Volumetric Capture of Humans with a Single RGBD Camera via Semi-Parametric Learning

Rohit Pandey, Anastasia Tkach, Shuoran Yang, Pavel Pidlypenskyi, Jonathan Taylor, Ricardo Martin-Brualla, Andrea Tagliasacchi, George Papandreou, Philip Davidson, Cem Keskin, Shahram Izadi, Sean Fanello
Google Inc.

Abstract

Volumetric (4D) performance capture is fundamental for AR/VR content generation. Whereas previous work in 4D performance capture has shown impressive results in studio settings, the technology is still far from being accessible to a typical consumer who, at best, might own a single RGBD sensor. Thus, in this work, we propose a method to synthesize free viewpoint renderings using a single RGBD camera. The key insight is to leverage previously seen “calibration” images of a given user to extrapolate what should be rendered in a novel viewpoint from the data available in the sensor. Given these past observations from multiple viewpoints, and the current RGBD image from a fixed view, we propose an end-to-end framework that fuses both these data sources to generate novel renderings of the performer. We demonstrate that the method can produce high fidelity images, and handle extreme changes in subject pose and camera viewpoints. We also show that the system generalizes to performers not seen in the training data. We run exhaustive experiments demonstrating the effectiveness of the proposed semi-parametric model (i.e. calibration images available to the neural network) compared to other state of the art machine learned solutions. Further, we compare the method with more traditional pipelines that employ multi-view capture. We show that our framework is able to achieve compelling results, with substantially less infrastructure than previously required.

1. Introduction

The rise of Virtual and Augmented Reality has increased the demand for high quality 3D content to create compelling user experiences where the real and virtual world seamlessly blend together. Object scanning techniques are already available for mobile devices [30], and they are already integrated within AR experiences [20]. However, neither the industrial nor the research community have yet been able to devise practical solutions to generate high quality volumetric renderings of humans.

At the cost of reduced photo-realism, the industry is currently overcoming the issue by leveraging “cartoon-like” virtual avatars. On the other end of the spectrum, complex capture rigs [7, 39, 3] can be used to generate very high quality volumetric reconstructions. Some of these methods [8, 18] are well established, and lie at the foundation of special effects in many Hollywood productions. Despite their success, these systems rely on multi-view capture rigs that require several (∼4-8) calibrated RGBD sensors.
Conversely, our goal is to make the volumetric capture technology accessible through consumer level hardware. Thus, in this paper, we focus on the problem of synthesizing volumetric renderings of humans. Our goal is to develop a method that leverages recent advances in machine learning to generate 4D videos using as little infrastructure as possible – a single RGBD sensor. We show how a semi-parametric model, where the network is provided with calibration images, can be used to render an image of a novel viewpoint by leveraging the calibration images to extrapolate the partial data the sensor can provide. Combined with a fully parametric model, this produces the desired rendering from an arbitrary camera viewpoint; see Fig. 1.

In summary, our contribution is a new formulation of volumetric capture of humans that employs a single RGBD sensor, and that leverages machine learning for image rendering. Crucially, our pipeline does not require complex infrastructure typically required by 4D video capture setups.

We perform exhaustive comparisons with machine learned, as well as traditional state-of-the-art capture solutions, showing how the proposed system generates compelling results with minimal infrastructure requirements.

2. Related work

Capturing humans in 3D is an active research topic in the computer vision, graphics, and machine learning communities. We categorize related work into three main areas that are representative of the different trends in the literature: image-based rendering, volumetric capture, and machine learning solutions.

Image based rendering. Despite their success, most of methods in this class do not infer a full 3D model, but can nonetheless generate renderings from novel viewpoints. Furthermore, the underlying 3D geometry is typically a proxy, which means they cannot be used in combination with AR/VR where accurate, metric reconstructions can enable additional capabilities. For example, [9, 21] create impressive renderings of humans and objects, but with limited viewpoint variation. Modern extensions [1, 41] produce 360° panoramas, but with a fixed camera position. The method of Zitnick et al. [50] infers an underlying geometric model by predicting proxy depth maps, but with a small 30° coverage, and the rendering heavily degrades when the interpolated view is far from the original. Extensions to these methods [14, 4, 47] have attempted to circumvent these problems by introducing an optical flow stage warping the final renderings among different views, but with limited success.

