S^3: Neural Shape, Skeleton, and Skinning Fields for 3D Human Modeling (Supplementary Material)

Ze Yang[1,2] Shenlong Wang[1,2] Sivabalan Manivasagam[1,2] Zeng Huang[3]
Wei-Chiu Ma[1,4] Xinchen Yan[1] Ersin Yumer[1] Raquel Urtasun[1,2]
[1]Uber Advanced Technologies Group [2]University of Toronto,
[3]University of Southern California [4]Massachusetts Institute of Techonology
{zeyang, slwang, manivasagam, urtasun}@cs.toronto.edu,
zenghuan@usc.edu, weichium@mit.edu, {xcyan, yumer}@uber.com

In this supplementary material, we provide additional details, analyses and qualitative results of our method. We first describe the details about our network architecture and dataset preprocessing in Section A. Then in Section B we show the additional ablation study on the RenderPeople dataset. Next, we present more details about our animation formulation in Section C, including the Inverse Kinematics (IK) solution that transfers the reconstructed skeleton to the target skeleton, and the Linear Blend Skinning (LBS) formulation used for mesh re-animation. Finally, we showcase more qualitative results in Section D.

A. Additional Details

3D U-Net Architecture: We adopt the 3D U-Net architecture from [3], where the encoder consists of 8 convolution layers. We gather intermediate features and place a max-pooling layer after every two conv layers with $stride = 2$. The decoder consists of 6 convolution layers. We fuse the intermediate features and place an upsampling layer after every two conv layers with $stride = 2$. We pass the resulting feature through one conv layer and obtain the final output. It has the same spatial resolution as the input. Please see Figure 1 for more details.

Dataset Pre-processing: We purchase 793 RenderPeople rigged characters [2] and animate them with 39 different animations downloaded from Mixamo [1]. See Table 1 for a complete list of our selected animation. We randomly select 3 frames per animation for each model, where the sampled frames may be different across characters. To test the generalization ability of our model, we use another 10 *held out* animations from Mixamo [1]. Please see Table 2 for more details. We randomly select one HDRI image and rotate the light direction during the rendering.

There are three factors that will affect the size of a character in the rendered image: (1) the depth of the character, (2) camera intrinsics, and (3) the size of the character. We fix the character depth at 10 meters and randomly perturb the camera's position tangent to its viewing direction to push characters away from the optical axis center. This simulates the severe perspective distortion observed in real images, reducing the sim-to-real gap. We also adjust the focal length to make sure the height of the rendered character is roughly equivalent to 90% of the image. The output image resolution is 512×512 pixels. To generate the corresponding LiDAR points of a 64-beam LiDAR sensor, we ray-cast the posed mesh at a 0.18° azimuth interval and about 0.4° elevation interval from $[2^\circ, -24^\circ]$ using Intel Embree [5] ray tracing kernel. To simulate ray-drop effect and sensor noise, we randomly drop 10% ray-casted points and perturb each ray-casted points with a Gaussian noise along the ray direction ($\sigma = 1$cm).

B. Additional Ablations

Ablation on point location encoding: We compare three different point location features. (1) Compute the depth of the query points in camera coordinate and then normalize them to be zero centered. This can be viewed as the perspective version of PIFu [4]. (2) Compute the point location of the query points in camera coordinate and then normalize them to be zero centered. (3) Our proposed viewpoint encoding (Equation 2 in the main paper). We note that in this ablation we use the image only model where the point location feature is more important. As shown in Table 3, our proposed viewpoint encoding achieves the best performance. This suggests the importance of the viewpoint encoding $\phi_{\text{view}}(\mathbf{p})$ to disambiguate the query points lying on the same camera ray.

Ablation on multi-task learning: Here, we examine whether multi-task learning can improve individual tasks through our unified neural fields. We train a single occupancy head, a pose head, and a multi head network. As shown in Table 4, multi-task learning benefits occupancy

