Towards Accurate 3D Human Body Reconstruction from Silhouettes

Supplementary Material

Brandon M. Smith, Visesh Chari , Amit Agrawal, James Rehg, Ram Sever
Amazon.com Inc.
{smithugh,viseshc,aaagrawa,jamerehg,severram}@amazon.com

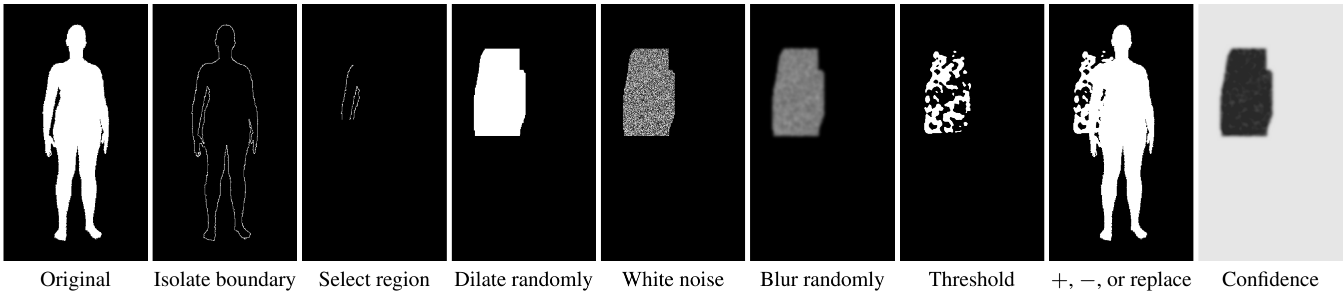
1. Synthetic Data Generation Details

Large segmentation errors are created by selecting regions of the silhouette at random and filling them with random splotches. The process is illustrated in Figure 1. This mimics the behavior of the segmentation network in uncertain image regions, and is generated as follows: (1) select a random silhouette boundary pixel, (2) construct a randomly sized region of interest (ROI) around the selected pixel (from 10 to 200 pixels on a side), (3) select all boundary pixels withing the ROI, (4) dilate these pixels using either a square or round kernel with random diameter between 10 and 30 pixels, (5) fill the dilated region with white noise, (4) smooth the dilated region with a random-bandwidth Gaussian filter, and (5) threshold, which creates splotches of various sizes. These splotchy regions closely mimic the kinds of segmentation errors observed in low confidence regions in real images, where foreground and background labels are predicted randomly. Once created, noise patches are added to, subtracted from, or replace the corresponding region of the silhouette image at random. Empirically we modeled low confidence with a score of 0.2, and high confidence with a score of 0.9. We use these synthetic images for pre-training our network, and then subsequently fine-tune using real segmentation results on photorealistic CAESAR [1] scans rendered in front of random background images from the LSUN dataset [2].

2. Repeatability Examples

We investigated the repeatability of our system on more realistic CAESAR[1] renderings. We rendered 1k color scan in front of ten different backgrounds, each from a different virtual camera viewpoint. This produced a dataset of 10k instances. To better handle the subtleties of real segmentation noise and confidence, we fine-tuned our full network on 9k instances. On the remaining 1k instances (100 unique scans outside the training set) we measured the standard deviation of each vertex across the 10 different renderings of each scan. The average vertex standard deviation was 3.09mm. For reference, 3mm corresponds to approximately 1 pixel for an average height person occupying 600 vertical pixels of a VGA (640 × 480) image.

Three example repeated sessions are shown in Figure 2. The estimated 3D model are consistent (low standard deviation) across a wide range of inputs for the same person. The plot in Figure 2 shows that our method produces results with low *intra*-subject measurement variance compared to the amount of variance between different subjects.



Original | Isolate boundary | Select region | Dilate randomly | White noise | Blur randomly | Threshold | $+$, $-$, or replace | Confidence

Figure 1. An example that illustrates synthetic segmentation noise and confidence mask generation.

