Reconstructing NBA Players Supplementary Material

Luyang Zhu, Konstantinos Rematas, Brian Curless,
Steven M. Seitz, and Ira Kemelmacher-Shlizerman

University of Washington

1 NB2K Dataset Capture

In this section we provide more details of how we select captures of the NBA2K dataset.

One way to decide which frames to capture is to let the game use its AI where two teams play against each other, however we found that the variety of poses captured in this manner is rather limited. It captures mostly walking and running people, while we target more complex basketball moves. Instead, we have people play the game and proactively capture frames where dunk, dribble, shooting, and other complex basketball moves occur.

2 PoseNet

In this section we provide more details for the PoseNet architecture and setup.

The input is a single, person-centered image with dimensions 256×256. We extract ResNet [17] features from layer 4 and supply them to four separate network branches (2D pose, 3D pose, jump class, jump height). The 2D and 3D pose branches consist of 3 set of Deconvolution-BatchNorm-ReLu blocks. For the jump class, we use a fully connected layer followed by two linear residual blocks [10] to get the final output and we use the same network architecture for the jump height branch. We estimate both the jump class and the jump height because the jump class can serve as a threshold to reject the inaccurate jump height prediction in the global position estimation.

The 2D pose branch outputs a set of 2D 64×64 heatmaps, one for every keypoint, indicating where a particular keypoint is located. Similarly, the 3D pose branch outputs a set of 2D 64×64 location maps [11], where each location map indicates the possible 3D location for every pixel. Each location map has 3 channels that encode the XYZ position of a keypoint with respect to pelvis. To generate the ground truth heatmaps, we first transform the 2D pose from its original image resolution (256×256) to 64×64 resolution, and then generate a 2D Gaussian map centered at each joint location. For ground truth XYZ location maps, we put the 3D joint location at the position where the heatmap has non-zero value. To obtain the final output, we take the location of the maximum value in every keypoint heatmap to get the 2D pose at 64×64 resolution and use it to sample the 3D pose from the XYZ location maps. After that, the 2d pose is transformed to original 256 × 256 resolution. The ground truth jump height is directly extracted from the game, and the jump class is set to 1 if the jump height is greater than 0.1m.

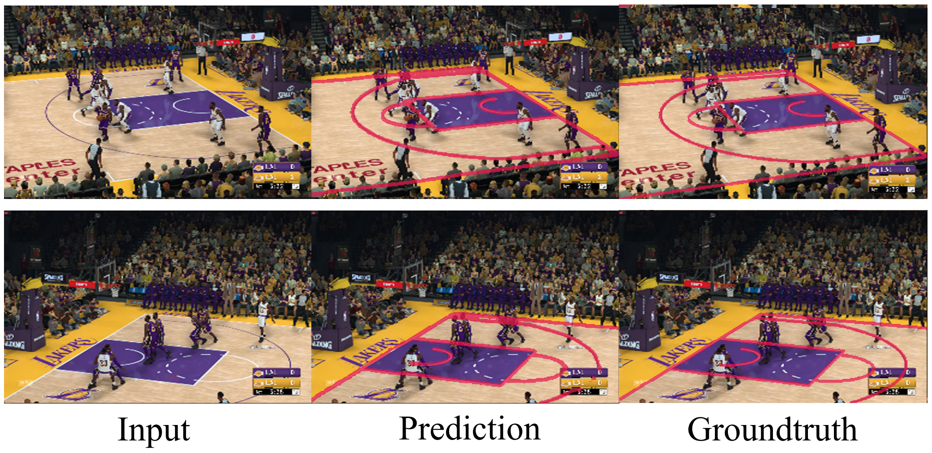

Fig. 1. Court line generation on synthetic data. For every example, from left to right: input image, predicted court lines overlaid on the input image, ground truth court lines overlaid on the input image.

3 Global Position

In this section we describe the process of placing a 3D player in its corresponding position on (or above) the basketball court.

Since a basketball court with players typically has more occlusions (and curved lines) than a soccer field, we found the traditional line detection method used in [13] fails. To get robust line features, we train a pix2pix [6] network to translate basketball images to court line masks. For the training data, we use synthetic data from NBA2K, where the predefined 3D court lines are projected to image space using the extracted camera parameters. To demonstrate the robustness of our line feature extraction method, we provide the results on synthetic data in Figure 1 and real data in Figure 2.

