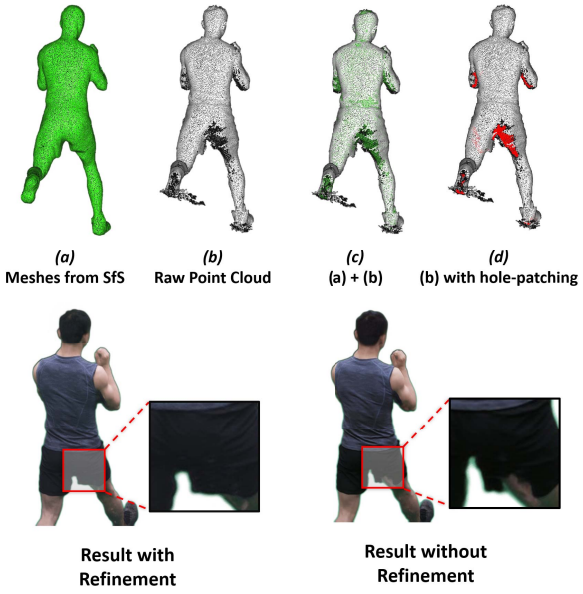

Figure 4. Geometry refinement using NHR. (a) shows the visual hull results (using Shape-from-Silhouette) from densely rendered views using NHR. (b) shows the raw 3D reconstruction using SfM. (c) illustrates high coherence on the visual hull result and SfM geometry. (d) patches holes in (b) using (a). Bottom row shows closeup views of the NHR results with and without geometry refinement.

Recall that our NHR also produces an auxiliary human mask at each new view, we therefore resort to the visual hull approach. We observe, compared with RGB images that contain rich details and thus noise, the masks generated by our networks are much cleaning. To refine geometry, we render a dense set of new views and use the resulting masks as silhouettes and conduct visual hull reconstruction based on space-carving or shape-from-silhouettes (SfS). We can then use the approximated visual hull to patch the holes.

Mask and Shape Generation. To briefly reiterate, we use the MVS point cloud to train the rendering module to output a matte analogous to RGB images. We then render masks at a uniformly sampled set of new viewpoints towards the performer, each with known camera parameters. Specifically, we render the masks at a resolution of 800×600. Next, we conduct voxel-based SfS to reconstruct human mesh.

Points Sampling and Colorization. Notice that the SfS results $\hat{P}_t$ only contains geometry but not color. For each

point $\hat{p}_t^i$ in $\hat{P}_t$, we can further compute its color by using its nearest point in the MVS point cloud P_t. We hence obtain $\hat{Y}_t$ corresponding to $\hat{P}_t$.

Holes Patching. Although we can directly use the SfS results as the refined geometry, it is well-known that volumetric reconstruction is restricted by its resolution and the number of input views. Further the shape recovered from silhouettes generally appear polygonal. Therefore, we only patch the holes U_t in P_t from $\hat{P}_t$, i.e., $U_t \subset \hat{P}_t$. Specifically, assume $\phi(\cdot,\cdot)$ represents the Euclidean distance between two 3D points. Our holes patching scheme is based on the observation that for each point $u_t^i \in U_t$, $\phi(u_t^i, p_t^j)$ to its nearest point p_t^j in P_t is generally bigger than the points in $\hat{P}_t - U_t$. We hence adopt a statistical approach to find U_t.

Let $b_t^i = min\{\phi(\hat{p}_t^i, p_t^j)\}$, we set a threshold τ_1 as:

$$\tau_1 = \lambda_t \cdot max(b_t^i) + (1 - \lambda_t) \cdot median(b_t^i) \qquad (5)$$

where λ_t is the weighting factor and is set to 0.2 in all our experiments.

Next, we count the number of points in $\hat{P}_t$ whose distance to p_t^j is below τ_1 as s_t^i, where $s_t^i = \#\{b_t^i | b_t^i < \tau_1\}$. Conceptually, s_t^i is inversely proportional to the probability of the point belonging to U_t.