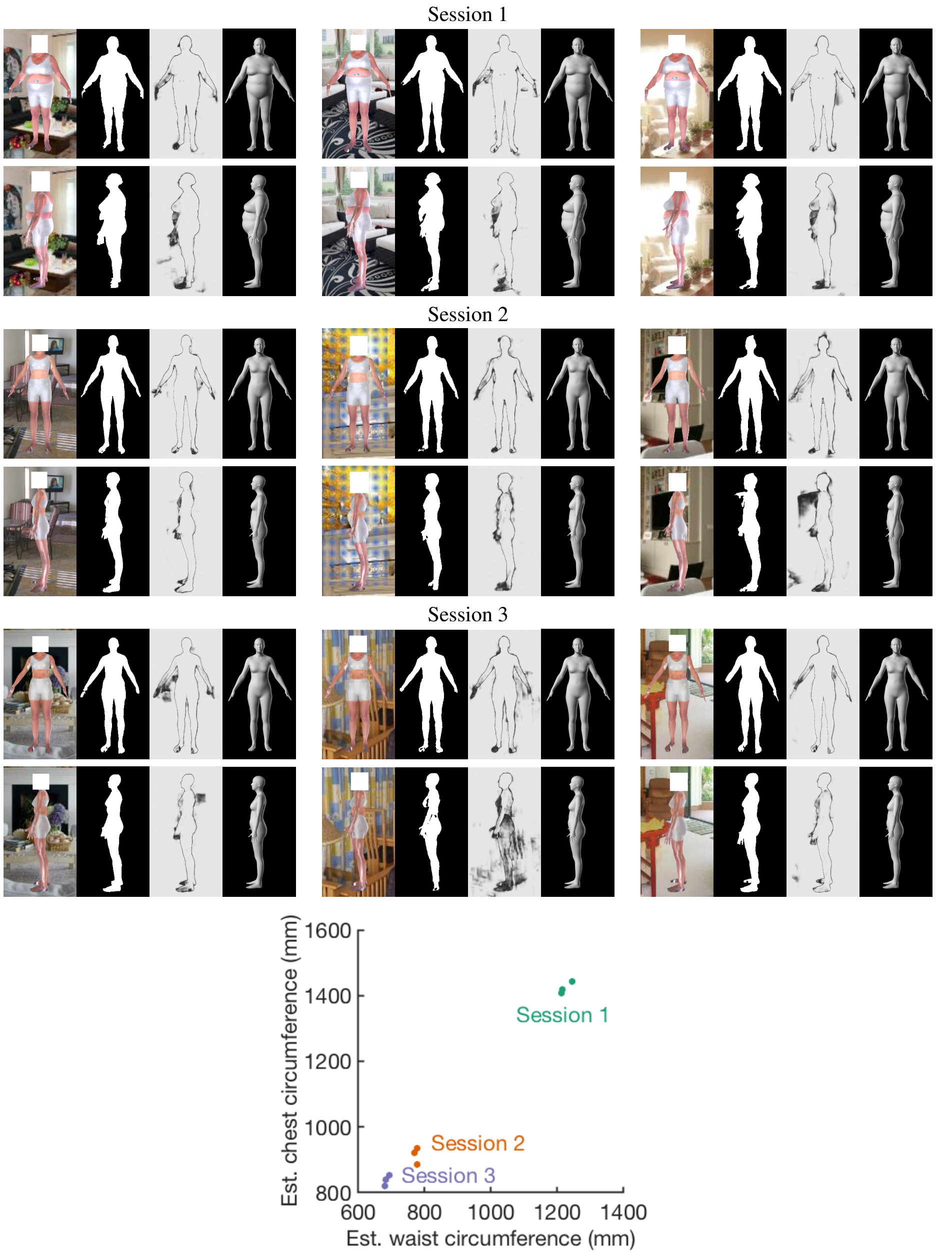


Figure 2. Repeatability examples. **Top:** Three example repeated sessions (input RGB images, binary segmentation masks, segmentation confidence, and estimated 3D body model). Despite significant camera variation and segmentation noise, the estimated 3D models are consistent within each session. **Bottom:** For repeated captures of the same person with different camera positions and backgrounds, our method produces results with low intra-subject measurement variance compared to the amount of variance between sessions/subjects, *i.e.*, clusters are tight and well separated.

3. Qualitative Results on CAESAR Renderings

Here we present additional qualitative results from the experiment described in Section 4.4 of the main paper. Each input example is an RGB scan from the CAESAR dataset [1] rendered in front of a random background sampled from the LSUN image dataset [2]. Figures 3, 4, 5, and 6 highlight good results, and Figure 7 highlights several problem cases.

4. Architecture Details

Our proposed system was implemented in Keras with a TensorFlow backend. In Figure 8 we provide our network architecture. Please see Figure 1 in the main paper for a high-level overview.