After estimating the camera parameters, we place the player mesh in 3D by considering its 2D pose in the image and the jumping height (Sec 4.1):

$$V_c = \begin{bmatrix} (x_p - p_x)\frac{z_c}{f} \\ (y_p - p_y)\frac{z_c}{f} \\ z_c \end{bmatrix} \tag{1}$$

$$y_w = R_2 \cdot (V_c - T) \tag{2}$$

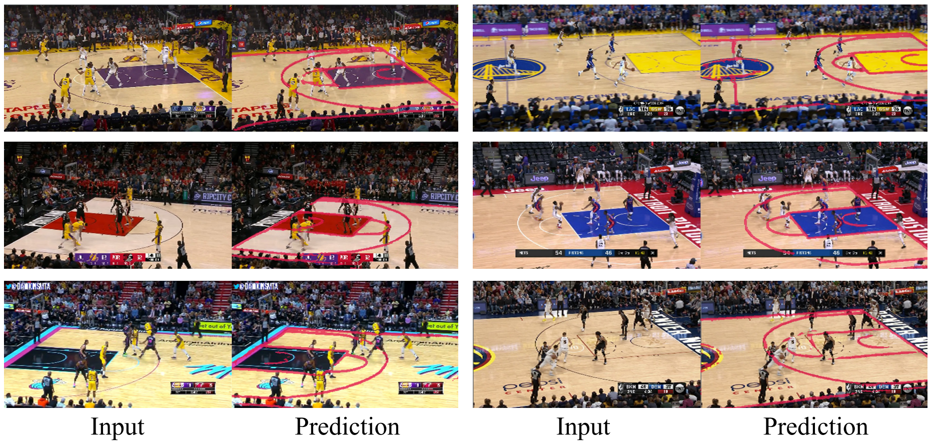

Fig. 2. Court line generation on real data. For every example, left is input image, right is predicted court lines overlaid on the input image.

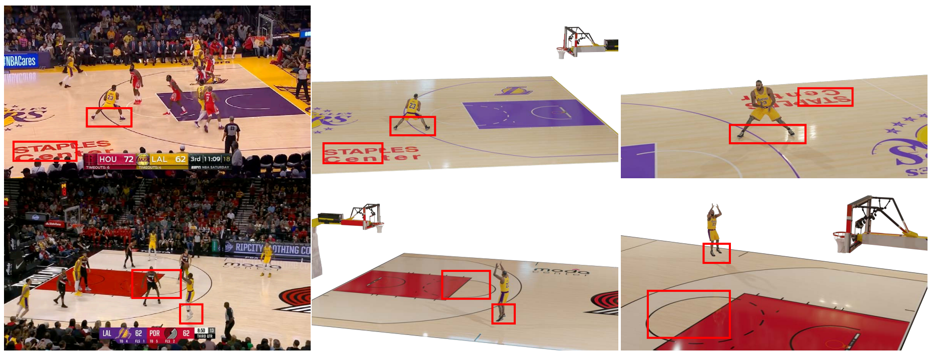

Fig. 3. Global position estimation. Please zoom in to see details. From left to right: input images, two views of the estimated location (middle and right). Note the location of players with respect to court lines (marked with red boxes).

where R_2 is the second column of the extrinsic rotation matrix; T is the extrinsic translation; f is focal length; (p_x, p_y) is the principle point; V_c is the camera coordinates of the lowest joint (foot); y_w is the world coordinate y-component of the lowest joint, which equals the predicted jump height; (x_p, y_p) are the pixel coordinates of the lowest joints. Substituting Eqn. 1 into Eqn. 2, we can solve for z_c (camera coordinate in z-component for lowest joints), from which we can further compute the global position of the player. In Figure 3, we show our results of global position estimation. We can see that our method can accurately place players (both airborne and on the ground) on the court due to accurate jump estimation.

4 Mesh Generation

4.1 SkinningNet

In this section we provide more details for the SkinningNet architecture.