Next, we compute the histogram of s_t^i for all the points in set $\hat{P}_t$ using 15 bins, evenly separated by the maximal distance value. We observe in all cases the first bin contains significantly more points than the second and therefore can directly help to identify the closest points. We thus use the maximal distance in the first bin as a threshold τ_2 for selecting U_t as:

$$U_t = \{\hat{p_t}^j | s_t^j < \tau_2, \hat{p_t}^j \in \hat{P}_t\} \qquad (6)$$

Fig. 4 shows that refined geometry using our SfS based hole patching technique greatly reduces flickering when changing viewpoints. It is worth noting that the final geometry may still exhibit artifacts as its quality depends on the reliability of τ_1 and τ_2.

6. Experimental Results

All experiments are conducted on 3D dynamic human data collected by a multi-camera dome system with up to 80 cameras arranged on a cylinder. All cameras are synchronized and capture at a resolution of 2048×1536 at 25 frames per second. In this paper, we use 5 sets of datasets where the performers are in different clothing and perform different actions. All sequences have a length between 8 to 24 seconds. Specifically sport1, sport2, sport3 correspond to dumbbell lifting with relatively tight clothing, dance contains complex and highly deformable clothing, and basketball involves interactions between a player and a ball. We use chrome key based matting followed by manual fixing to extract the ground truth masks for all views. 3D reconstruction is not the main focus of the paper and we use one of the best commercial SfM software Metashape to compute the initial 3D point clouds for all frames.

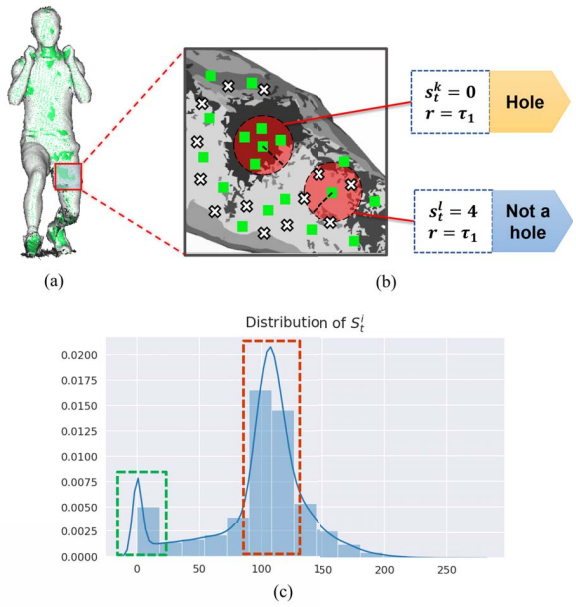

Figure 5. Our statistical hole patching techniques explores using the visual hull results to patch holes in terms of their distance distribution as discussed in § 5.

In our network training and forward prediction steps, we set the target rendering resolution to be 800×600. The FE module extracts features vectors of 24 dimensions.

Comparisons. We compare our NHR with a number of conventional and NR approaches.

Ground Truth (GT). We resize captured image data to a resolution of 800×600, which is same as the output resolution of our network.

Point Cloud Rendering (PCR). We directly project the recovered color point clouds onto the target camera and then use our PR module to render the projected pixels to form the final RGB image sequences.

Textured Mesh (TM). We use Metashape to triangulate the point clouds and construct the texture maps for all frames. The results are rendered onto the image via standard rasterization.

PCR + U-net (PCR-U). We project RGB point cloud into target view and feed it into the U-Net directly to refine the rendering results.

NHR with Geometry Refinement (NHR w GR). We first refine geometry as described in § 5 and then use the refined point cloud to re-train the network with 3 epochs.

To verify the effectiveness of the proposed method, we compare the rendering results from various algorithms to

