Supplementary Material for POSEFusion: Pose-guided Selective Fusion for Single-view Human Volumetric Capture

Zhe Li[1], Tao Yu[1], Zerong Zheng[1], Kaiwen Guo[2], Yebin Liu[1]
[1]Department of Automation, Tsinghua University, China [2]Google, Switzerland

In this supplementary material, we mainly introduce our implementation details, the system performance, and additional experiments.

A. Implementation Details

A.1. Initialization

Given the tracked SMPL model [2] in the current frame, we firstly allocate a 3D volume which contains the SMPL. For each voxel, we calculate the Euclidean distance between its center with SMPL, and select valid voxels (points) near the SMPL with a threshold of 8cm.

A.2. Pose-guided Keyframe Selection

SMPL Remeshing Due to the uneven distribution of vertices of the original SMPL [2], we remesh the original SMPL into an isotropic triangular template using [1] as shown in Fig. 1. We then transfer the parameters (e.g., blending weights) in [2] to the new template. Using the new template with uniform triangles, the visibility energy can be calculated without the negative impact of dense vertices in the face and hand regions as shown in Fig. 1(a). In our experiment, the new template contains 6839 vertices and 13690 faces. For simplicity, we also call the new template SMPL in the main paper and this supplementary material.

Block Division by Visibility Though the formulated dynamic programming (DP) solution can provide a temporally continuous keyframe trail, this trail in the first iteration denoted as $\mathcal{T}^1$ may cross the diagonal region of the energy matrix E (the red rectangle in Fig. 2(a)), which indicates that in this region the selected keyframe t_i is very close to the current i-th frame, i.e., the i-th and t_i-th frames have roughly the same visible region of the human body. In the second iteration, the DP generates another trail denoted as $\mathcal{T}^2$ as shown in Fig. 2(b). It is obvious that for the red rectangle region $\mathcal{T}^2$ has more contribution for the visibility complementarity, however, $\mathcal{T}^2$ may be discontinuous with $\mathcal{T}^1$ on the boundary of the red rectangle. This will deteriorate the temporal continuity of the reconstructed details in invisible regions in this rectangle region.

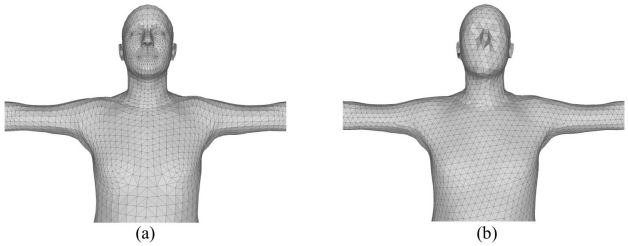

Figure 1. SMPL remeshing using [1]. (a)(b) The wireframes of the original SMPL and the new template, respectively.

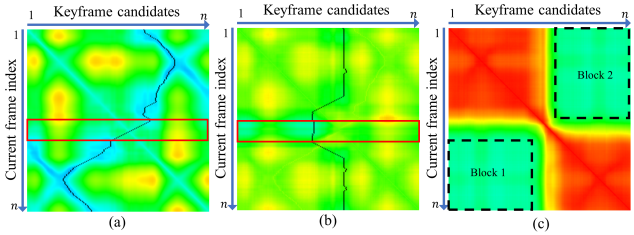

Figure 2. Illustration of the block division by the visibility energy. (a)(b) the keyframe trail on the energy matrix E in the first and second iteration without block division, respectively, and in the red rectangle the selected keyframe in the first iteration is very close to the current frame; (c) the divided blocks by the visibility matrix E_{v} of the first iteration.

Our observation is that the red rectangle usually occurs when the visible region changes significantly, e.g., the performer turns from facing the camera to back facing it, which can be intuitively seen from the visibility matrix E_{v} of the first iteration (the (i, j)-th element of E_{v} is $1 - E_{\text{visibility}}(\mathcal{K}_i, j)$) as shown in Fig. 2(c). We therefore firstly divide the whole sequence as several blocks using E_{v} as shown in Fig. 2(c). If the trail crosses two individual blocks, the red rectangle will be bounded, then we maintain a fixed-size FIFO (first-in-first-out) queue $\mathcal{Q}$ for the rectangle region and push the selected keyframes of previous frames into $\mathcal{Q}$. $\mathcal{Q}$ is considered as the keyframe set of the red rectangle region to guarantee the temporally smooth transition of the reconstructed invisible details. In our experiment, the threshold of block division is 0.3, and the size of $\mathcal{Q}$ is 10.