Volumetric capture. Commercial volumetric reconstruction pipelines employ capture studio setups to reach the highest level of accuracy [7, 39, 12, 11, 36]. For instance, the system used in [7, 39], employs more than 100 IR/RGB cameras, which they use to accurately estimate depth, and then reconstruct 3D geometry [27]. Non-rigid mesh alignment and further processing is then performed to obtain a temporally consistent atlas for texturing. Roughly 28 minutes per frame are required to obtain the final 3D mesh. Currently, this is the state-of-the-art system, and is employed in many AR/VR productions. Other methods [51, 35, 12, 11, 36, 13], further push this technology by using highly customized, high speed RGBD sensors. High framerate cameras [16, 15, 46] can also help make the non-rigid tracking problem more tractable, and compelling volumetric capture can be obtained with just 8 custom RGBD sensors rather than hundreds [28]. However these methods still suffer from both geometric and texture aberrations, as demonstrated by Dou et al. [11] and Du et al. [13].

Machine learning techniques. The problem of generating images of an object from novel viewpoints can also be cast from a machine learning, as opposed to graphics, standpoint. For instance, Dosovitskiy et al. [10] generates re-renderings of chairs from different viewpoints, but the quality of the rendering is low, and the operation is specialized to discrete shape classes. More recent works [25, 38, 49] try to learn the 2D-3D mapping by employing some notion of 3D geometry, or to encode multiview-stereo constraints directly in the network architecture [17]. As we focus on humans, our research is more closely related to works that attempt to synthesize 2D images of humans [48, 2, 43, 32, 31, 34, 5]. These focus on generating people in unseen poses, but usually from a fixed camera viewpoint (typically frontal) and scale (not metrically accurate). The coarse-to-fine GANs of [48] synthesizes images that are still relatively blurry. Ma et al. [31] detects pose in the input, which helps to disentangle appearance from pose, resulting in improved sharpness. Even more complex variants [32, 43] that attempt to disentangle pose from appearance, and foreground from background, still suffer from multiple artifacts, especially in occluded regions. A dense UV map can also be used as a proxy to re-render the target from a novel viewpoint [34], but high-frequency details are still not effectively captured. Of particular relevance is the work by Balakrishnan et al. [2], where through the identification and transformation of body parts results in much sharper images being generated. Nonetheless, note how this work solely focuses on frontal viewpoints.

Our approach. In direct contrast, our goal is to render a subject in unseen poses and arbitrary viewpoints, mimicking the behavior of volumetric capture systems. The task at hand is much more challenging because it requires disentangling pose, texture, background and viewpoint simultaneously. This objective has been partially achieved by
Martin-Brualla et al. [33] by combining the benefits of geometrical pipelines [11] to those of convolutional architectures [42]. However, their work still necessitates a complete mesh being reconstructed from multiple viewpoints. In contrast, our goal is to achieve the same level of photo-realism from a single RGBD input. To tackle this, we resort to a semi-parametric approach [40], where a calibration phase is used to acquire frames of the users appearance from a few different viewpoints. These calibration images are then merged together with the the current view of the user in an end-to-end fashion. We show that the semi-parametric approach is the key to generating high quality, 2D renderings of people in arbitrary poses and camera viewpoints.

3. Proposed Framework

As illustrated in Figure 1, our method receives as input: 1) an RGBD image from a single viewpoint, 2) a novel camera pose with respect to the current view and 3) a collection of a few calibration images observing the user in various poses and viewpoints. As output, it generates a rendered image of the user as observed from the new viewpoint. Our proposed framework is visualized in Figure 2, and includes the four core components outlined below.

Re-rendering & Pose Detector: from the RGBD image \(I\) captured from a camera \(\bar{v}\), we re-render the colored depthmap from the new camera viewpoint \(v\) to generate an image \(I_{\text{cloud}}\), as well as its approximate normal map \(N\). Note we only re-render the foreground of the image, by employing a fast background subtraction method based on depth and RGB as described in [15]. We also estimate the pose \(\kappa\) of the user, i.e. keypoints, in the coordinate frame of \(v\), as well as a scalar confidence \(c\), measuring the divergence between the camera viewpoints:

\[
I_{\text{cloud}}, \kappa, N, c = R(\bar{I}, \bar{v}, v). \tag{1}
\]