References

[1] K. M. Robinette, S. Blackwell, H. Daanen, M. Boehmer, S. Fleming, T. Brill, D. Hoeferlin, and D. Burnsides. Civilian American and European Surface Anthropometry Resource (CAESAR) final report. *Tech. Rep. AFRL-HEWP-TR-2002-0169, US Air Force Research Laboratory*, 2002. 1, 2

[2] F. Yu, Y. Zhang, S. Song, A. Seff, and J. Xiao. Lsun: Construction of a large-scale image dataset using deep learning with humans in the loop. *arXiv preprint arXiv:1506.03365*, 2015. 1, 2

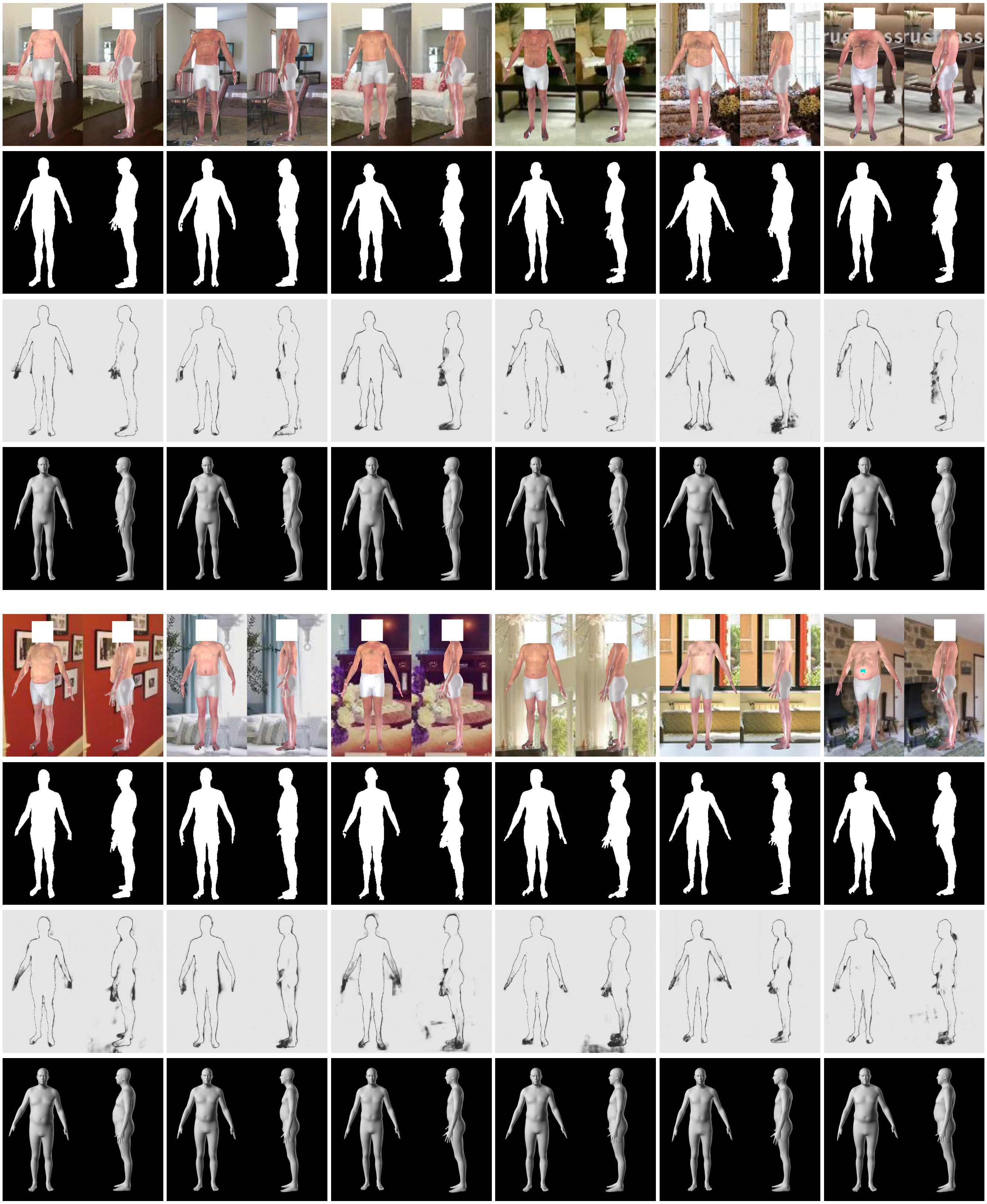


Figure 3. Qualitative results on male CAESAR images, part 1.