As we noted in the main paper, the pose encoder is comprised of linear residual block [10] followed by a fully connected layer. The linear residual block consists of four FC-BatchNorm-ReLu-Dropout blocks with skip connection from the input to the output. For the mesh part, we denote Spiral Convolution [3] as SC, mesh downsampling and upsampling operator [2] as DS and US. The mesh encoder consists of four SC-ELU [4]-DS blocks, followed by a FC layer. The mesh decoder consists of a FC layer, four US-SC-ELU blocks, and a SC layer for final processing. We follow COMA [2] to perform the mesh sampling operation where vertices are removed by minimizing quadric errors [5] during down-sampling and added using barycentric interpolation during up-sampling. In table 1, we provide detailed settings for the mesh encoders and decoders of different body parts.

Training details. For training IdentityNet and SkinningNet, we use batch size of 16 for 200 epochs and optimize with the Adam solver [7] with weight decay set to 5×10^{-5}. Learning rate for IdentityNet is 0.0002 while learning rate for SkinningNet is 0.001 with a decay of 0.99 after every epoch. The weights of different losses are set to $\omega_Z = 5, \omega_{mesh} = 50$.

4.2 Combining body part meshes

In this section, we provide details of the interpenetration optimization.

As we noted in the main paper, we first detect all the body part vertices in collision with clothing as in [12], and then follow [15, 16] to deform the mesh by moving collision vertices inside the garment while preserving local rigidity of the mesh. This detection-deformation process is repeated until there is no collision or the number of iterations is above a threshold (10 in our experiments). Before each mesh deformation step, collision vertices are first moved in the direction opposite their vertex normals by 10mm. Then we optimize the remaining vertex positions of body parts by minimizing the following loss:

$$\mathcal{L}_{pen} = \omega_{data}\mathcal{L}_{data} + \omega_{lap}\mathcal{L}_{lap} + \omega_{el}\mathcal{L}_{el} \tag{3}$$

	head	arm	shoes	shirt	pant	leg
NV	348	842	937	2098	1439	372
DS Factor	(2,2,1,1)	(2,2,2,1)	(2,2,2,1)	(4,2,2,2)	(2,2,2,2)	(2,2,1,1)
NZ	32 for all body parts					
Filter Size	(16,32,64,64) for encoders, (64,32,16,16,3) for decoders					
Dilation	(2,2,1,1) for encoders, (1,1,2,2,2) for decoders					
Step Size	(2,2,1,1) for encoders, (1,1,2,2,2) for decoders					

Table 1. Network architecture for mesh encoders and decoders of different body parts. NV represents vertices numbers, DS factor represents downsampling factors. NZ represents the hidden size of latent vector. Filter Size represents the output channel of SC. Dilation represents dilation ratio for SC. Step size represents hops for SC.

$\mathcal{L}_{data} = \|V - V^*\|_2$ forces optimized vertices V to stay close to the SkinningNet inferred vertices $V^* = V(Z_{pred})$, $\mathcal{L}_{lap} = \|\Delta_V - \Delta_{V^*}\|_F$ is the Frobenius norm of Laplacian difference between the optimized and inferred meshes, and $\mathcal{L}_{el} = \|\frac{E}{E^*} - 1\|$ encourages the optimized edge length E to be same as the inferred edge length E^*. Each of these losses is taken as a sum over all vertices or edges. We set $\omega_{data} = 1, \omega_{lap} = 0.1, \omega_{el} = 0.1$ respectively. We use an L-BFGS solver [9], running for 20 iterations. Note that detected collision vertices, after being moved inward, are fixed during the optimization process. This hard constraint ensures the optimization will not move these vertices outside garments in future iterations. Figure 4 shows results before and after interpenetration optimization for two examples.

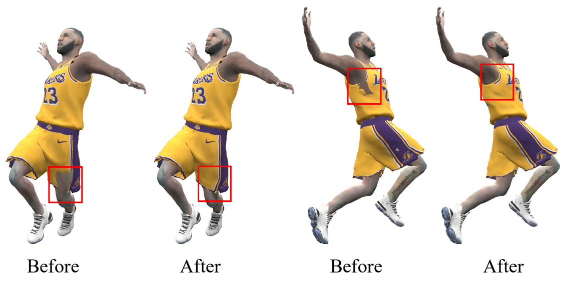

Fig. 4. Before and after interpenetration optimization. Note the garment in the red square. Ground truth textures are used to better visualize the intersection.