Calibration Image Selector: from the collection of calibration RGBD images and poses \(\{I^n_{\text{calib}}, \kappa^n_{\text{calib}}\}\), we select one that best resembles the target pose \(\kappa\) in the viewpoint \(v\):

\[
\hat{I}_{\text{calib}}, \hat{\kappa}_{\text{calib}} = S(\{I^n_{\text{calib}}, \kappa^n_{\text{calib}}\}, \kappa). \tag{2}
\]

Calibration Image Warper: given the selected calibration image \(I_{\text{calib}}\) and the user’s pose \(\hat{\kappa}_{\text{calib}}\), a neural network \(W\) with learnable parameters \(\omega\) warps this image into the desired pose \(\kappa\), while simultaneously producing the silhouette mask \(I^s_{\text{warp}}\) of the subject in the new pose:

\[
I_{\text{warp}}, I^s_{\text{warp}} = W(\hat{I}_{\text{calib}}, \hat{\kappa}_{\text{calib}}, \kappa). \tag{3}
\]

Neural Blender: finally, we blend the information captured by the traditional re-rendering in (1) to the warped calibration image (3) to produce our final image \(I_{\text{out}}\):

\[
I_{\text{out}} = B_\beta(I_{\text{cloud}}, I_{\text{warp}}, I^s_{\text{warp}}, \kappa, c). \tag{4}
\]

Note that while (1) and (2) are not learnable, they extract quantities that express the geometric structure of the problem. Conversely, both warper (3) and (4) are differentiable and trained end-to-end where the loss is the weighted sum between warper \(L_{\text{warp}}\) and blender \(L_{\text{blender}}\) losses. The weights \(\omega_{\text{warp}}\) and \(\omega_{\text{blender}}\) are chosen to ensure similar contributions between the two. We now describe each component in detail, motivating the design choices we took.

3.1. Re-rendering & Pose Detector

We assume that camera intrinsic parameters (optical center \(o\) and focal length \(f\)) are known and thus the function \(\Pi^{-1}(p, z|o, f) : \mathbb{R}^3 \rightarrow \mathbb{R}^3\) maps a 2D pixel \(p = (x, y)\) with associated depth \(z\) to a 3D point in the local camera coordinate frame.

Rendering \(\rightarrow I_{\text{cloud}}\): Via the function \(\Pi^{-1}\), we first convert the depth channel of \(\bar{I}\) into a point cloud of size \(M\) in matrix form as \(P \in \mathbb{R}^{4 \times M}\). We then rotate and translate this point cloud into the novel viewpoint coordinate frame as \(P = TP\), where \(T \in \mathbb{R}^{4 \times 4}\) is the homogeneous transformation representing the relative transformation between \(\bar{v}\) and \(v\). We render \(P\) to a 2D image \(I_{\text{cloud}}\) in OpenGL by splatting each point with a \(3 \times 3\) kernel to reduce re-sampling artifacts. Note that when input and novel camera viewpoints are close, i.e. \(\bar{v} \sim v\), then \(I_{\text{out}} \sim I_{\text{cloud}}\), while when \(\bar{v} \sim \bar{v}\) then \(I_{\text{cloud}}\) would mostly contain unusable information.

Pose detection \(\rightarrow \kappa\). We also infer the pose of the user by computing 2D keypoints \(\bar{k}_{2D} = K(\bar{I})\) using the method of Papandre et al. [37] where \(K\) is a pre-trained feed-forward network. We then lift 2D keypoints to their 3D counterparts \(\kappa\) by employing the depth channel of \(\bar{I}\) and, as before, transform them in the camera coordinate frame \(v\) as \(\kappa\). We extrapolate missing keypoints when possible relying on the rigidity of the limbs, torso, face, otherwise we simply discard the frame. Finally, in order to feed the keypoints \(\kappa\) to the networks in (3) and (4) following the strategy in [2]: we encode each point in an image channel (for a total of 17 channels) as a Gaussian centered around the point with a fixed variance. We tried other representations, such as the one used in [43], but found that the selected one lead to more stable training.