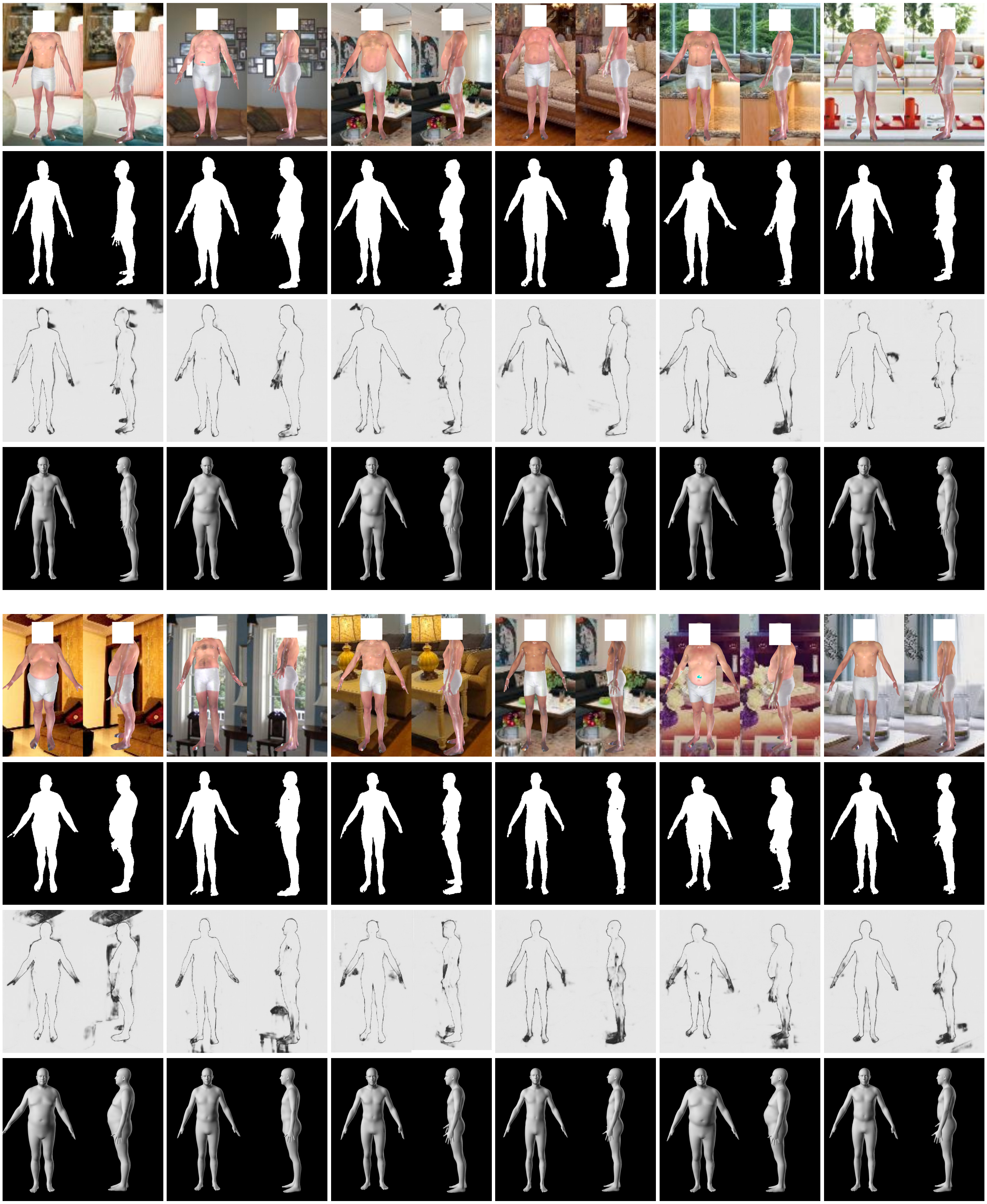


Figure 4. Qualitative results on male CAESAR images, part 2.