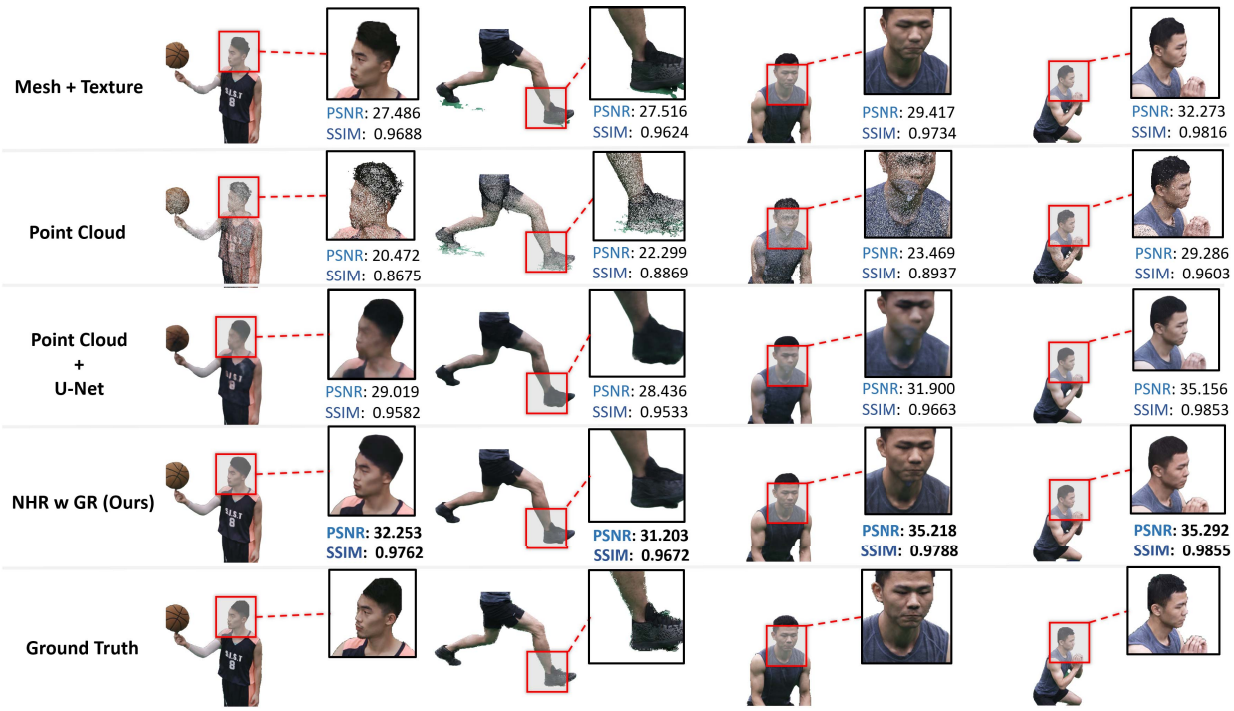

Figure 6. Comparisons on rendering quality using different approaches. Our NHR manages to handle challenging cases such as hair, hands, foot, ball where 3D reconstruction methods fail. Additional results can be found in the supplementary materials and videos.

	sport1	sport2	sport3	dance	basketball
PCR	17.458 / 0.827 / 0.025	20.002 / 0.855 / 0.013	19.760 / 0.849 / 0.015	20.664 / 0.890 / 0.014	19.463 / 0.839 / 0.015
PCR-U	25.104 / 0.963 / 0.003	25.654 / 0.968 / 0.003	26.006 / 0.969 / 0.003	22.796 / 0.967 / 0.009	24.130 / 0.964 / 0.004
TM	22.419 / 0.957 / 0.007	22.103 / 0.954 / 0.007	22.318 / 0.956 / 0.007	20.749 / 0.957 / 0.011	21.947 / 0.955 / 0.007
IBR	22.632 / 0.960 / 0.006	22.369 / 0.958 / 0.006	22.644 / 0.961 / 0.006	20.269 / 0.961 / 0.014	22.120 / 0.961 / 0.007
NHR w/o GR	26.951 / 0.975 / **0.002**	26.713 / 0.975 / **0.002**	**27.218 / 0.976 / 0.002**	22.602 / 0.969 / 0.009	24.660 / 0.969 / 0.004
NHR w GR	**27.385 / 0.977** / 0.002	**26.925 / 0.979** / 0.002	26.889 / 0.975 / 0.002	**23.367 / 0.973 / 0.007**	**25.138 / 0.971 / 0.004**

Table 1. Quantitative comparison. PSNR/SSIM/MSE of each approach on different datasets compared with ground truth are listed in this table.