Confidence and normal map \(\rightarrow c, N\). In order for (4) to determine whether a pixel in image \(I_{\text{cloud}}\) contains appropriate information for rendering from viewpoint \(v\) we provide two sources of information: a normal map and a confidence score. The normal map \(N\), processed in a way analogous to \(I_{\text{cloud}}\), can be used to decide whether a pixel in \(\bar{I}\) has been well observed from the input measurement \(\bar{v}\) (e.g. the network should learn to discard measurements taken at low-grazing angles). Conversely, the relationship between \(\bar{v}\) and \(v\), encoded by \(c\), can be used to infer whether
Figure 2. **Proposed framework** – We take in input the current RGBD image, a novel viewpoint and a collection of images acquired in a calibration stage, which depict the users in different poses observed from several viewpoints. The Re-rendering & pose-detector projects the texture using depth information and re-project back to the final viewpoint, together with the target pose. We also compute a confidence score of the current observations with respect to the novel viewpoint. This score is encoded in the normal map \( N \) and the confidence \( c \). The Calibration Image Selector picks the closest image (in terms of viewpoint) from a previously recorded calibration bank. The Calibration Image Warper tries to align the selected calibration image with the current pose, it also produces a silhouette mask. The Neural Blender combines the information from the warped RGB image, aligned calibration image, silhouette image and viewpoint confidence to recover the final, highly detailed RGB image.

A novel viewpoint is back-facing (i.e. \( c < 0 \)) or front-facing it (i.e. \( c > 0 \)). We compute this quantity as the dot product between the cameras view vectors: \( c = \langle 0, 0, 1 \rangle \cdot r_z / \| r_z \| \), where \( \bar{v} \) is always assumed to be the origin and \( r_z \) is the third column of the rotation matrix for the novel camera viewpoint \( v \). An example of input and output of this module can be observed in Figure 2, top row.

### 3.2. Calibration Image Selector

In a pre-processing stage, we collect a set of calibration images \( \{ I_{\text{calib}} \} \) from the user with associated poses \( \{ \bar{\kappa}_{\text{calib}} \} \). For example, one could ask the user to rotate in front of the camera before the system starts; an example of calibration set is visualized in the second row of Figure 2. While it is unreasonable to expect this collection to contain the user in the desired pose, and observed exactly from the viewpoint \( v \), it is assumed the calibration set will contain enough information to extrapolate the appearance of the user from the novel viewpoint \( v \). Therefore, in this stage we select a reasonable image from the calibration set that, when warped by (3) will provide sufficient information to (4) to produce the final output. We compute a score for all the calibration images, and the calibration image with the highest score is selected. A few examples of the selection process are shown in the supplementary material. Our selection score is composed of three terms:

\[
S^c = \omega_{\text{head}} S_{\text{head}}^c + \omega_{\text{torso}} S_{\text{torso}}^c + \omega_{\text{sim}} S_{\text{sim}}^c
\]

From the current 3D keypoints \( \kappa \), we compute a 3D unit vector representing the forward looking direction of the user’s head. The vector is computed by creating a local coordinate system from the keypoints of the eyes and nose. Analogously, we compute 3D unit vectors \( \{ d_{\text{calib}}^{\text{calib}} \} \) from the calibration images keypoints \( \{ \bar{\kappa}_{\text{calib}} \} \). The head score is then simply the dot product \( S_{\text{head}}^c = \bar{d} \cdot \bar{d}_{\text{calib}}^{\text{calib}} \), and a similar process is adopted for \( S_{\text{torso}}^c \), where the coordinate system is created from the left/right shoulder and the left hip keypoints. These two scores are already sufficient to accurately select a calibration image from the desired novel viewpoint, however they do not take into account the configuration of the limbs. Therefore we introduce a third term, \( S_{\text{sim}}^c \), that computes a similarity score between the keypoints \( \bar{\kappa}_{\text{calib}}^{\text{calib}} \) in the calibration images to those in the target pose \( \kappa \). To sim-
3.3. Calibration Warper

The selected calibration image \( \hat{I}_{calib} \) should have a similar viewpoint to \( v \), but the pose \( \hat{\kappa}_{calib} \) could still be different from the desired \( \kappa \), as the calibration set is small. Therefore, we warp \( \hat{I}_{calib} \) to obtain an image \( \hat{I}_{warp} \), as well as its silhouette \( \hat{I}_{warp}^{*} \). The architecture we designed is inspired by Balakrishnan et al. [2], which uses U-NET modules [42]; see Figure 3 for an overview.