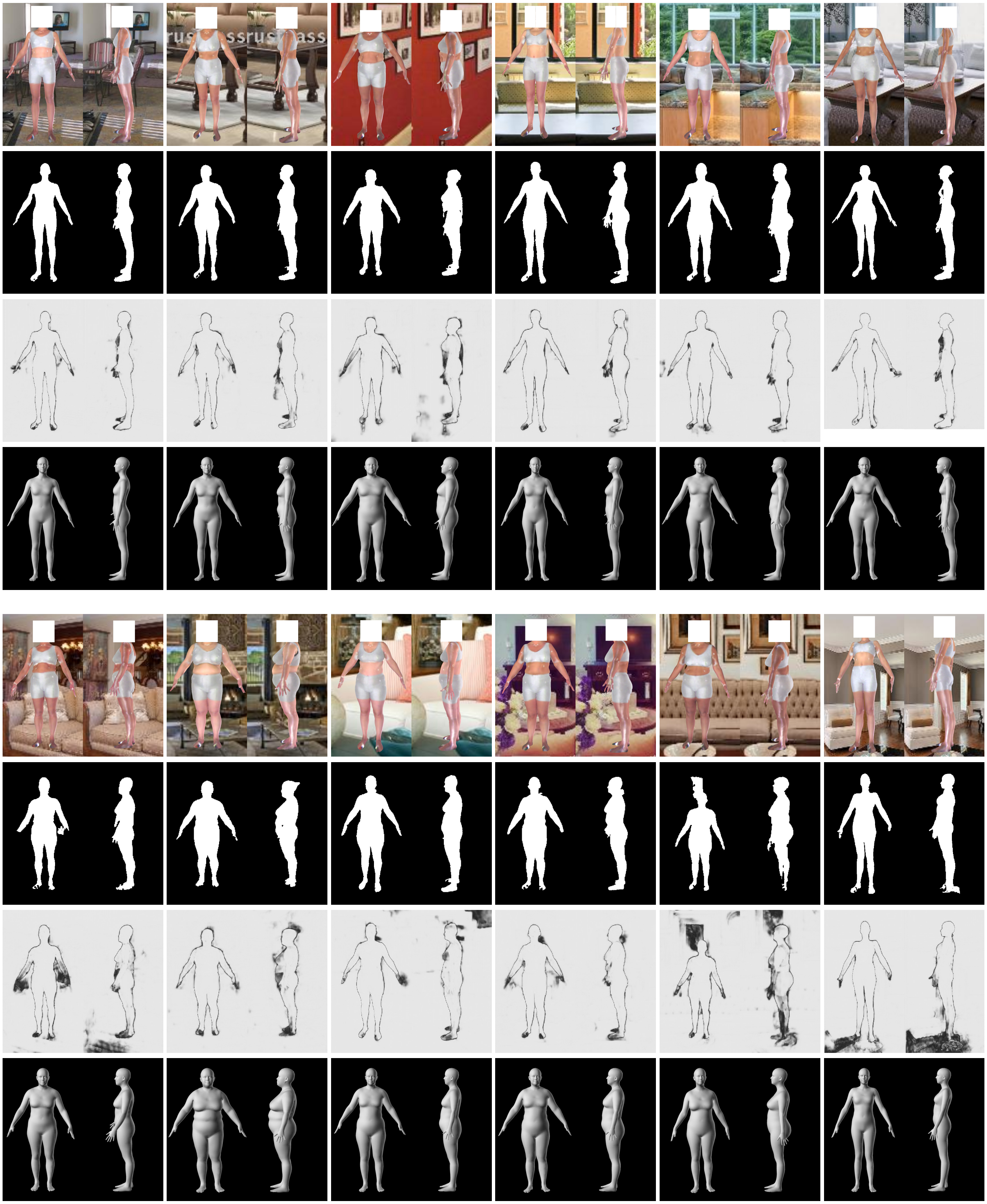


Figure 5. Qualitative results on female CAESAR images, part 1.