5 Further Qualitative Evaluation

In this section, we provide additional qualitative comparisons that further demonstrate the effectiveness of our system.

Fig 5 shows qualitative comparison with tex2shape [1]. Note that tex2shape is only trained with their A-pose data and directly tested on NBA images. We can see our method can generate better shirt wrinkles and body details under different poses.

In the main paper, we only provide qualitative comparisons for synthetic data with state-of-the-art methods. In Figure 6, we compare our method against the best-performing SMPL-based methods [12, 8] on real images. In Figure 7, we additionally compare with PIFu [14], the state-of-the-art method for clothed subjects, on real images. Our system generates more stable poses and more realistic, fine details for real images.

In Figure 8, we provide additional qualitative results of our method for real images. Our method can reconstruct 3D shape of different people under various poses on real images.

In Figure 9, we provide examples where our approach fails to reconstruct a correct 3D shape from single view images.

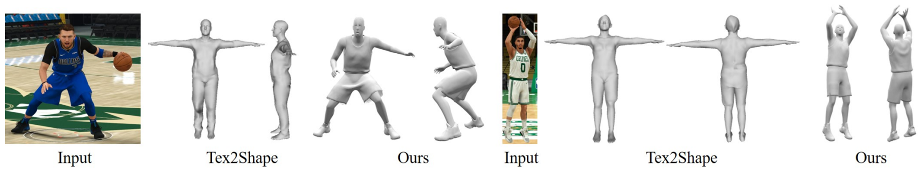

Fig. 5. Comparison with Tex2shape[1]. Note that tex2shape only predicts rough body shape compared to our reconstructions. We follow their advice to select images where person is large and fully visible.

References

1. Alldieck, T., Pons-Moll, G., Theobalt, C., Magnor, M.: Tex2shape: Detailed full human body geometry from a single image. In: IEEE International Conference on Computer Vision (ICCV). IEEE (2019)
2. Anurag Ranjan, Timo Bolkart, S.S., Black, M.J.: Generating 3D faces using convolutional mesh autoencoders. In: European Conference on Computer Vision (ECCV). pp. 725–741. Springer International Publishing (2018), http://coma.is.tue.mpg.de/
3. Bouritsas, G., Bokhnyak, S., Ploumpis, S., Bronstein, M., Zafeiriou, S.: Neural 3d morphable models: Spiral convolutional networks for 3d shape representation learning and generation. In: The IEEE International Conference on Computer Vision (ICCV) (2019)
4. Clevert, D.A., Unterthiner, T., Hochreiter, S.: Fast and accurate deep network learning by exponential linear units (elus). arXiv preprint arXiv:1511.07289 (2015)
5. Garland, M., Heckbert, P.S.: Surface simplification using quadric error metrics. In: Proceedings of the 24th annual conference on Computer graphics and interactive techniques. pp. 209–216. ACM Press/Addison-Wesley Publishing Co. (1997)

