CSCI 1430 Final Project Report: Avatar

Avatar: Muhiim Ali, David Eskilson, Ziheng Huang, Charumathi Narayanan.
TA name: Sue An. Brown University

Abstract

We present an end-to-end pipeline for transforming real-world human subjects into animatable 3D avatars. Our system captures a stationary subject from four angles using an Azure Kinect depth camera, fuses the resulting point clouds into a unified 3D representation using CloudCompare, and reconstructs a surface mesh using the Ball-Pivoting Algorithm. The mesh is then refined in Blender to correct reconstruction artifacts such as holes and disconnected geometry, which would otherwise prevent stable animation. Finally, we fit the SMPL-X parametric body model to the cleaned mesh and transfer skinning weights via k-nearest-neighbor interpolation, producing avatars that deform consistently across multiple motion sequences. Our results demonstrate a practical human digitization pipeline that connects multi-view 3D reconstruction with avatar animation while highlighting the importance of mesh quality for downstream deformation stability.

1. Introduction

Creating an animatable 3D avatar of a real person is a fundamental challenge at the intersection of computer vision and computer graphics, with wide-ranging applications in immersive VR and AR experiences, digital communication, and content creation. Systems like Meta Codec Avatars [5] demonstrate the transformative potential of this technology, but they rely on highly specialized capture hardware and large-scale training infrastructure far beyond typical research settings.

The core difficulty lies in the chain of transformations required to go from raw sensor data to a moving, believable character: multi-angle depth captures must be precisely registered into a unified representation despite subject movement and scene noise, and the resulting point cloud must be converted into a clean triangle mesh free of holes, disconnected geometry, and topological artifacts. These reconstruction imperfections compound further at the animation stage, where even small topological defects can cause a skeletal rig to produce physically implausible deformations, making mesh quality a prerequisite for any successful animation pipeline.

We present a practical end-to-end pipeline that tackles each of these stages: capturing a stationary subject from four angles with Azure Kinect depth camera, fusing point clouds in CloudCompare, reconstructing a surface mesh via the Ball-Pivoting Algorithm (BPA), refining it in Blender, and fitting SMPL-X [8] with kNN skinning weight transfer to produce a fully animatable avatar.

2. Related Work

Multi-view depth fusion. Early work by Curless and Levoy [2] introduced volumetric TSDF fusion, later extended to real-time RGB-D operation by KinectFusion [7]. We use CloudCompare to register pre-segmented point clouds via manual alignment followed by the Iterative Closest Point (ICP) algorithm, prioritizing robustness over automation given our noisy, low-texture input.

Surface reconstruction. Poisson Surface Reconstruction [4] produces watertight meshes but is sensitive to normal estimation error in noisy scans. We instead employ the BPA [1], which directly triangulates the point cloud and proved more tolerant of our input quality.

Parametric body models and skinning. Our animation stage builds on SMPL [6] and its expressive extension SMPL-X [8], which factorize human shape and pose into low-dimensional coefficients compatible with motion-capture data. We transfer skinning weights from the fitted SMPL-X surface to scan vertices via kNN interpolation; more sophisticated approaches such as bounded biharmonic weights [3] or neural rigging [10] could improve quality around hands and clothing boundaries in future research.

End-to-end avatar creation. PIFu [9] and Meta Codec Avatars [5] demonstrate fully learned pipelines for human digitization, but they require large training datasets and specialized hardware. Our pipeline achieves animatable results by combining geometric reconstruction with a parametric

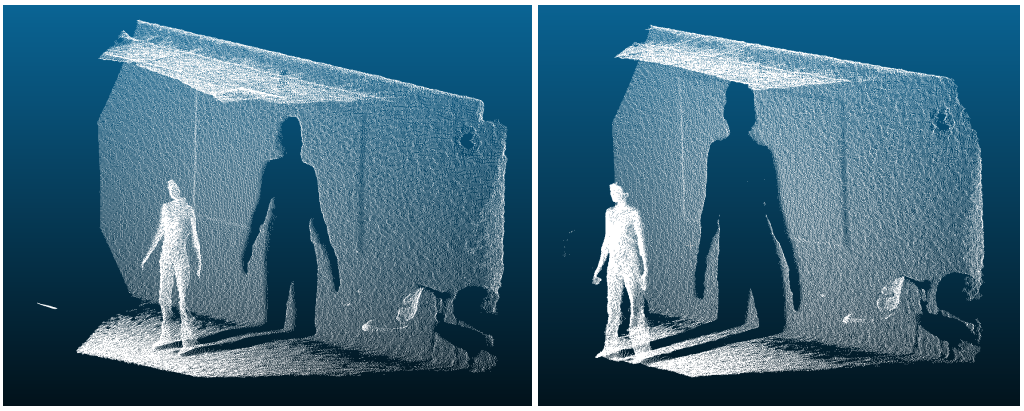

Figure 1. Examples of our raw data collection from both cameras. *Left:* 3D point cloud of the front of the model taken using Zed Mini camera. There is a notable lack of detail, including bumpy surfaces and poor capturing of the fingers and other details. This bumpiness is somewhat hard to see visually in small images, but it is clearer upon investigation using 3D point cloud viewing software such as CloudCompare. *Right:* 3D point cloud of the front of the model taken using Azure Kinect camera. Compared to the Zed Mini, the details are improved and the point cloud is more continuous.

body model, without learned priors beyond the SMPL-X shape space.