The calibration pose \( \hat{\kappa}_{calib} \) tensor (17 channels, one per keypoint) and calibration image \( \hat{I}_{calib} \) go through a U-NET module that produces as output part masks \( \{ I_{part,p}^{*} \} \) plus a background mask \( I_{bg}^{*} \). These masks select which regions of the body should be warped according to a similarity transformation. Similarly to [2], the warping transformations are not learned, but computed via (6) on keypoint groups of at least two 2D points; we have 10 groups of keypoints (see supplementary material for details). The warped texture \( \hat{I}_{warp,p}^{*} \) has 3 RGB channels for each keypoints group \( p \) (30 channels in total). However, in contrast to [2], we do not use the masks just to select pixels to be warped, but also warp the body part masks themselves to the target pose \( \kappa \). We then take the maximum across all the channels and supervise the synthesis of the resulting warped silhouette \( I_{warp,p}^{*} \). We noticed that this is crucial to avoid overfitting, and to teach the network to transfer the texture from the calibration image to the target view and keeping high frequency details. We also differ from [2] in that we do not synthesize the background, as we are only interested in the performer, but we do additionally predict a background mask \( I_{bg}^{*} \).

Finally, the 10 channels encoding the per-part texture \( \hat{I}_{warp,p}^{*} \) and the warped silhouette mask \( I_{part,warp}^{*} \) go through another U-NET module that merges the per-part textures and refines the final foreground mask. Please see additional details in the supplementary material.

The Calibration Warper is training minimizing multiple losses:

\[
\mathcal{L}_{warp} = \omega_{rec} \mathcal{L}_{rec}^{W} + \omega_{fg} \mathcal{L}_{fg}^{W} + \omega_{bg} \mathcal{L}_{bg}^{W} + \omega_{stg} \mathcal{L}_{stg}^{W} + \omega_{GAN} \mathcal{L}_{GAN}^{W} \tag{8}
\]

where all the weights \( \omega_{j} \) are empirically chosen such that all the losses are approximately in the same dynamic range.

**Warp reconstruction loss** \( \mathcal{L}_{rec}^{W} \). Our perceptual reconstruction loss \( \mathcal{L}_{rec}^{W} = \| \text{VGG}(\hat{I}_{warp}^{*}) - \text{VGG}(I_{gt}^{*}) \|_{2} \) measures the difference in VGG feature-space between the predicted image \( \hat{I}_{warp}^{*} \) and the corresponding groundtruth image \( I_{gt}^{*} \). Given the nature of calibration images, \( I_{warp}^{*} \) may lack high frequency details such as facial expressions. Therefore, we compute the loss selecting features from conv2 up to conv5 layers of the VGG network.

**Warp background loss** \( \mathcal{L}_{bg}^{W} \). In order to remove the background component of [2], we have a loss \( \mathcal{L}_{bg}^{W} = \| I_{bg}^{*} - I_{bg,gt}^{*} \|_{1} \) between the predicted mask \( I_{bg}^{*} \) and the groundtruth mask \( I_{bg,gt}^{*} = 1 - I_{fg}^{*} \). We considered other losses (e.g., logistic) but they all produced very similar results.

**Warp foreground loss** \( \mathcal{L}_{fg}^{W} \). Each part mask is warped into target pose \( \kappa \) by the corresponding similarity transformation. We then merge all the channels with a max-pooling operator, and retrieve a foreground mask \( I_{part,warp}^{*} \) over which we impose our loss \( \mathcal{L}_{fg}^{W} = \| I_{part,warp}^{*} - I_{fg,gt}^{*} \|_{1} \). This loss is crucial to push the network towards learning transformation rather than memorizing the solution (i.e., overfitting).

**Warp foreground refinement loss** \( \mathcal{L}_{fgref}^{W} \). The warped part masks \( I_{part,g}^{*} \) may not match the silhouette precisely due
to the assumption of similarity transformation among the body parts, therefore we also refine the mask producing a final binary image $I_{\text{warp}}$. This is trained by minimizing the loss $L_{W}^W_{gt} = \|I_{\text{warp}} - I_{\text{gt}}\|_1$.

Warp GAN loss $L_{GAN}^W$. We finally add a GAN component that helps hallucinating realistic high frequency details as shown in [2]. Following the original paper [19] we found more stable results when used the following GAN component: $L_{GAN}^W = -\text{log}(D(I_{\text{warp}}))$, where the discriminator $D$ consists of 5 conv layers with 256 filters, with max pooling layers to downsample the feature maps. Finally we add 2 fully connected layers with 256 features and a sigmoid activation to produce the discriminator label.