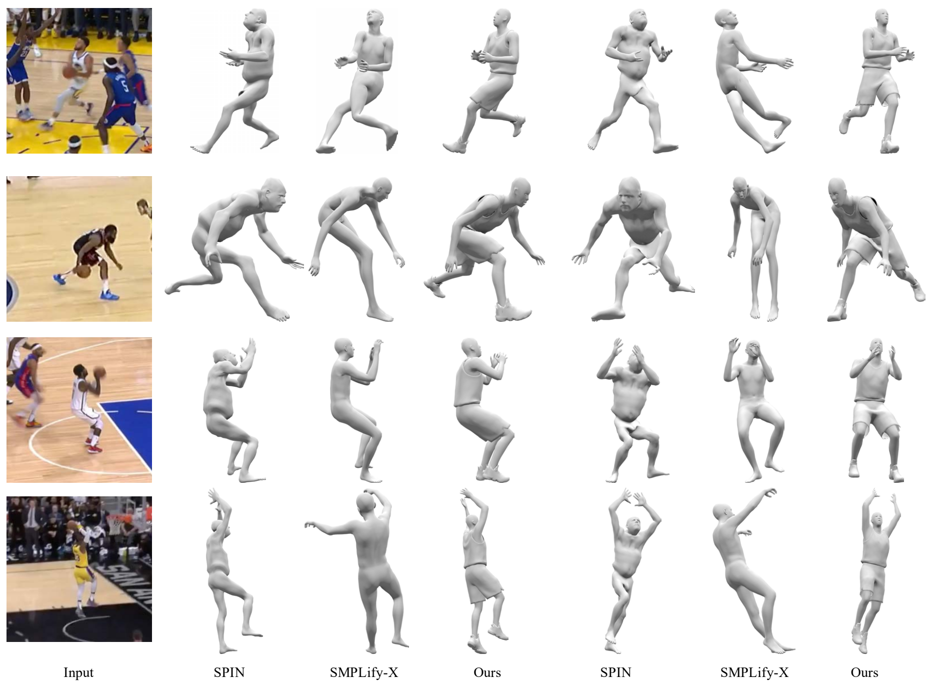

Fig. 6. Comparison with SMPL-based methods on real images. Column 1 is input, columns 2-4 are reconstructions in the image view, columns 5-7 are visualizations from a novel viewpoint. Note the significant difference in body pose between ours and SMPL-based methods; our results are qualitatively much more similar to what is seen in the input images. In addition, SMPL-based methods do not handle clothing.

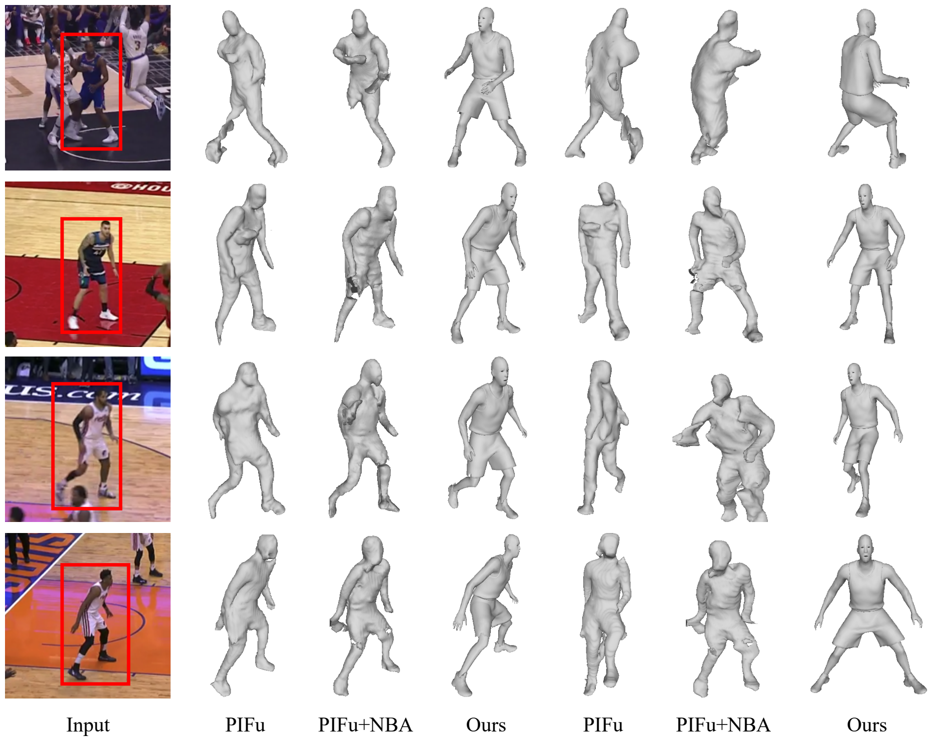

Fig. 7. Comparison with PIFu [14] on real images. Column 1 is input (red box shows the target player), columns 2-4 are reconstructions in the image view, columns 5-7 are reconstructions in a novel view. PIFu fails to reconstruct high quality human shapes from real images, even when the players are in nearly standing poses.