GT. For fairness, we only compare the rendering of the foreground. Since NHR already predicts a mask at each target view, we can simply use the result to segment the NHR rendered foreground. For other techniques such as PCR and IBR, the foreground can be directly separated from the background using the mesh.

Fig. 6 compares the rendering appearance of rich textures and shapes (silhouettes) using different approaches. Compared with PCR and PCR-U, using the same point cloud as input, NHR manages to correct many visual artifacts as well as preserve fine texture and geometric details. The use of U-Net in PCR-U can partially reduce noise and patch holes in the final rendering but its result exhibits excessive blurs at these regions. In contrast, NHR exhibits much less blur. This is because our network is specifically tailored to extract spatial features consistent across the temporal sequences whereas most existing U-Net solutions are designed for static meshes. In other words, our technique can infer missing geometric information at a specific frame from other frames in the dynamic sequence. The advantages of our approach is particularly obvious on heavily corrupted 3D geometry such as missing noses, fingers, holes on the body, etc, commonly observed in TM as shown in Fig. 6. In fact, even when the GT mask contains errors such as zigzagged or broken edges, our network manages to fix the mask and produces even better quality masks. Fig. 6 shows the quantitative comparisons. NHR uniformly outperforms the rest of the methods in PSNR and SSIM. Quantitative results with other methods, including PCR, TM, PCR-U and Image-based Rendering(IBR) [6], are listed in Table 1.

The use of geometry refinement (GR) procedure further improves the model and rendering quality by filling in holes caused by occlusions as shown in Fig. 4. This is particularly important to avoid flickering in NR-based rendering: with holes filled, the renderer can correctly handle depth ordering, avoiding undesirable see-through artifacts. In previous

Figure 7. Free View Video results on a challenging dance scene using NHR. The red blouses imposes significant challenges in 3D reconstruction. Our NHR can produce high quality FVV rendering from poor reconstructions.

NRs, see-through is prevented in the image space and therefore when the viewpoint changes, the "patched" part can exhibit strong variance, causing flickering. By filling in the holes, such flickering is significantly reduced.

Other Applications. View synthesis on dynamic humans can lead to a range of new rendering techniques. For example, the bullet-time effect which is an amazing stopping-time illusion. However, currently available dome system does not suffice the need of movie quality production. Since our NHR manages to reduce or even eliminate strong visual artifacts, it can be used to produce bullet-time effects. In addition to regular bullet-time, i.e., fixing the time but changing the viewpoint, we can use NHR to simultaneously change time and viewpoint, i.e., as the viewer changes the perspective, the perform continues his/her movement. Demonstrations is shown in Fig. 7.

Since our extracted features using FE module preserve spatio-temporal coherence over the sequence, they can be potentially used to generate an animation mesh (AM) from the point cloud sequence. Fig. 8 illustrates color coded features at different frames within the sequence. These features exhibit strong semantic coherence, despite that the point clouds are individually reconstructed at each frame without coherence. This implies effectiveness and potential usefulness of the FE module for correspondence matching.

As NHR only render single human image, we can composite a multi-user scene image by integrating NHRs' results. For this purpose, we train individual NHR models for each person. Next, we render each human instance in the same target view. Then we can use depth maps and human masks to infer their visibility of RGB images in final result image and blend together.

Limitations. Chroma segmentation used for GT mask generation can easily lead to over- or under-segmentation. Such artifacts cause network training to wrongfully assume the green background as foreground and produce greenish visual artifacts. To render a novel view, our technique projects the recovered noisy point cloud on to the image.

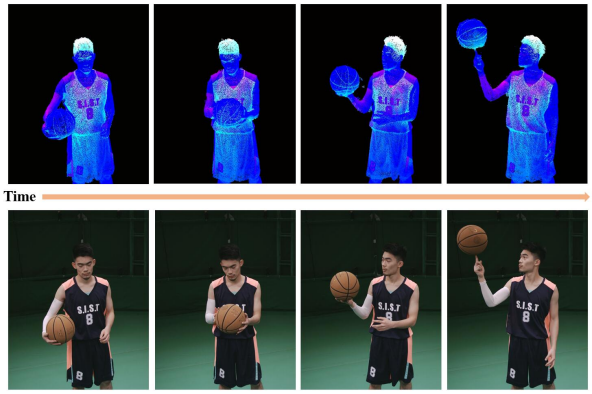

Figure 8. We visualize the FE features from the point cloud sequence in color. They exhibit high semantic coherence that can be used for correspondence matching and subsequently constructing animated meshes.