3.4. Neural Blender

The re-rendered image $I_{\text{cloud}}$ can be enhanced by the content in the warped calibration $I_{\text{warp}}$ via a neural blending operation consisting of another U-NET module: please see the supplementary material for more details regarding the architecture. By design, this module should always favor details from $I_{\text{cloud}}$ if the novel camera view $v$ is close to the original $\bar{v}$, while it should leverage the texture in $I_{\text{warp}}$ for back-facing views. To guide the network towards this, we pass as input the normal map $N$, and the confidence $c$, which is passed as an extra channel to each pixel. These additional channels contain all the information needed to disambiguate frontal from back views. The mask $I_{\text{warp}}$ acts as an additional feature to guide the network towards understanding where it should hallucinate image content not visible in the re-rendered image $I_{\text{cloud}}$.

The neural blender is supervised by the following loss:

$$L_{\text{blender}} = w_{\text{rec}} L_{\text{rec}} + w_{\text{GAN}} L_{\text{GAN}}$$ (9)

Blender reconstruction loss $L_{\text{rec}}^B$. The reconstruction loss computes the difference between the final image output $I_{\text{out}}$ and the target view $I_{\text{gt}}$. This loss is defined $L_{\text{rec}}^B = \|\text{VGG}(I_{\text{out}}) - \text{VGG}(I_{\text{gt}})\|_2 + w_{\text{rec}} \|I_{\text{out}} - I_{\text{gt}}\|_1$. A small ($w_{\text{rec}} = 0.01$) photometric ($\ell_1$) loss is needed to ensure faster color convergence.

Blender GAN loss $L_{GAN}^B$. This loss follows the same design of the one described for the calibration warper network.

4. Evaluation

We now evaluate our method and compare with representative state-of-the-art algorithms. We then perform an ablation study on the main components of the system. All the results here are shown on test sequences not used during training; additional exhaustive evaluations can be found in the supplementary material.

4.1. Training Data Collection

The training procedure requires input views from an RGBD sensor and multiple groundtruth target views. Recent multi-view datasets of humans, such as Human 3.6M [24], only provides 4 RGB views and a single low-resolution depth (TOF) sensor, which is insufficient for the task at hand; therefore we collected our own dataset with 20 subjects. Similarly to [33], we used a multi-camera setup with 8 high resolution RGB views coupled with a custom active depth sensor [46]. All the cameras were synchronized at 30Hz by an external trigger. The raw RGB resolution is $4000 \times 3000$, whereas the depth resolution is $1280 \times 1024$. Due to memory limitations during the training, we downsampled also the RGB images to $1280 \times 1024$ pixels.

Each performer was free to perform any arbitrary movement in the capture space (e.g. walking, jogging, dancing, etc.) while simultaneously performing facial movements and expressions. For each subject we recorded 10 sequences of 500 frames. For each participant in the training set, we left 2 sequences out during training. One sequence is used as calibration, where we randomly pick 10 frames at each training iteration as calibration images. The second sequence is used as test to evaluate the performance of a seen actor but unseen actions. Finally, we left 5 subjects out from the training datasets to assess the performances of the algorithm on unseen people.

Silhouette masks generation. As described in Sec. 3.3 and Sec. 3.4, our training procedure relies on groundtruth foreground and background masks ($I_{\text{fg}}$ and $I_{\text{bg,gt}} = 1 - I_{\text{fg}}$). Thus, we use the state-of-the-art body semantic segmenta-
Figure 5. Comparisons with state of the art methods. Notice how the proposed framework favorably compares with traditional volumetric capture rigs that use many (8) cameras from multiple viewpoints. Notice that due to its real-time nature, Motion2Fusion [11] can afford only low resolution (1280 × 1024) RGB images for the texturing phase, whereas FVV [7] accepts as input 4000 × 3000 images.

4.2. Comparison with State of the Art

We now compare the method with representative state of the art approaches: we selected algorithms for comparison representative of the different strategies they use. The very recent method by Balakrishnan et al. [2] was selected as a state of the art machine learning based approach due to its high quality results. We also re-implemented traditional capture rig solutions such as FVV [7] and Motion2Fusion [11]. Finally we compare with LookinGood [33], a hybrid pipeline that combines geometric pipelines with deep networks. Notice, that these systems use all the available views (8 cameras in our dataset) as input, whereas our framework relies on a single RGBD view.