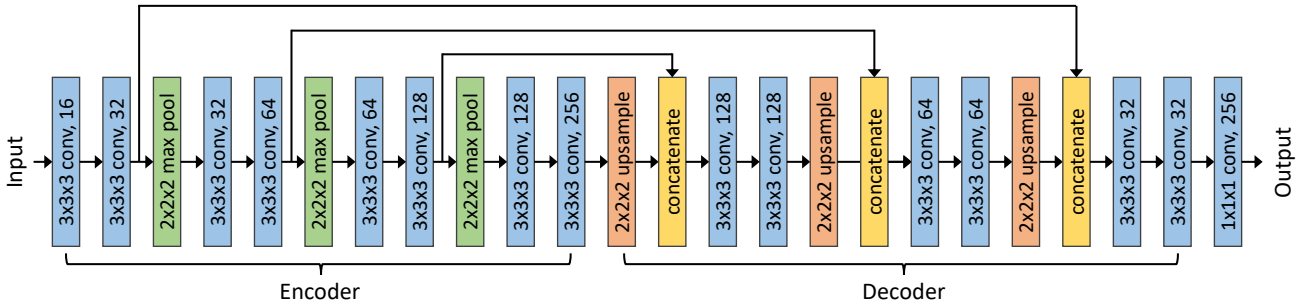


Figure 1: Network architecture of the 3D U-Net.

Agreeing	Bored
Breakdance Ready	Defeat
Defeated	Dwarf Idle
Female Tough Walk	Hands Forward Gesture
Holding Idle	Jogging
Look Over Shoulder	Old Man Idle
Patting	Pointing
Put Back Rifle Behind Shoulder	Run Look Back
Running Right Turn	Running
Searching Files High	Shoulder Rubbing
Standing Clap	Standing Greeting
Standing Torch Idle 02	Standing Turn 90 Right
Standing Turn Left 90	Stop Jumping Jacks
Strut Walking	Talking On Phone
Talking Phone Pacing	Talking
Texting And Walking	Walking Backwards
Walking Left Turn	Walking Turn 180
Walking While Texting	Walking
Walking-2	Yawn
Yelling	

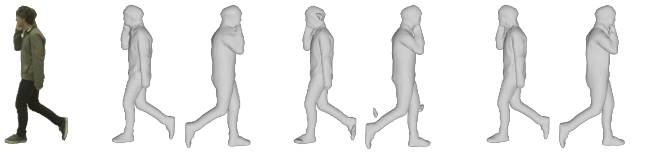
Table 1: List of animations to generate training/test set.

Drunk Idle Variation	Drunk Walk
Jog In Circle	Pacing And Talking On A Phone
Picking Up Object	Right Turn
Slow Run	Taunt
Thankful	Wheelbarrow Walk

Table 2: List of animations to generate unseen test set.

Geometry encoding	Chamfer↓	P2S↓	Normal↑
depth	1.025	1.045	0.882
point location	0.935	0.952	0.886
ours	**0.922**	**0.928**	**0.891**

Table 3: Ablation on different point location encoding. Our viewpoint encoding performs best.



(a) Uniform (b) Biased (c) Uniform + Biased

Figure 2: Qualitative comparison of models trained with different sampling strategy. Using uniform sampling alone tends to have coarse and inaccurate surface. Using biased sampling alone tends to introduce artifacts outside the mesh.

Head	Shape		Pose
	Chamfer↓	P2S↓	MPJPE↓
Occupancy	0.661	0.651	-
Pose	-	-	**1.988**
Occupancy + Pose	**0.647**	**0.632**	2.051

Table 4: Ablation on multi-task learning. Multi-task learning is helpful for shape reconstruction.

predictions, suggesting that skeleton prediction may help the occupancy net through the unified neural field representation.

Ablation on sampling strategy: We show the effects of different query points sampling strategy during training for both human surface reconstruction and human pose estimation in Table 5. A naive sampling strategy is to uniformly sample the query points (or sample grid points) in the space. Although this strategy can supervise the neural field in the full 3D space, the query points will be sparse over the human surface or joints location under memory constraints, preventing the network to learn fine-grained surface details and key-points location. An alternative is to distribute the query points around the object surface (for shape reconstruction) and key-points location (for pose estimation). We observe that combining the uniform sampling strategy and biased sampling can achieve the best performance for both human surface reconstruction and human pose estimation. See Figure 2 for a qualitative comparison. Visually we found that the model trained with uniform sampling alone tends to predict

Strategy	Shape		Pose
	Chamfer↓	P2S↓	MPJPE↓
Uniform	0.777	0.756	3.295
Biased	2.503	4.340	2.448
Uniform + Biased	**0.647**	**0.632**	**2.051**

Table 5: Ablation on sampling strategy. The combination of uniform sampling and biased sampling performs best.

coarse mesh surface and ignore local details. In contrast, the model trained with biased sampling alone tends to predict artifacts outside the decision boundary, which significantly hurts the Chamfer and P2S results. The observation is consistent with [4] where training with hybrid sampling can improve the 3D reconstruction.