As a result, if we move the camera really close to the object, the projected points can be rather sparse, i.e., with large gaps between the points. The problem then becomes very ill-posed with strong ambiguity and our technique can fail.

7. Conclusions and Future Work

We have presented a novel Neural Human Renderer (NHR) for high quality rendering of dynamic 3D human models captured under the multi-view dome system. While most existing neural rendering (NR) techniques have been focused on static scenes or objects, NHR explicitly seeks temporal correspondences to compensate for sparsity in spatial/angular sampling. By exploiting PointNet++ [34] learned over time, our approach manages to establish and subsequently uses spatio-temporal 3D correspondences to significantly improve the rendering quality even with poor 3D reconstructions. In particular, NHR has shown superior performance on hair, hands, nose, foot, etc, that are very difficult to correctly reconstruct even using very densely sample cameras or using active 3D sensors.

We have further demonstrated using the NHR synthesized new views to further improve 3D reconstruction via shape-from-silhouette. In the future, we plan to explore the possibility of directly fixing the point cloud geometry using the spatio-temporal sequences via deep learning so that the modeler and the renderer can be seamlessly integrated into a unified, end-to-end solution.

Acknowledgement

This work is supported by the National Key Research and Development Program (2018YFB2100500), the programs of NSFC (61976138 and 61977047), STCSM (2015F0203-000-06), and SHMEC (2019-01-07-00-01-E00003).We gratefully acknowledge the support of DGene Inc. with the help of human 3D capture.

References

[1] 8i. https://www.8i.com/.

[2] Dgene. https://www.dgene.com/cn/.

[3] Kara-Ali Aliev, Dmitry Ulyanov, and Victor S. Lempitsky. Neural point-based graphics. *CoRR*, abs/1906.08240, 2019.

[4] Thiemo Alldieck, Marcus Magnor, Weipeng Xu, Christian Theobalt, and Gerard Pons-Moll. Video based reconstruction of 3d people models. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 8387–8397, 2018.

[5] Dragomir Anguelov, Praveen Srinivasan, Daphne Koller, Sebastian Thrun, Jim Rodgers, and James Davis. Scape: shape completion and animation of people. In *Acm Siggraph*, 2005.

[6] Chris Buehler, Michael Bosse, Leonard McMillan, Steven Gortler, and Michael Cohen. Unstructured lumigraph rendering. In *Proceedings of the 28th annual conference on Computer graphics and interactive techniques*, pages 425–432. ACM, 2001.

[7] Joel Carranza, Christian Theobalt, Marcus A Magnor, and Hans-Peter Seidel. *Free-viewpoint video of human actors*, volume 22. ACM, 2003.

[8] Dan Casas, Christian Richardt, John Collomosse, Christian Theobalt, and Adrian Hilton. 4d model flow: Precomputed appearance alignment for real-time 4d video interpolation. In *Computer Graphics Forum*, volume 34, pages 173–182. Wiley Online Library, 2015.

[9] Caroline Chan, Shiry Ginosar, Tinghui Zhou, and Alexei A Efros. Everybody dance now. *arXiv preprint arXiv:1808.07371*, 2018.

[10] Anpei Chen, Minye Wu, Yingliang Zhang, Nianyi Li, Jie Lu, Shenghua Gao, and Jingyi Yu. Deep surface light fields. *Proceedings of the ACM on Computer Graphics and Interactive Techniques*, 1(1):14, 2018.

[11] Paul Debevec, Yizhou Yu, and George Borshukov. Efficient view-dependent image-based rendering with projective texture-mapping. In *Rendering Techniques 98*, pages 105–116. Springer, 1998.

[12] Paul Ernest Debevec, Camillo J Taylor, and Jitendra Malik. *Modeling and rendering architecture from photographs*. University of California, Berkeley, 1996.