Qualitative Results. We show qualitative results on Figure 5. Notice how our algorithm, using only a single RGBD input, outperforms the method of Balakrishnan et al. [2]: we synthesize sharper results and also handle viewpoint and scale changes correctly. Additionally, the proposed framework generates compelling results, often comparable to multiview methods such as LookinGood [33], Mo-
Table 1. Quantitative evaluations on test sequences. We computed multiple metrics such as Photometric Error ($\ell_1$ loss), PSNR, MS-SSIM and Perceptual Loss. We compared the method with the output of the rendering stage $I_{cloud}$, the output of the calibration selector $I_{calib}$ and the output of the calibration warper $I_{warp}$. We also show how our method outperforms on multiple metrics the state of the art method of Balakrishna et al. [2]. We also favorably compare with full capture rig solutions such as Motion2Fusion [11], FVV [7] and the LookinGood system [33].

<table>
<thead>
<tr>
<th>Method</th>
<th>$\ell_1$ Loss (1 view)</th>
<th>PSNR (8 views)</th>
<th>MS-SSIM (8 views)</th>
<th>VGG Loss (8 views)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Proposed</td>
<td>17.40</td>
<td>28.43</td>
<td>0.92</td>
<td>12.50</td>
</tr>
<tr>
<td>$I_{cloud}$</td>
<td>27.27</td>
<td>22.35</td>
<td>0.84</td>
<td>21.20</td>
</tr>
<tr>
<td>$I_{calib}$</td>
<td>20.02</td>
<td>21.10</td>
<td>0.87</td>
<td>21.41</td>
</tr>
<tr>
<td>$I_{warp}$</td>
<td>18.70</td>
<td>27.32</td>
<td>0.91</td>
<td>13.96</td>
</tr>
<tr>
<td>Balakrishnan et al. [2]</td>
<td>18.01</td>
<td>22.93</td>
<td>0.86</td>
<td>20.16</td>
</tr>
<tr>
<td>LookinGood [33]</td>
<td>38.80</td>
<td>29.93</td>
<td>0.92</td>
<td>10.65</td>
</tr>
<tr>
<td>M2F [11]</td>
<td>33.72</td>
<td>32.21</td>
<td>0.96</td>
<td>5.34</td>
</tr>
<tr>
<td>FVV [7]</td>
<td>7.39</td>
<td>32.60</td>
<td>0.96</td>
<td>6.51</td>
</tr>
</tbody>
</table>

Quantitative Comparisons. To quantitatively assess and compare the method with the state of the art, we computed multiple metrics using the available groundtruth images. The results are shown in Table 1. Our system clearly outperforms the multiple baselines and compares favorably to state of the art volumetric capture systems that use multiple input views.

4.3. Ablation Study

We now quantitatively and qualitatively analyze each each stage of the pipeline. In Figure 6 notice how each stage of the pipeline contributes to achieve the final high quality result. This proves that each component was carefully designed and needed. Notice also how we can also generalize to unseen subjects thanks to the semi-parametric approach we proposed. These excellent results are also confirmed in the quantitative evaluation we reported in Table 1: note how the output of the full system consistently outperforms the one from the re-rendering ($I_{cloud}$), the calibration image selector ($I_{calib}$), and the calibration image warper ($I_{warp}$). We refer the reader to the supplementary material for more detailed examples.

Comparison with fully parametric model. In this experiment we removed the semi-parametric part of our frame-
References


[34] N. Neverova, R. A. Güler, and I. Kokkinos. Dense pose transfer. ECCV, 2018. 2


In this supplementary material, we provide additional information regarding our method’s implementation, more details and more ablation studies on important components of the proposed framework.

1. Framework Details

We detail here the choices of various parameters to aid in reproducing results.

1.1. Calibration Image Selector - Params

In Figure 1 we show some example outputs of the calibration selector module. Note how the module selects the calibration image that most closely matches the viewpoint the person is seen from, based on the target pose. For Eq. 5 we empirically select \( \omega_{\text{head}} = 5 \), \( \omega_{\text{torso}} = 3 \), and \( \omega_{\text{limbs}} = 1 \) so as to weigh the head and torso components of the score highest, then factor in the transformation scores of the limbs.