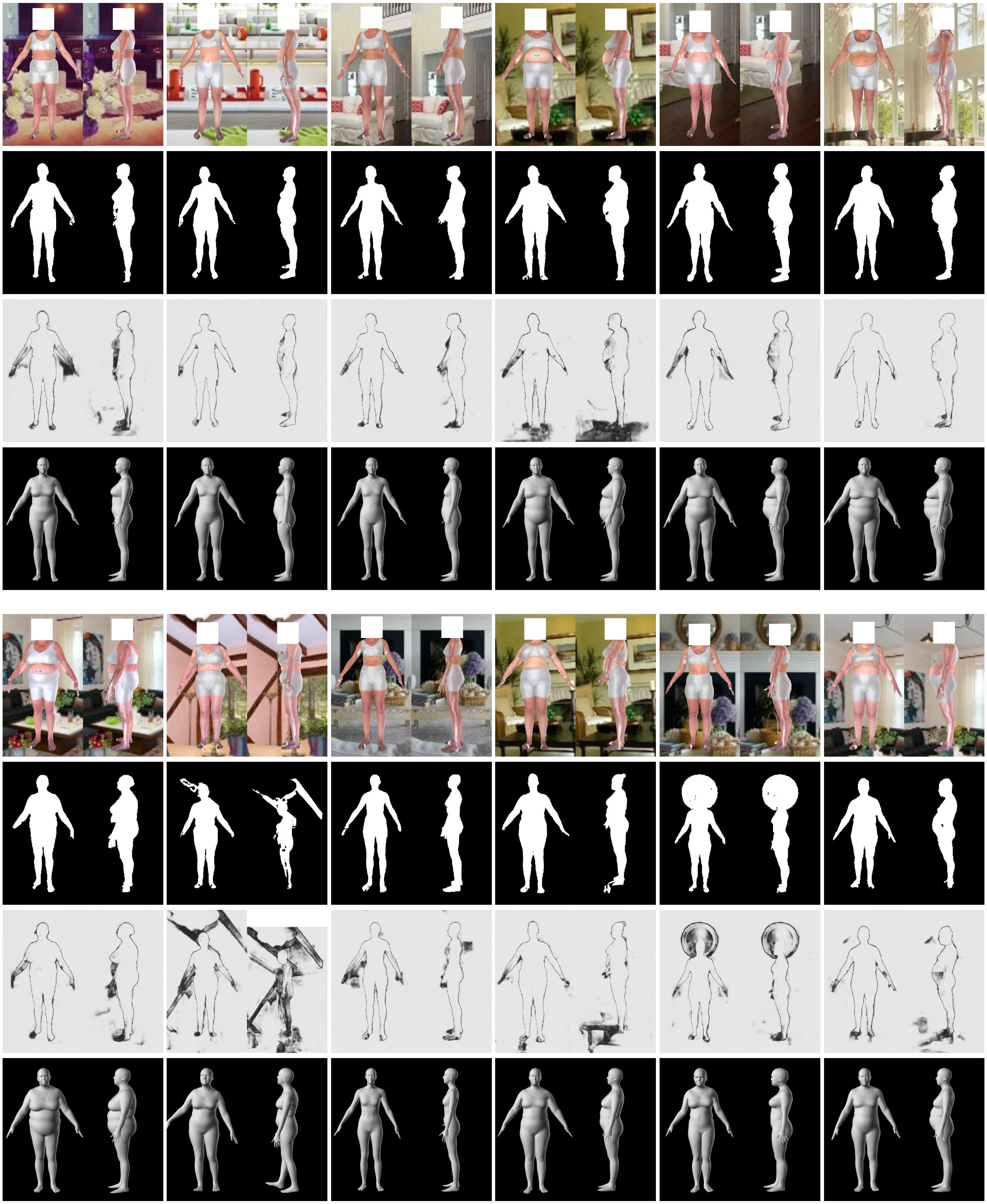


Figure 6. Qualitative results on female CAESAR images, part 2.