3. Method

The primary goal of the project is to go from data collected on a human model to a full animated 3D mesh; our novel end-to-end pipeline that solves this problem is composed of five key stages: data collection, unified point cloud, mesh generation, mesh refinement, and animation.

3.1. Phase 1: Data Collection

We used a Zed Mini and an Azure Kinect camera for our data collection. The Zed Mini camera is a stereo-based depth camera, whereas the Azure Kinect camera relies on light to detect depth. For the Zed Mini camera, we utilized the ZED SDK 5.3 from Stereolabs on a Windows machine. For the Azure Kinect camera, we made use of the Azure Kinect Sensor SDK in addition to the pyKinectAzure library to allow for extraction and saving of the 3D point cloud raw data, also running on a Windows machine.

Data collection was conducted in a spacious, well-lit environment (CIT 506). The human model in the scene held an A-pose to facilitate ease of animation in the rest of the pipeline. We conducted initial testing with both the Zed Mini and Azure Kinect, and found that the Kinect point clouds were more detailed than those of the Zed Mini, perhaps because the skin and clothing of the model were insufficiently textured to allow the stereo calculations to work effectively. Since both cameras produced 3D point clouds, we chose the Azure Kinect for the remainder of our testing due to its superior detail.

We tried multiple ways of capturing the model in the scene. Initially, we fixed the position of the camera and had the model first look straight at the camera, then rotate $90°$, $180°$, and $270°$ to allow for the capturing of four distinct angles. However, we encountered difficulties completing the combining point clouds phase with the raw data we collected here due to slight variations in the position of the model (for instance, slight variations in the positioning of the torso relative to the head and neck).

Subsequently, we had the model stay perfectly still while we moved the camera around them to capture four distinct angles: front, left, right, and back. Since the model remained still here, the point clouds were much easier to combine in the next phase, and the resulting combined point cloud was significantly less noisy.

The final output of this phase was a set of four 3D point clouds, taken with the Azure Kinect camera, that served as the input into the next phase where we actually combined them.

Shown in Figure 1 is an example of a 3D point cloud taken of the front of the model using both cameras, and it highlights the improved detail and quality of the Azure Kinect compared to the Zed Mini.

3.2. Phase 2: Unified Point Cloud

The input to this phase was the output of the previous phase: four point clouds, taken of the front, right, left, and back of the model, which included the background and other artifacts from the location in which we collected data.

In order to create a unified point cloud, we made use of CloudCompare software. We uploaded our point clouds to the software and then conducted manual work and used tools provided by the software to create a single point cloud containing a full, $360°$ representation of the model.

First, we used the "Segment" tool to separate out the raw point clouds into two point clouds: one just containing the model, and the other containing the rest of the scene data.

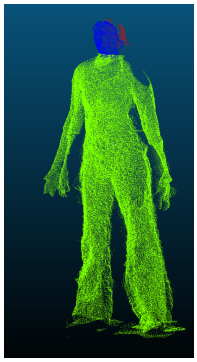

Figure 2. Front view of the unified 3D point cloud created from the second phase in our end-to-end pipeline.

We then only used the point cloud containing the model, and ignored the one with the scene data as it was not relevant to the project.

Next, we aligned the segmented point clouds of the model in an iterative process. We started with one point cloud and then manually aligned a second point cloud of a different angle using the "Translate/Rotate" tool. With the two mostly aligned point clouds, we made use of a tool that finally registers already aligned point clouds, which uses an implementation of the ICP algorithm; the algorithm makes use of Singular Value Decomposition (SVD) to find an optimal rotation and translation matrix between a subset of closest points between a reference and source point cloud, proceeding in an iterative manner by applying the optimal transformation and then recalculating the new optimal transformation to apply.

Once we had aligned two of the point clouds, we then aligned the third using the same process, and then the fourth. With the completion of this phase, we were left with the output of this phase: a single 3D point cloud, just containing the model.

It is important to note that this phase and the data collection phase were done iteratively, where we would collect some raw data, run through the unified point cloud phase, assess the quality, and then conduct more data collection to focus on issues in the output. For instance, sometimes when combining the point clouds, we noticed the model had moved slightly, so we then went back to collect more data with the subject remaining especially still.