1.2. Calibration Image Warper

**Keypoint grouping:** We detect 17 keypoints and group them into 10 body parts. The body parts consist of 1) head 2) body 3) left upper arm 4) right upper arm 5) left lower arm 6) right lower arm 7) left upper leg 8) right upper leg 9) left lower leg, and 10) right lower leg. The head keypoints consist of the nose, left/right eyes, and left/right ears. The body keypoints consist of the left/right shoulder, and left/right hip keypoints. Each limb consists of two keypoints; the shoulder and elbow for the upper arms, the elbow and wrist for the lower arms, the hip and knee for the upper legs, and the knee and ankle for the lower legs.

**Network architecture:** Our U-Net architecture consists of 5 encoder blocks followed by 5 decoder blocks. Each encoder block downsamples the input by a factor of 2 and consists of 2 convolutional layers; the first with a kernel size of 3 and stride 1 and the second with a kernel size of 4 and stride 2. Each decoder block consists of a bilinear upsampling layer that upsamples the input by a factor of 2, followed by a convolutional layer with kernel size 3 and stride 1. The encoder blocks use 64 filters for the first block followed by 128 filters for the remaining 4 blocks. The decoder blocks use 128 filters for the first 4 convolutional layers and 32 filters for the final block. Additionally, we add skip connections from the encoder to the decoder in the form of concatenation of feature maps of matching size. Leaky ReLU activations (with \( \alpha = 0.2 \)) are used for all convolutional layers.
The final Neural Blender module takes the output RGB and mask of the calibration warper module, as well as the warped RGB, normals, and viewpoint confidence from the re-rendering module, and learns how to blend them into the final output RGB.

The calibration image warper module adds a final convolutional layer with 4 channels to produce the RGB output $I_{\text{warp}}$, and the mask $I_{\text{warp}}^\cdot$. Tanh activation is used for the RGB and sigmoid for the mask prediction.

### 1.3. Neural Blender

The neural blender module is shown in Figure 2. For simplicity we re-use the same U-Net architecture described in the calibration image warper module section above. However, the last convolutional layer now outputs only 3 channels for the final blended RGB.

### 1.4. Training Details

Our networks are implemented in Tensorflow and trained in parallel on 16 NVIDIA V100 GPUs each with 16 GB of memory. We use the Adam optimizer [1] with a learning rate of $10^{-5}$ for the generator and $10^{-6}$ for the discriminator. We perform light data augmentation during training with random cropping in a size range of 0.85 to 1. times of the input image size. Additionally, we add standard $\ell_2$ loss regularization (with weight $10^{-5}$) to the weights of the network. We found that our data augmentation and regularization, coupled with the variations introduced via using all possible combinations of source and target cameras in our training set, were sufficient to prevent over-fitting and make our network generalize to unseen poses, viewpoints and people.

### 2. Running time of the system

The proposed architecture has an end-to-end runtime of 104.3ms on a Titan V GPU. Note that, this is the unoptimized runtime and does not take advantage of float16 inference or tensor cores on the volta architectures. We leave achieving realtime inference using the proposed architecture for future work.

In Figure 3 we show some examples of the predicted warped part masks and refined masks compared to the ground truth foreground mask. Note that the warped part mask is limited in the accuracy of the silhouette it can produce due to the assumption of 2D similarity transformation between body parts, however, the predicted refined mask is able to overcome these limitations and produce a much cleaner silhouette.

In Figure 4 we show the effect of adding more images to the calibration image pool on the output of the calibration warping module. All calibration images are chosen at random from a held out sequence of the user. Notice how the quality of the output improves as the number of calibration images increases from 6 to 210. This is due to the higher probabil-
Figure 5. Additional results showing various stages of the pipeline.

Figure 6. Additional results showing the viewpoint generalization of the proposed method.

One of the limitations of the proposed approach is that the calibration warper produces blurry results when the viewpoint of the selected calibration image is far from the target viewpoint. We notice this in results when the calibration image pool is small as shown in Figure 4. A larger calibration pool or a predefined calibration sequence where the user turns around the camera can help alleviate this issue. Another limitation of the system is that it struggles to produce reasonable outputs where keypoints are not present, for example hands. We hypothesize that adding additional finger keypoints like fingers and more facial keypoints can help both the calibration selection module, as well as the hallucination modules, to produce better results. Finally, the system shows some temporal flickering as can be seen in the supplementary video. This is especially evident when the selected calibration image changes, and likely can be alleviated via temporal architectures like RNNs or temporal coherency losses.

References