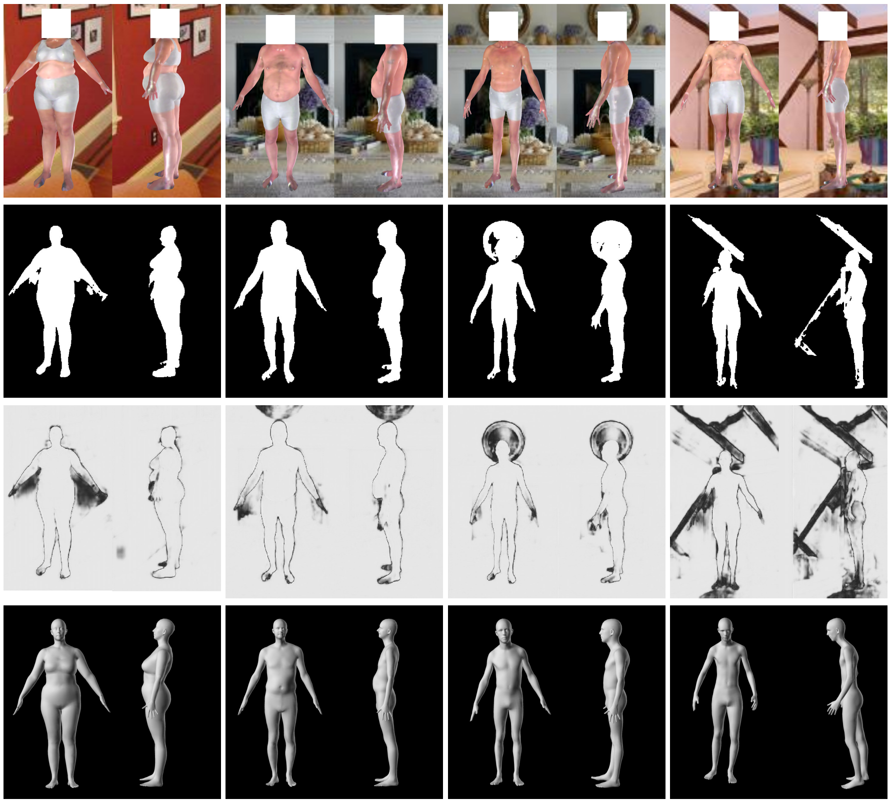


Figure 7. Problem cases. Large segmentation errors, although rare, can lead BaReNet to estimate 3D body shapes that don't accurately reflect the image.

