ShowMak3r: Compositional TV Show Reconstruction

Supplementary Material

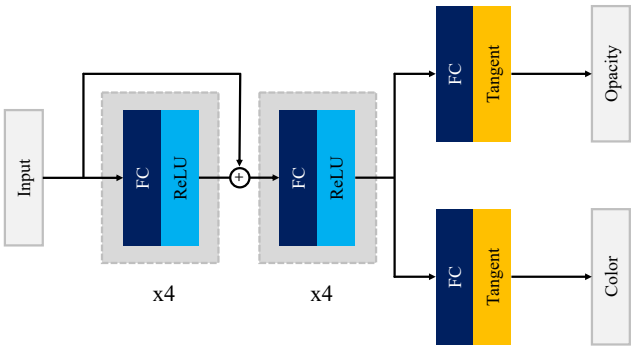

Figure 10. The architecture of our *face fitting network*.

A. Overview

This supplementary material presents additional implementation details and results to support the main paper.

- In Section B, we explain details of the *Face-Fitting network* architecture and provide further implementation specifics.
- In Section C, we visualize the aligned actors from *3DLocator* and provide a detailed explanation of how actors are tracked with *ShotMatcher*.
- Section D shows the results of additional TV show and CMU dataset videos.

B. Implementation Details

Our unoptimized implementation runs offline, which can be boosted with parallel processing. When processing 100 frames of a single person in TBBT scenes in Sitcoms3D dataset [45], our pipeline takes about 30 minutes for stage reconstruction, 10 minutes for SMPL alignment, 1 hour for custom diffusion training, and 3 hours for actor reconstruction. We utilized a single NVIDIA A6000 GPU for training.

3DLocator (Sec. 3.4) optimizes global translation $\mathbf{t}$ and scale s for the first 10k iterations using depth loss (Eq. 5). To ensure SMPL aligns with the image, we freeze $\{\mathbf{t}, s\}$ and optimize pose parameter θ by comparing projected 3D joints with the detected 2D key points for the subsequent 2k iterations. We use Adam optimizer and apply early stopping. We set the learning rate as 0.01 for the first 2.5k iterations and then increase it to 0.1. The learning rate is then reduced with a scale of 10 every 2.5k iterations.

Face fitting network architecture. Fig. 10 shows the architecture of our face-fitting network (Sec. 3.6). We modify the deformation network from D-3DGS [20]. Instead of deforming the position, rotation, and scale of Gaussians, our network adjusts the color and opacity of Gaussians at each time step. In this way, our approach can capture the detailed expression change of the actors.

The face-fitting network takes Gaussian positions and time embeddings as input. To handle multiple actors, we concatenate each actor Gaussian set as $concat\{\mathcal{G}_n^{\text{actor}}\}_{n=1}^{N}$. Concatenated input is then processed through eight fully connected layers with ReLU activation functions. Additionally, the feature vector from the fourth layer is concatenated with the input. Output is a 256-dimensional feature vector, which is then passed to two separate fully connected layers. D-3DGS [20] does not utilize normalization at the end. However, since the opacity and color have a value between 0 and 1, we add a tangent hyperbolic activation function at the end to prevent overflow.

Preprocessing process. First, SAM [23] generates masks for the background stage without actors. These masks are used to exclude actor-containing regions during the feature-matching stage. Subsequently, camera parameters and the SfM point clouds are obtained using GLOMAP [40].

For object removal, Stable Diffusion XL-inpainting model [47] is used to inpaint the specified regions. Depth-Pro [1] is then utilized to predict per-frame depth maps, which are aligned with SfM point clouds. Lastly, we use 4D-Humans [9] to estimate the SMPL parameters of multiple actors.

C. Reconstruction Visualization

In this section, we present the visualization results from *3DLocator* and the association algorithm from *ShotMatcher*.

Alignment visualization. As shown in Fig. 11, by using *3DLocator*, actors are correctly aligned to the stage. Additional results are given in Fig. 12. The green points indicate the centers of the actor Gaussians.

Unlike methods [11, 17, 24] that determine the scale of SMPL by identifying the intersection point between the ground plane and the feet, our approach optimizes scale using aligned depth information. This approach is robust to scenarios where actors are cropped or occluded by objects.

Actor Association. As we see in *Scene 1* of Fig. 4, some actors may not appear in the frame when the shot changes. *ShotMatcher* ensures robust tracking across multiple shots by associating actors with their location. In *Scene 2* of Fig. 4, a different person appears in each shot. If matching is performed using minimum distance alone, the two individuals will be identified as the same person. To address

Figure 11. Visualization of **aligned actors**, **estimated cameras**, and **reconstructed 3D stage**.

this problem, we employ a matching threshold to correctly identify the two individuals. The detailed matching algorithm is provided below.

D. Additional Results

We evaluate additional qualitative results from CMU Panoptic dataset [19]. This dataset captures multiple people interacting with each other within the multi-view camera system. To simulate a TV show within the dataset, we select 8 cameras (out of the original 31) that capture frontal views of the human subject. These 8 views are used for stage reconstruction, while only one among them is used for actor reconstruction. Fig. 13 illustrates novel view synthesis results, achieving a PSNR of 25.21 on average.