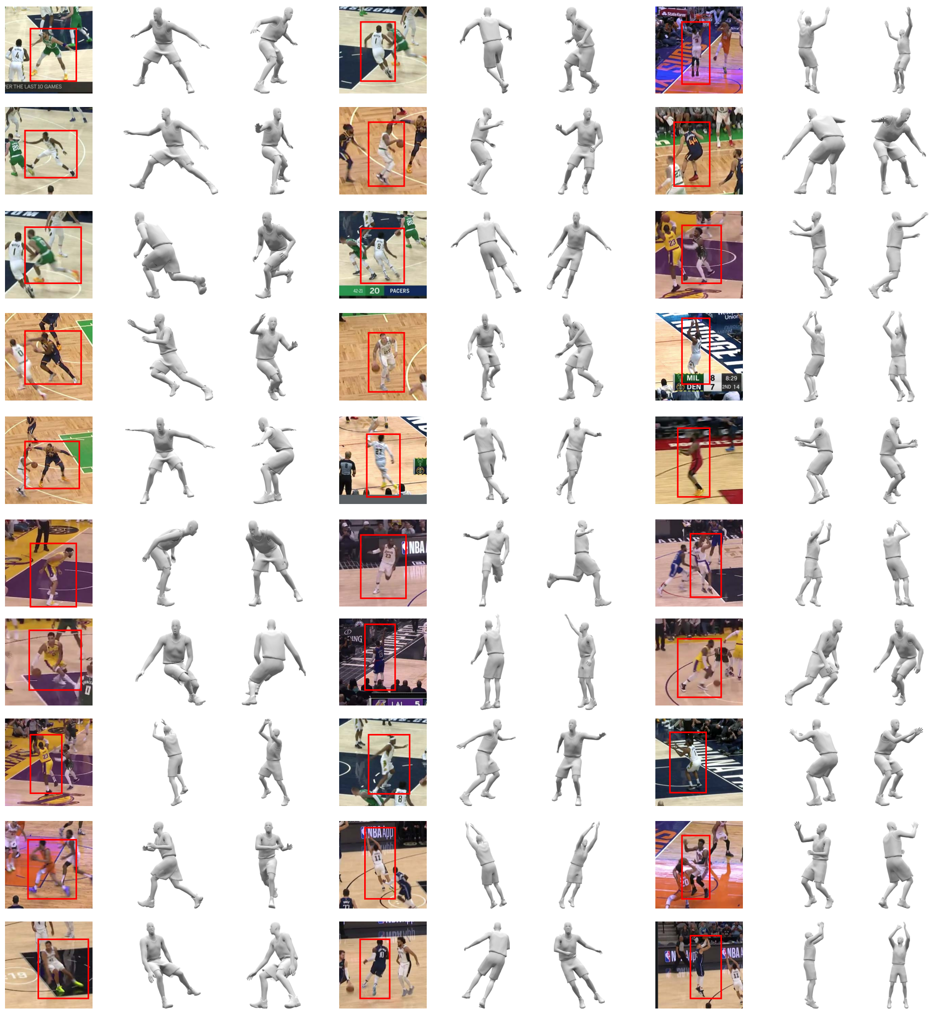

Fig. 8. Qualitative Results on real images. Please zoom in to see details. For every example, left is input (red box shows the target player), middle is reconstruction in the image view, right is reconstruction in a novel view. Our method generalizes well on real images under a variety of poses.