[13] Ruofei Du, Ming Chuang, Wayne Chang, Hugues Hoppe, and Amitabh Varshney. Montage4d: Interactive seamless fusion of multiview video textures. 2018.

[14] Patrick Esser, Ekaterina Sutter, and Björn Ommer. A variational u-net for conditional appearance and shape generation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 8857–8866, 2018.

[15] Ingo Feldmann, Wolfgang Waizenegger, Nicole Atzpadin, and Oliver Schreer. Real-time depth estimation for immersive 3d videoconferencing. In *2010 3DTV-Conference: The True Vision-Capture, Transmission and Display of 3D Video*, pages 1–4. IEEE, 2010.

[16] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *Advances in neural information processing systems*, pages 2672–2680, 2014.

[17] Steven J Gortler, Radek Grzeszczuk, Richard Szeliski, and Michael F Cohen. The lumigraph. In *Siggraph*, volume 96, pages 43–54, 1996.

[18] Marc Habermann, Weipeng Xu, Michael Zollhoefer, Gerard Pons-Moll, and Christian Theobalt. Livecap: Real-time human performance capture from monocular video. *ACM Transactions on Graphics (TOG)*, 38(2):14, 2019.

[19] N Hasler, C Stoll, M Sunkel, B Rosenhahn, and H. P Seidel. A statistical model of human pose and body shape. *Computer Graphics Forum*, 28(2):337–346, 2010.

[20] Benno Heigl, Reinhard Koch, Marc Pollefeys, Joachim Denzler, and Luc Van Gool. Plenoptic modeling and rendering from image sequences taken by a hand-held camera. In *Mustererkennung 1999*, pages 94–101. Springer, 1999.

[21] Justin Johnson, Alexandre Alahi, and Li Fei-Fei. Perceptual losses for real-time style transfer and super-resolution. In *European conference on computer vision*, pages 694–711. Springer, 2016.

[22] Takeo Kanade, Peter Rander, and PJ Narayanan. Virtualized reality: Constructing virtual worlds from real scenes. *IEEE multimedia*, 4(1):34–47, 1997.

[23] Hyeongwoo Kim, Pablo Carrido, Ayush Tewari, Weipeng Xu, Justus Thies, Matthias Niessner, Patrick Pérez, Christian Richardt, Michael Zollhöfer, and Christian Theobalt. Deep video portraits. *ACM Transactions on Graphics (TOG)*, 37(4):163, 2018.

[24] Orest Kupyn, Volodymyr Budzan, Mykola Mykhailych, Dmytro Mishkin, and Jiří Matas. Deblurgan: Blind motion deblurring using conditional adversarial networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 8183–8192, 2018.

[25] Lingjie Liu, Weipeng Xu, Michael Zollhoefer, Hyeongwoo Kim, Florian Bernard, Marc Habermann, Wenping Wang, and Christian Theobalt. Neural rendering and reenactment of human actor videos. *ACM Transactions on Graphics (TOG)*, 38(5):1–14, 2019.

[26] Stephen Lombardi, Tomas Simon, Jason Saragih, Gabriel Schwartz, Andreas Lehrmann, and Yaser Sheikh. Neural volumes: Learning dynamic renderable volumes from images. *ACM Transactions on Graphics (TOG)*, 38(4):65, 2019.

[27] Matthew Loper, Naureen Mahmood, Javier Romero, Gerard Ponsmoll, and Michael J Black. Smpl: A skinned multi-person linear model. *Acm Transactions on Graphics*, 34(6):248, 2015.

[28] Ricardo Martin-Brualla, Rohit Pandey, Shuoran Yang, Pavel Pidlypenskyi, Jonathan Taylor, Julien Valentin, Sameh Khamis, Philip Davidson, Anastasia Tkach, Peter Lincoln, et al. Lookingood: enhancing performance capture with real-time neural re-rendering. *ACM Transactions on Graphics (TOG)*, 37(6):1–14, 2018.

[29] Wojciech Matusik, Chris Buehler, Ramesh Raskar, Steven J Gortler, and Leonard McMillan. Image-based visual hulls. In *Proceedings of the 27th annual conference on Computer graphics and interactive techniques*, pages 369–374. ACM Press/Addison-Wesley Publishing Co., 2000.