C. Human Animation Details

We describe the state of the predicted human skeleton as a set of joints $\mathbf{J} = \{\mathbf{j}_k\}_{k=1}^K$ and inter-connected bones. The movement of human skeleton can be represented as a set of relative rotations for the joints in the skeleton. Given a target pose $\bar{\mathbf{J}} = \{\bar{\mathbf{j}}_k\}_{k=1}^K$, we first calculate the Inverse Kinematics (IK) that transforms our reconstructed skeleton to this target skeleton. We denote the solution as $\mathbf{\Theta}_k \in \mathrm{SO}(3)$, where each relative rotation $\mathbf{\Theta}_k$ describes how the joint $\mathbf{j}_k$ will rotate with respect to its parent joint. Then we animate the human mesh to new pose via the Linear Skinning Model (LBS). Now we describe our IK solution and the LBS model.

Inverse Kinematics Solution: Given the predicted skeleton $\mathbf{J} = \{\mathbf{j}_k\}_{k=1}^K$ and target skeleton $\bar{\mathbf{J}} = \{\bar{\mathbf{j}}_k\}_{k=1}^K$, we aim to find the rotation matrix for each joint that can transform the predicted skeleton to the target skeleton as close as possible. We solve the inverse kinematics analytically by rotating each joint in the predicted skeleton so that its associated bone (the line connecting the joint and its child joint) is parallel to the corresponding bone in the target skeleton. Specifically, we use $\mathbf{b}_k$ to denote the bone direction of the k-th joint $\mathbf{j}_k$ in the predicted skeleton, and $\bar{\mathbf{b}}_k$ to denote the bone direction of the k-th joint $\bar{\mathbf{j}}_k$ in the target skeleton. Then the solution $\{\mathbf{\Theta}_k\}_{k=1}^K$ can be found by solving:

$$\bar{\mathbf{b}}_k = \prod_{p \in A(k)} \mathbf{\Theta}_p \mathbf{b}_k, \quad k = 1, 2, \cdots, K \tag{1}$$

where $A(k)$ is the set of joint ancestors of the k-th joint in order. It is noted that there are infinitely many rotations that map $\mathbf{b}_k$ to $\bar{\mathbf{b}}_k$. We choose the "shortest-arc" rotation between $\mathbf{b}_k$ and $\bar{\mathbf{b}}_k$, *i.e.*, rotate along the cross product of $\mathbf{b}_k$ and $\bar{\mathbf{b}}_k$.

LBS model: Now we describe how we animate the human mesh to new pose using the LBS model. Formally, we denote the predicted human mesh as a set of N vertices $\mathbf{V} = \{\mathbf{v}_i\}_{i=1}^N$, the predicted skeleton as K joints $\mathbf{J} = \{\mathbf{j}_k\}_{k=1}^K$, and the predicted skinning weight as a matrix $\mathbf{W} \in \mathbb{R}^{N \times K}$. We then traverse the kinematic tree and construct the rigid transformation matrix $\mathbf{T}_k(\mathbf{\Theta}_k)$ for each joint using forward kinematics:

$$\mathbf{T}_k(\mathbf{\Theta}_k) = \prod_{p \in A(k)} \begin{bmatrix} \mathbf{\Theta}_p & (\mathbf{I} - \mathbf{\Theta}_p)\mathbf{j}_p \\ \mathbf{0} & 1 \end{bmatrix} \tag{2}$$

where $A(k)$ is the set of joint ancestors of the k-th joint in order, $\mathbf{\Theta}_p$ is the rotation matrix of the p-th joint w.r.t. its parent, and $\mathbf{j}_p$ is the coordinate of the p-th joint in the predicted skeleton. The LBS model assumes the transformation for each vertex $\mathbf{v}_i$ in the human mesh as a linear combination of the rigid transformation matrix $\mathbf{T}_k(\mathbf{\Theta}_k)$ and skinning weight $w_{i,k}$. The coordinate for the i-th vertex after transformation can now be computed as:

$$\bar{\mathbf{v}}_i = \sum_{k=1}^{K} w_{i,k} \mathbf{T}_k(\mathbf{\Theta}_k) \mathbf{v}_i \tag{3}$$

where $w_{i,j}$ is the skinning weight describing the influence of the k-th joint on the i-th vertex in the predicted mesh.

D. Additional Results

We provide more results of our model on real data captured in urban scene with different viewpoints, LiDAR sparsity, lighting and clothes topology. The shape reconstruction results are visualized in Figure 3, in each cell from left to right: camera image overlaid with LiDAR points, foreground image, shape reconstruction. The animation results are visualized in Figure 4, we sample frames from Mixamo [1] animation as the target skeleton pose, in each cell from left to right: foreground image, reconstructed skeleton, reconstructed shape, target skeleton, re-animated shape.