Figure 2 shows the front view of the unified 3D point cloud we used for the remainder of the pipeline.

3.3. Phase 3: Mesh Generation

Taking the input to this phase of a single unified 3D point cloud, we needed to convert it into a single triangle mesh. We used Open3D implementations and experimented with multiple methods to do this and visually assessed results to determine the best method.

The first method we tried was Poisson Surface Reconstruction, which essentially solves for an indicator function using the normals of the 3D point cloud that indicates the shape of the surface. However, this technique is not robust to noisy point cloud data since the normals are angled in unpredictable and jarring ways, and when we attempted to create a mesh, it was highly noisy with many cavities and limited clear features.

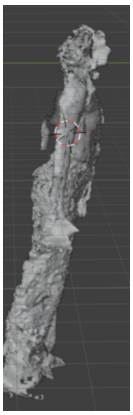

Figure 3. Mesh generated using the BPA. While clearly humanoid, significant noise and bumpiness remains, while the hands and fingers are poorly defined.

We then switched to using the BPA; the algorithm conceptually speaking involves continuously moving from a starting point on the mesh to the closest points to create triangles, analogous to rolling a ball along the surface of the 3D point cloud. While a significant amount of noise persisted—requiring refinement in the next phase—our mesh was clearly humanoid and provided a very solid foundation upon which to do further work.

Crucially, this phase was also done in tandem with the prior two phases, where we would run through all three initial phases, identify mistakes in the mesh such as excessive noisiness or overlap, and then go back to unifying point clouds or even collecting new data. This way, we refined our data collection and point cloud unification process and ensured the highest possible quality for our input data.

Exhibited in Figure 3 is a side view of the mesh generated from the point cloud in Figure 2.

3.4. Phase 4: Mesh Refinement

Despite our best efforts, the mesh we obtained after the mesh generation phase of our project had some significant issues, including holes and overlapped extremities caused by imperfect alignment of the underlying 3D point clouds. To fix these issues, we utilized Blender Remesh as a starting point to fill in holes, and then went beyond this using primitive polygon modeling to add more definition and clarity to the humanoid structure. We also used built-in smoothing tools to remove some of the noise.

The success of the refinement process is clearly visible

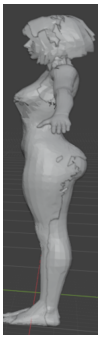

Figure 4. Refined mesh using an array of both automated and manual tools in Blender.

in Figure 4, which is much smoother than the mesh that was generated using the BPA in Figure 3.

3.5. Phase 5: Animation

Finally, with our refined mesh from the previous phase, it was time for animation. Given our limited experience with Blender and the inherent challenges of cleaning the mesh, we anticipated extensive difficulties getting a non-trivial animation. Therefore, we decided to use the SMPL framework, a parametric 3D body model.

We fit this framework to our mesh, requiring us to initialize parameters and preprocess our refined mesh for input. The fitting involved extensive experimentation, and we conducted numerous runs of the process to find the optimal parameters.

Once we had the SMPL-X fit in place, we animated the scan by applying motion-capture pose sequences frame by frame. At each frame, we ran an SMPL-X forward pass to obtain the deformed body surface, compute per-vertex displacements relative to the fitted rest pose, and propagate those displacements to the scan vertices via the precomputed kNN weights. To reduce jitter in the output, we applied Gaussian smoothing to the pose and translation sequences prior to deformation.

We iterated on the fitting parameters after visually assessing issues in the animation output, such as arms bending in a non-physically consistent manner, and adjusted initialization accordingly to achieve improved results. A more extensive discussion of these results and visualizations is contained in the following section.

4. Results

Our final pipeline successfully generated animatable 3D avatars from multi-view captures using the Azure Kinect system. The complete workflow consisted of multi-angle image and depth acquisition, point cloud generation, BPA-based surface reconstruction, mesh refinement in Blender, and final skeletal animation evaluation through both Mixamo and an SMPL-based animation pipeline. Figures 2 and 3

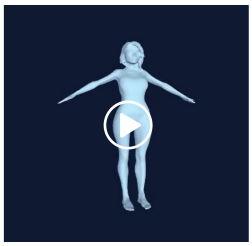

Figure 5. Rotation visualization of the reconstructed and refined avatar mesh generated from multi-view capture data.

illustrate the transition from raw point cloud captures to reconstructed meshes, while Figures 3 and 4 demonstrate the impact of mesh refinement and animation integration on the final avatar quality. Linked in Figure 5 is a video showing a full 360° view of our refined mesh. The video linked in Figure 6 displays an animation of our mesh generated using SMPL, while Figure 7 highlights fitting SMPL-X to our mesh.

Figure 6. Dance animation, created using our mesh and the SMPL framework.