A.3. Occupancy Inference

Our occupancy inference network is based on PIFu [4] which consists of a Stacked Hourglass network [3] as the image encoder and a MLP with 258, 1024, 512, 256, 128, and 1 neurons in each layer. To fully utilize the depth information, we also encode the depth image and add a PSDF feature to MLP. The PSDF of a 3D point $\mathbf{x}$ is defined as

$$PSDF(\mathbf{x}) = \begin{cases} \mathbf{x}_z - d & , \mathbf{x}_z - d < 0 \\ 0 & , \mathbf{x}_z - d \geq 0 \end{cases}, \quad (1)$$

where $\mathbf{x}_z$ is the z-axis coordinate of $\mathbf{x}$, and d is the depth value sampled at the projected location of $\mathbf{x}$ on the depth image. We render a synthetic dataset (https://web.twindom.com/) to generate depth and color images as training data, and utilize 5000 images to train this network. When training, the batch size is 4, the learning rate is 1×10^{-3} and the number of epochs is 100.

A.4. Collision Handling

Collision Detection Based on the SMPL model, we first calculate the distance between every two vertices, and construct a symmetrical distance matrix $D \in \mathbb{R}^{N \times N}$ (N is the vertex number), and D_{ij} is the distance between the i-th and j-th vertices. For the current frame (the t-th frame) and one keyframe (the k-th frame), we calculate D^t and D^k respectively. And the collided vertices on the current SMPL are detected by the condition:

$$D^t_{ij} < \tau_1, \quad D^k_{ij} > \tau_2, \quad (2)$$

which means that the i-th and j-th vertices collide with each other in the current frame. All the collided SMPL vertices are denoted as $\mathbf{V}_c$. The voxels corresponding to $\mathbf{V}_c$ are also detected. In our experiment, $\tau_1 = 0.02m$ and $\tau_2 = 0.05m$.

Searching No-collision Frame Starting from the current frame, we search for a no-collision frame both forward and backward. We calculate the proportion of SMPL vertices that belong to $\mathbf{V}_c$ but do not meet the collision condition (Eq. 2) during searching each frame. If the proportion is less 30%, the no-collision frame is found. Then we deform the reconstructed model of the no-collision frame to the current frame, and integrate the deformed model into implicit surface fusion. In the fusion procedure, we first generate a 3D mask in the volume by the collision flag, and then perform 3D distance transform to generate a continuous weight volume for integrating the deformed model.

A.5. Parameter Setting

In the keyframe selection, we select $K = 4$ keyframes for each frame. And in the first iteration, $\lambda_{\text{visibility}} = 1.5$, and $\lambda_{\text{visibility}} = 3.0$ in the subsequent iterations. In the adaptive blending weight, we set $\tau = 0.02$ and $\sigma = 100$.

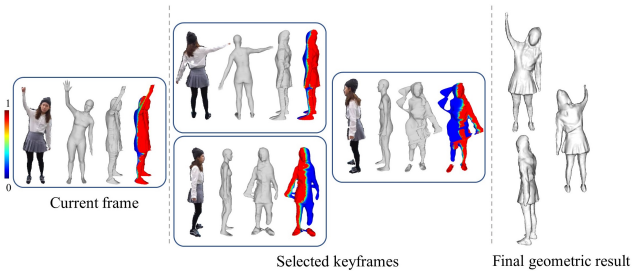

Figure 3. Intermediate results during reconstruction of one frame. In each rectangle, from left to right are the reference color image, warped SMPL, dense reconstruction and color-encoded blending weights, respectively.

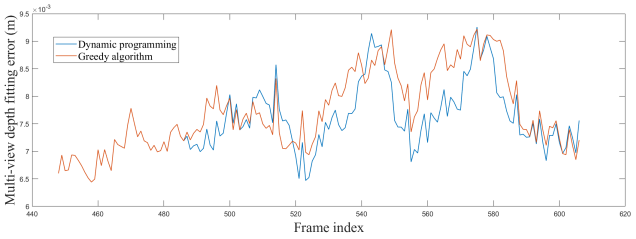

Figure 4. Quantitative comparison against the greedy algorithm on the mean multi-view depth fitting error.