References

[1] Mixamo, https://www.mixamo.com, 2020. 1, 3

[2] Renderpeople, https://renderpeople.com/3d-rigged-people, 2020. 1

[3] Özgün Çiçek, Ahmed Abdulkadir, Soeren S Lienkamp, Thomas Brox, and Olaf Ronneberger. 3d u-net: learning dense volumetric segmentation from sparse annotation. In *International conference on medical image computing and computer-assisted intervention*, 2016. 1

[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 *ICCV*, 2019. 1, 3

[5] Ingo Wald, Sven Woop, Carsten Benthin, Gregory S Johnson, and Manfred Ernst. Embree: a kernel framework for efficient cpu ray tracing. *ACM Transactions on Graphics (TOG)*, 2014. 1

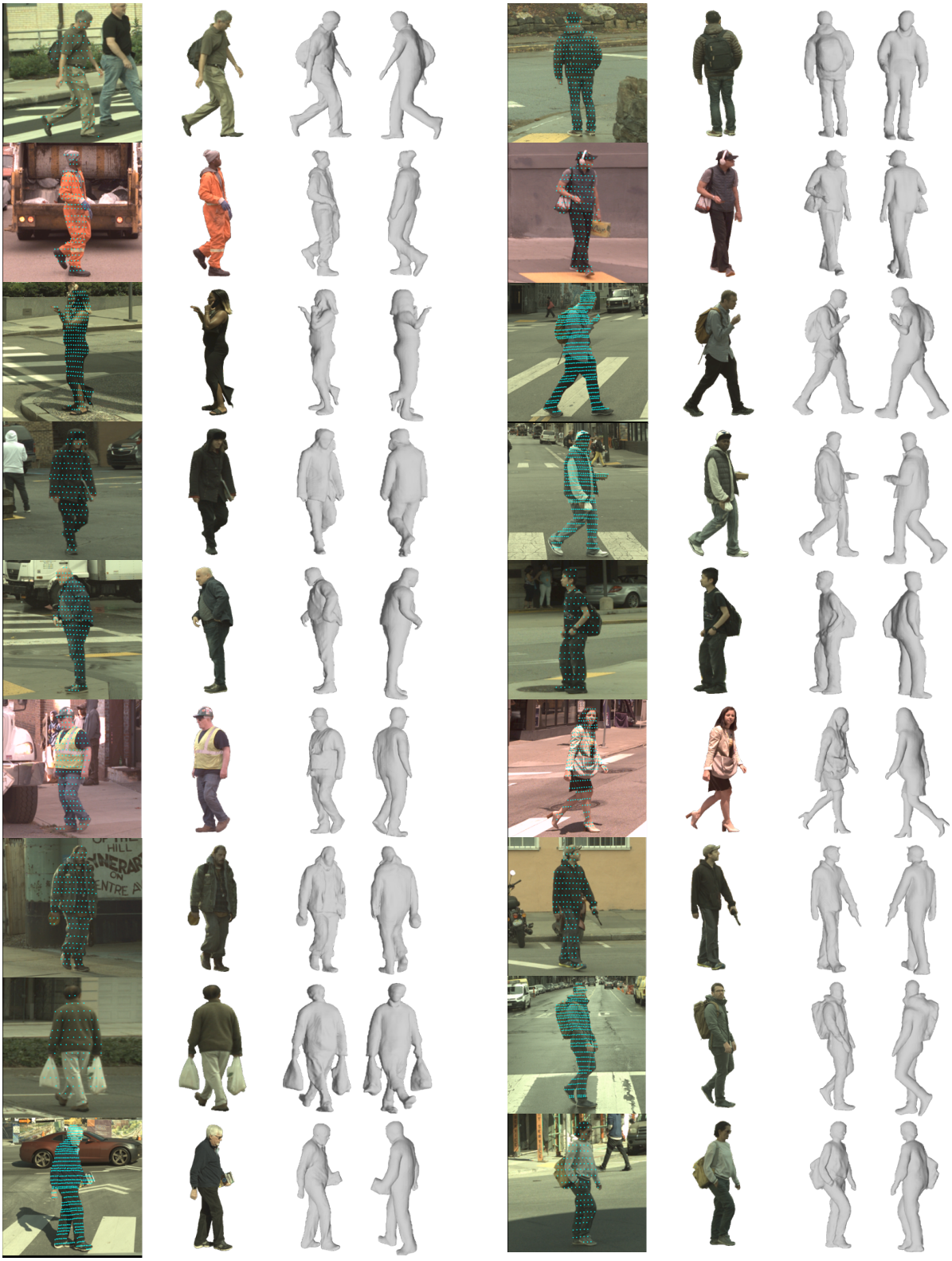


Figure 3: Visualization of reconstruction on more urban scenes. From left to right in each cell: camera image overlaid with LiDAR points, foreground image, shape reconstruction.