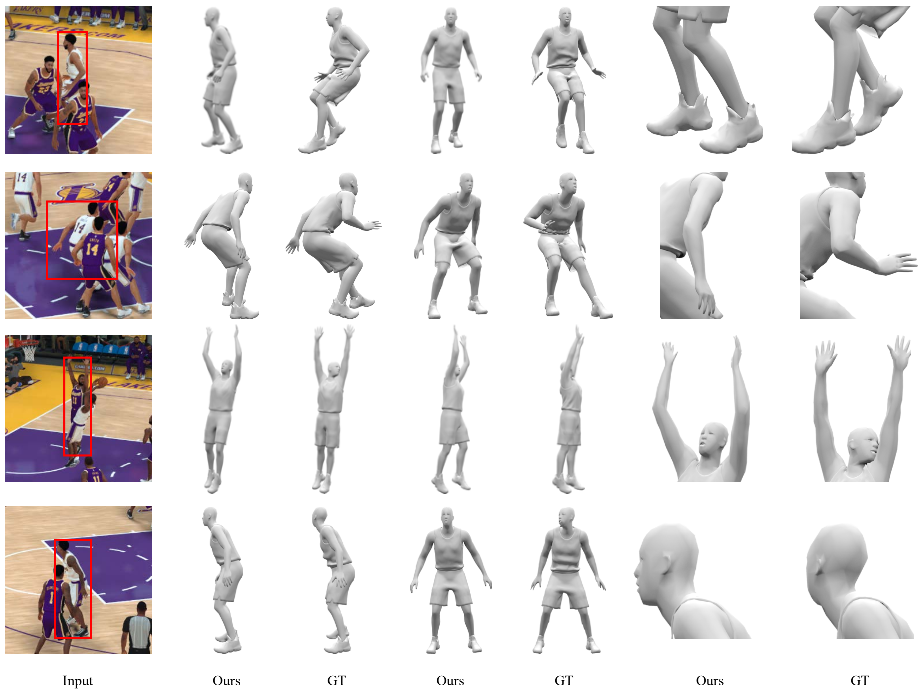

Fig. 9. Typical failure cases of our approach. Column 1 is input (red box shows the target player), columns 2-3 are reconstructions in the image view, columns 4-5 are reconstructions in a novel view, columns 6-7 are zoomed-in versions of main errors. Failures include erroneous pose due to heavy occlusion in multi-person scenes (first and second example), incorrect orientation of head and hands (third and fourth example).

6. Isola, P., Zhu, J.Y., Zhou, T., Efros, A.A.: Image-to-image translation with conditional adversarial networks. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 1125–1134 (2017)
7. Kingma, D.P., Ba, J.: Adam: A method for stochastic optimization. arXiv preprint arXiv:1412.6980 (2014)
8. Kolotouros, N., Pavlakos, G., Black, M.J., Daniilidis, K.: Learning to reconstruct 3d human pose and shape via model-fitting in the loop. In: Proceedings of the IEEE International Conference on Computer Vision (2019)
9. Liu, D.C., Nocedal, J.: On the limited memory bfgs method for large scale optimization. Mathematical programming **45**(1-3), 503–528 (1989)
10. Martinez, J., Hossain, R., Romero, J., Little, J.J.: A simple yet effective baseline for 3d human pose estimation. In: Proceedings of the IEEE International Conference on Computer Vision. pp. 2640–2649 (2017)
11. Mehta, D., Sridhar, S., Sotnychenko, O., Rhodin, H., Shafiei, M., Seidel, H.P., Xu, W., Casas, D., Theobalt, C.: Vnect: Real-time 3d human pose estimation with a single rgb camera. ACM Transactions on Graphics (TOG) **36**(4), 44 (2017)
12. Pavlakos, G., Choutas, V., Ghorbani, N., Bolkart, T., Osman, A.A.A., Tzionas, D., Black, M.J.: Expressive body capture: 3d hands, face, and body from a single image. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (2019)
13. Rematas, K., Kemelmacher-Shlizerman, I., Curless, B., Seitz, S.: Soccer on your tabletop. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 4738–4747 (2018)
14. Saito, S., , Huang, Z., Natsume, R., Morishima, S., Kanazawa, A., Li, H.: Pifu: Pixel-aligned implicit function for high-resolution clothed human digitization. arXiv preprint arXiv:1905.05172 (2019)
15. Sorkine, O., Alexa, M.: As-rigid-as-possible surface modeling. In: Symposium on Geometry processing. vol. 4, pp. 109–116 (2007)
16. Sorkine, O., Cohen-Or, D., Lipman, Y., Alexa, M., Rössl, C., Seidel, H.P.: Laplacian surface editing. In: Proceedings of the 2004 Eurographics/ACM SIGGRAPH symposium on Geometry processing. pp. 175–184 (2004)
17. Xiao, B., Wu, H., Wei, Y.: Simple baselines for human pose estimation and tracking. In: European Conference on Computer Vision (ECCV) (2018)