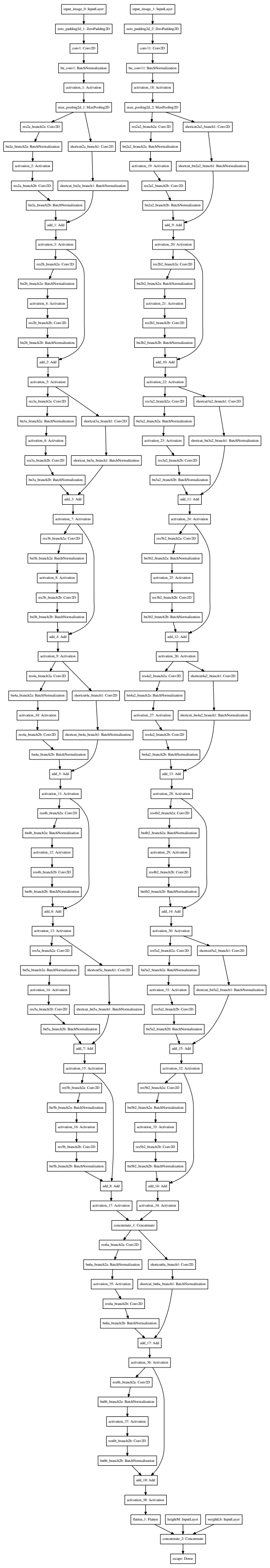

input_image_0: InputLayer
zero_padding2d_1: ZeroPadding2D
conv1: Conv2D
bn_conv1: BatchNormalization
activation_1: Activation
max_pooling2d_1: MaxPooling2D
res2a_branch2a: Conv2D
bn2a_branch2a: BatchNormalization
shortcut2a_branch1: Conv2D
activation_2: Activation
res2a_branch2b: Conv2D
shortcut_bn2a_branch1: BatchNormalization
bn2a_branch2b: BatchNormalization
add_1: Add
activation_3: Activation
res2b_branch2a: Conv2D
bn2b_branch2a: BatchNormalization
activation_4: Activation
res2b_branch2b: Conv2D
bn2b_branch2b: BatchNormalization
add_2: Add
activation_5: Activation
res3a_branch2a: Conv2D
bn3a_branch2a: BatchNormalization
shortcut3a_branch1: Conv2D
activation_6: Activation
res3a_branch2b: Conv2D
shortcut_bn3a_branch1: BatchNormalization
bn3a_branch2b: BatchNormalization
add_3: Add
activation_7: Activation
res3b_branch2a: Conv2D
bn3b_branch2a: BatchNormalization
activation_8: Activation
res3b_branch2b: Conv2D
bn3b_branch2b: BatchNormalization
add_4: Add
activation_9: Activation
res4a_branch2a: Conv2D
bn4a_branch2a: BatchNormalization
shortcut4a_branch1: Conv2D
activation_10: Activation
res4a_branch2b: Conv2D
shortcut_bn4a_branch1: BatchNormalization
bn4a_branch2b: BatchNormalization
add_5: Add
activation_11: Activation
res4b_branch2a: Conv2D
bn4b_branch2a: BatchNormalization
activation_12: Activation
res4b_branch2b: Conv2D
bn4b_branch2b: BatchNormalization
add_6: Add
activation_13: Activation
res5a_branch2a: Conv2D
bn5a_branch2a: BatchNormalization
shortcut5a_branch1: Conv2D
activation_14: Activation
res5a_branch2b: Conv2D
shortcut_bn5a_branch1: BatchNormalization
bn5a_branch2b: BatchNormalization
add_7: Add
activation_15: Activation
res5b_branch2a: Conv2D
bn5b_branch2a: BatchNormalization
activation_16: Activation
res5b_branch2b: Conv2D
bn5b_branch2b: BatchNormalization
add_8: Add
activation_17: Activation
input_image_1: InputLayer
zero_padding2d_2: ZeroPadding2D
conv11: Conv2D
bn_conv11: BatchNormalization
activation_18: Activation
max_pooling2d_2: MaxPooling2D
res2a2_branch2a: Conv2D
shortcut2a2_branch1: Conv2D
bn2a2_branch2a: BatchNormalization
activation_19: Activation
shortcut_bn2a2_branch1: BatchNormalization
res2a2_branch2b: Conv2D
bn2a2_branch2b: BatchNormalization
add_9: Add
activation_20: Activation
res2b2_branch2a: Conv2D
bn2b2_branch2a: BatchNormalization
activation_21: Activation
res2b2_branch2b: Conv2D
bn2b2_branch2b: BatchNormalization
add_10: Add
activation_22: Activation
res3a2_branch2a: Conv2D
shortcut3a2_branch1: Conv2D
bn3a2_branch2a: BatchNormalization
activation_23: Activation
shortcut_bn3a2_branch1: BatchNormalization
res3a2_branch2b: Conv2D
bn3a2_branch2b: BatchNormalization
add_11: Add
activation_24: Activation
res3b2_branch2a: Conv2D
bn3b2_branch2a: BatchNormalization
activation_25: Activation
res3b2_branch2b: Conv2D
bn3b2_branch2b: BatchNormalization
add_12: Add
activation_26: Activation
res4a2_branch2a: Conv2D
shortcut4a2_branch1: Conv2D
bn4a2_branch2a: BatchNormalization
activation_27: Activation
shortcut_bn4a2_branch1: BatchNormalization
res4a2_branch2b: Conv2D
bn4a2_branch2b: BatchNormalization
add_13: Add
activation_28: Activation
res4b2_branch2a: Conv2D
bn4b2_branch2a: BatchNormalization
activation_29: Activation
res4b2_branch2b: Conv2D
bn4b2_branch2b: BatchNormalization
add_14: Add
activation_30: Activation
res5a2_branch2a: Conv2D
shortcut5a2_branch1: Conv2D
bn5a2_branch2a: BatchNormalization
activation_31: Activation
shortcut_bn5a2_branch1: BatchNormalization
res5a2_branch2b: Conv2D
bn5a2_branch2b: BatchNormalization
add_15: Add
activation_32: Activation
res5b2_branch2a: Conv2D
bn5b2_branch2a: BatchNormalization
activation_33: Activation
res5b2_branch2b: Conv2D
bn5b2_branch2b: BatchNormalization
add_16: Add
activation_34: Activation
concatenate_1: Concatenate
res6a_branch2a: Conv2D
bn6a_branch2a: BatchNormalization
shortcut6a_branch1: Conv2D
activation_35: Activation
shortcut_bn6a_branch1: BatchNormalization
res6a_branch2b: Conv2D
bn6a_branch2b: BatchNormalization
add_17: Add
activation_36: Activation
res6b_branch2a: Conv2D
bn6b_branch2a: BatchNormalization
activation_37: Activation
res6b_branch2b: Conv2D
bn6b_branch2b: BatchNormalization
add_18: Add
activation_38: Activation
flatten_1: Flatten
heightM: InputLayer
weightLb: InputLayer
concatenate_2: Concatenate
sscape: Dense

Figure 8. SfSNet Keras architecture details, as generated by the Keras model visualization tool.