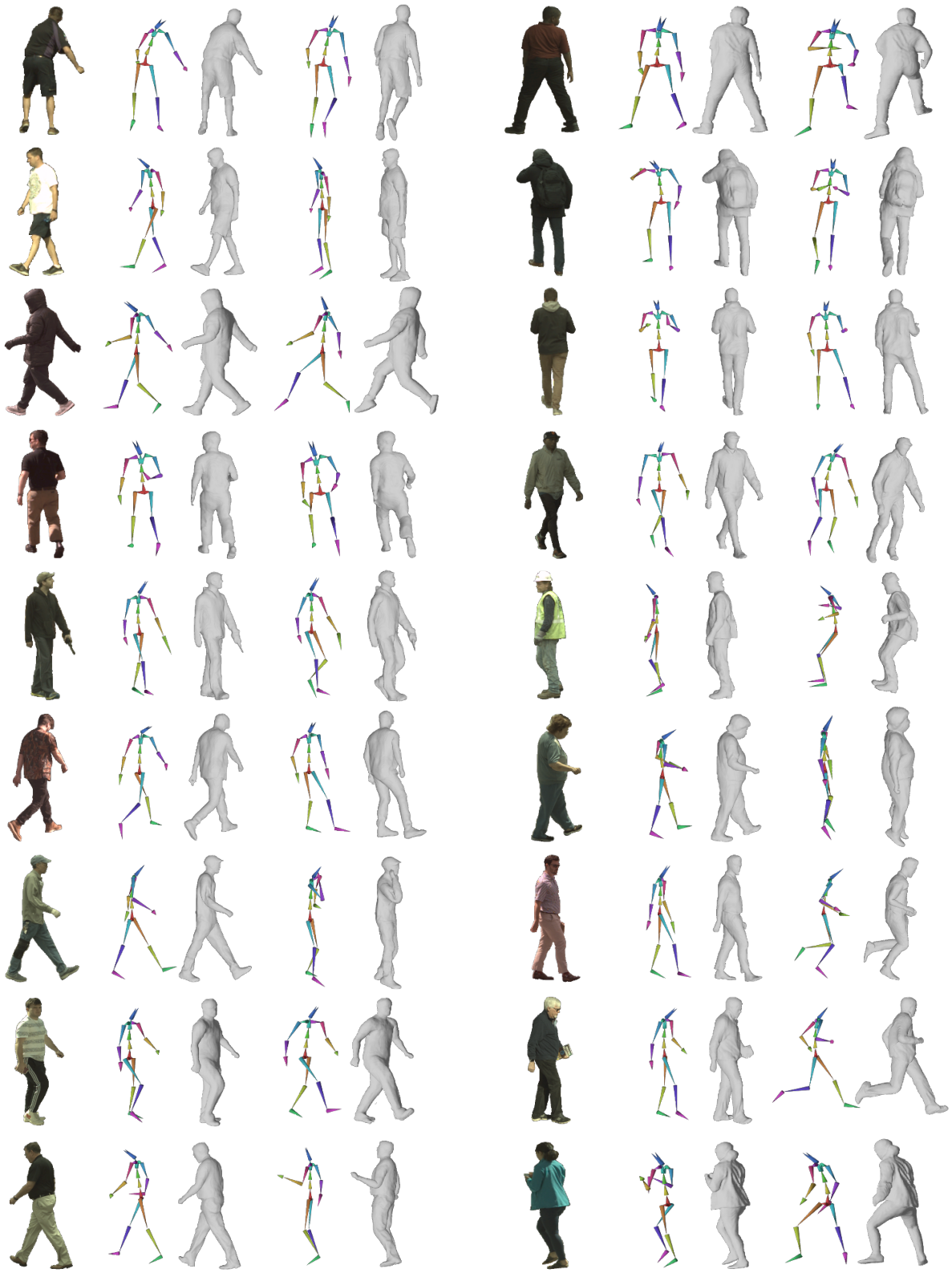


Figure 4: Visualization of animation on more urban scenes. From left to right in each cell: foreground image, reconstructed skeleton, reconstructed shape, target skeleton, re-animated shape.