Algorithm 1 Actor association algorithm

Input:
- F_i: last frame of the previous shot
- F_{i+1}: first frame of the subsequent shot
- $A = \{A_1, A_2, \ldots, A_n\}$: centers of N actors in F_i
- $B = \{B_1, B_2, \ldots, B_m\}$: centers of M actors in F_{i+1}
- λ: matching threshold

Output: P: matched pairs set
Begin:
$P \leftarrow \emptyset$
$B^{unmatched} \leftarrow B$
for $A_i \in A$ **do**
  $min\_distance \leftarrow \infty$
  $B^{selected} \leftarrow$ **None**
  for $B_j \in B^{unmatched}$ **do**
    $d \leftarrow \text{EuclideanDistance}(A_i, B_j)$
    if $d < min\_distance$ **then**
      $min\_distance \leftarrow d$
      $B^{selected} \leftarrow B_j$
    end if
  end for
  if $min\_distance < \lambda$ **then**
    $P \leftarrow P \cup \{(A_i, B^{selected})\}$
    $B^{unmatched} \leftarrow B^{unmatched} - B^{selected}$
  end if
end for
return P

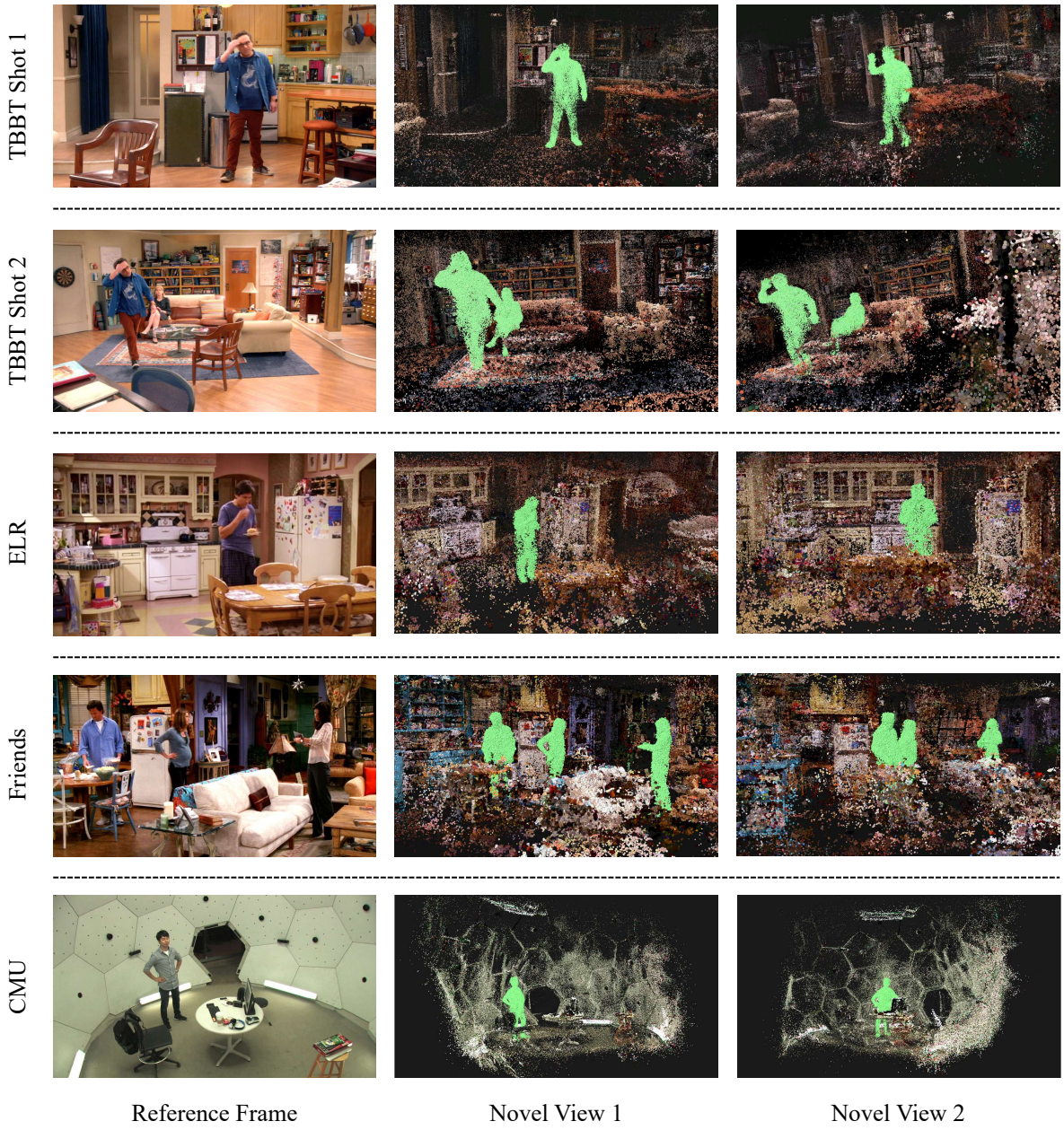

Figure 12. Additional results of the aligned actors.

Figure 13. Results on CMU dataset.