B. Performance

We implement POSEFusion on one PC with one NVIDIA RTX 2080Ti, and the performance mainly depends on the number of keyframes. The time of processing one keyframe is almost 1.87 secs, which consists of 1.53 secs for deforming points and collision detection, 0.02 secs for loading images, 0.3 secs for occupancy inference, and 0.02 secs for blending occupancy values. In our experiment, we perform the pose-guided keyframe selection to select $K = 4$ keyframes, and consider the current frame and the 3 frames around it as keyframes as well for denoising. So the keyframe number is 7, and the time cost for reconstructing one frame is 13 secs. Moreover, we believe the runtime of this system could be further improved by TensorRT and other GPU toolkits.

C. Additional Experiments

Intermediate Results In Fig. 3, given one current frame and its keyframe set, we demonstrate all the intermediate results including the reference color image, warped SMPL, dense reconstruction and color-encoded blending weights within the current frame and each keyframe.

Quantitative Comparison against Greedy Algorithm Fig. 4 demonstrates the mean multi-view fitting errors of the results using the dynamic programming (DP) and greedy algorithm, respectively. And the average error of the whole sequence using DP and greedy algorithm are 0.7507 cm and 0.7732 cm, respectively. It shows that dynamic programming can produce the global optimum by considering the energy sum of the whole sequence, and achieve more phys-

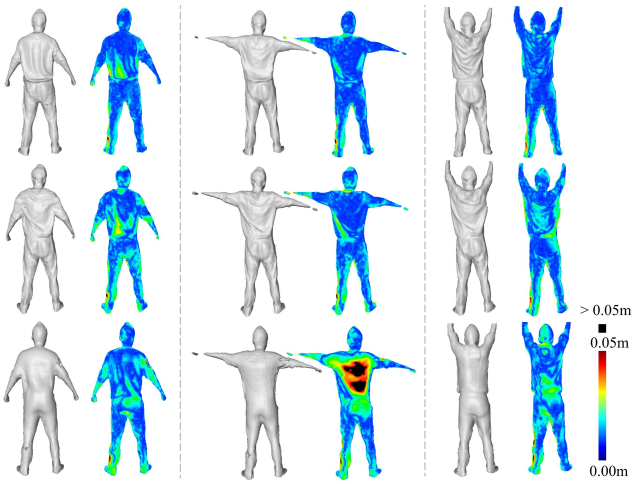

Figure 5. Reconstructed invisible details and visualization of per-vertex errors in several frames. From top to bottom are results with both energies, without the pose energy and without the visibility energy, respectively.

ically accurate reconstruction.

Quantitative Ablation Study of Pose-guided Keyframe Selection Besides the mean vertex errors demonstrated in the main paper (Fig. 13) , we also show the reconstructed results and visualization of per-vertex errors in this ablation study in Fig. 5. It shows that with both visibility and pose guidance, the selected keyframe can cover invisible regions and share the similar poses with the current frame, so that physically plausible invisible details are recovered.

References

[1] Wenzel Jakob, Marco Tarini, Daniele Panozzo, and Olga Sorkine-Hornung. Instant field-aligned meshes. *ACM Transactions on Graphics*, 34(6):189–1, 2015. 1

[2] Matthew Loper, Naureen Mahmood, Javier Romero, Gerard Pons-Moll, and Michael J Black. Smpl: A skinned multi-person linear model. *ACM Transactions on Graphics*, 34(6):1–16, 2015. 1

[3] Alejandro Newell, Kaiyu Yang, and Jia Deng. Stacked hourglass networks for human pose estimation. In *European conference on computer vision*, pages 483–499. Springer, 2016. 2

[4] Shunsuke Saito, Zeng Huang, Ryota Natsume, Shigeo Morishima, Angjoo Kanazawa, and Hao Li. Pifu: Pixel-aligned implicit function for high-resolution clothed human digitization. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2304–2314, 2019. 2