[30] Moustafa Meshry, Dan B Goldman, Sameh Khamis, Hugues Hoppe, Rohit Pandey, Noah Snavely, and Ricardo Martin-Brualla. Neural rerendering in the wild. In *Proceedings*

of the IEEE Conference on Computer Vision and Pattern Recognition, pages 6878–6887, 2019.

[31] Richard A Newcombe, Shahram Izadi, Otmar Hilliges, David Molyneaux, David Kim, Andrew J Davison, Pushmeet Kohli, Jamie Shotton, Steve Hodges, and Andrew W Fitzgibbon. Kinectfusion: Real-time dense surface mapping and tracking. In *ISMAR*, volume 11, pages 127–136, 2011.

[32] Manuel M Oliveira. Image-based modeling and rendering techniques: A survey. *Rita*, 9(2):37–66, 2002.

[33] Sergio Orts-Escolano, Christoph Rhemann, Sean Fanello, Wayne Chang, Adarsh Kowdle, Yury Degtyarev, David Kim, Philip L Davidson, Sameh Khamis, Mingsong Dou, et al. Holoportation: Virtual 3d teleportation in real-time. In *Proceedings of the 29th Annual Symposium on User Interface Software and Technology*, pages 741–754. ACM, 2016.

[34] Charles Ruizhongtai Qi, Li Yi, Hao Su, and Leonidas J Guibas. Pointnet++: Deep hierarchical feature learning on point sets in a metric space. In *Advances in neural information processing systems*, pages 5099–5108, 2017.

[35] Sainandan Ramakrishnan, Shubham Pachori, Aalok Gangopadhyay, and Shanmuganathan Raman. Deep generative filter for motion deblurring. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2993–3000, 2017.

[36] Vincent Sitzmann, Justus Thies, Felix Heide, Matthias Nießner, Gordon Wetzstein, and Michael Zollhofer. Deepvoxels: Learning persistent 3d feature embeddings. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2437–2446, 2019.

[37] Justus Thies, Michael Zollhöfer, and Matthias Nießner. Deferred neural rendering: Image synthesis using neural textures. *ACM Transactions on Graphics (TOG)*, 38(4):1–12, 2019.

[38] Justus Thies, Michael Zollhöfer, Christian Theobalt, Marc Stamminger, and Matthias Nießner. Ignor: Image-guided neural object rendering. *arXiv preprint arXiv:1811.10720*, 2018.

[39] Jiahui Yu, Zhe Lin, Jimei Yang, Xiaohui Shen, Xin Lu, and Thomas S Huang. Free-form image inpainting with gated convolution. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 4471–4480, 2019.

[40] Tao Yu, Zerong Zheng, Kaiwen Guo, Jianhui Zhao, Qionghai Dai, Hao Li, Gerard Pons-Moll, and Yebin Liu. Doublefusion: Real-time capture of human performances with inner body shapes from a single depth sensor. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 7287–7296, 2018.

[41] Richard Zhang. Making convolutional networks shift-invariant again. *arXiv preprint arXiv:1904.11486*, 2019.

[42] Yingliang Zhang, Xi Luo, Wei Yang, and Jingyi Yu. Fragmentation guided human shape reconstruction. *IEEE Access*, 7:45651–45661, 2019.

[43] Ke Colin Zheng, Alex Colburn, Aseem Agarwala, Maneesh Agrawala, David Salesin, Brian Curless, and Michael F Cohen. Parallax photography: creating 3d cinematic effects from stills. In *Proceedings of Graphics Interface 2009*, pages 111–118. Canadian Information Processing Society, 2009.

[44] Hao Zhu, Hao Su, Peng Wang, Xun Cao, and Ruigang Yang. View extrapolation of human body from a single image. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4450–4459, 2018.

[45] C Lawrence Zitnick, Sing Bing Kang, Matthew Uyttendaele, Simon Winder, and Richard Szeliski. High-quality video view interpolation using a layered representation. In *ACM transactions on graphics (TOG)*, volume 23, pages 600–608. ACM, 2004.