The initial reconstruction outputs frequently contained disconnected geometry, noisy topology, surface holes, and unstable regions around thin structures such as hands, shoes, and clothing boundaries. To improve mesh quality, we applied a refinement workflow combining Blender Remesh operations, Shrinkwrap surface fitting, external mesh-cleanup tools, and manual topology correction. These modifications substantially improved mesh continuity and generated cleaner geometry suitable for downstream animation.

To evaluate animation compatibility, reconstructed avatars were imported into Mixamo and an SMPL-based animation framework for skeletal rigging and motion testing. We observed that topology quality and mesh cleanliness were critical factors for successful animation, as disconnected geometry and unstable topology could produce distorted skeletal deformation during playback. After refinement, the final meshes demonstrated stable animation and improved skeletal compatibility across multiple animation sequences.

We also needed to make empirically driven adjustments to our preprocessing of the mesh for fitting with SMPL. A core adjustment we made was the addition of Gaussian smoothing, which better allowed the parameters to pick up on details such as the hands and arms, where some noise per-

Figure 7. SMPL-X fitting results. *Left:* Input reconstructed scan. *Middle:* Fitted SMPL-X body model. *Right:* Overlay visualization showing alignment between the reconstructed mesh and the fitted body model.

Optimization Stage	Chamfer Distance
Initial Alignment	0.2103
Stage 1	0.0507
Stage 2	0.0294

Table 1. SMPL-X fitting convergence during optimization. The decreasing Chamfer distance indicates improved alignment between the reconstructed scan and the fitted body model.

sisted in our mesh. Supplementary demo videos—including a before and after of our SMPL pipeline with improved data preprocessing—are externally linked in Figure 8 and not inserted directly due to PDF video playback limitations.

The decreasing Chamfer distance values (Table 1) demonstrate that the optimization process progressively improved the alignment between the reconstructed scan and the fitted SMPL-X model, contributing to more stable downstream animation results.

4.1. Technical Discussion

One of the primary challenges in our pipeline was balancing reconstruction completeness with mesh quality and animation compatibility. While BPA reconstruction was effective at generating usable surfaces from dense point clouds, the resulting meshes often contained holes, disconnected geometry, and noisy topology in sparse regions. These artifacts became particularly problematic during downstream rigging and animation, where unstable topology could produce distorted skeletal deformation or failed animation attempts.

The mesh refinement process introduced several important trade-offs. Blender Remesh operations improved topological consistency and generated cleaner geometry suitable for animation, but excessive re-meshing occasionally smoothed away finer geometric details from the original reconstruction. We also experimented with additional filtering and cleanup methods, but some approaches introduced invalid surfaces or large mesh holes. As a result, manual mesh correction remained necessary to preserve both structural integrity and visual quality, and this stage became one of the most time-consuming parts of the project.

Another challenge involved integrating reconstructed meshes into the SMPL-based animation workflow. The animation pipeline was highly sensitive to mesh cleanliness and skeletal compatibility, where small topological defects or disconnected regions could produce unstable animation artifacts during playback. Overall, the final pipeline demonstrated that combining multi-view capture, mesh refinement, and skeletal animation could generate functional animatable avatars while substantially improving mesh usability and animation stability.

5. Conclusion

We developed an end-to-end pipeline to transform real-world subjects into animatable digital avatars. Using a *Microsoft Azure Kinect* camera, we captured multi-angle depth data, fused the resulting point clouds in *CloudCompare*, and reconstructed surfaces via the *BPA*. The final mesh was refined in *Blender* and rigged using the *SMPL* parametric model to enable realistic human motion.

This workflow successfully bridges the gap between static 3D reconstruction and dynamic animation. By aligning meshes of an actual subject with the industry-standard *SMPL model*, we proved that consumer-grade depth sensors can produce high-quality outputs compatible for motion capture pipelines, eliminating the need for manual character modeling.

Future Impact: This pipeline advances the process of 3D content creation, offering a scalable blueprint for personalized avatars in VR/AR and digital communication. Future iterations could automate the entire pipeline, allowing for live translation from a depth stream to a production-ready digital presence.

References

[1] Fausto Bernardini, Joshua Mittleman, Holly Rushmeier, Cláudio Silva, and Gabriel Taubin. The ball-pivoting algorithm for surface reconstruction. *IEEE Trans. Vis. Comput. Graphics*, 5(4):349–359, 1999. 1

[2] Brian Curless and Marc Levoy. A volumetric method for building complex models from range images. In *Proc. SIGGRAPH*, pages 303–312, 1996. 1

[3] Alec Jacobson, Ilya Baran, Jovan Popović, and Olga Sorkine. Bounded biharmonic weights for real-time deformation. *ACM Trans. Graphics (Proc. SIGGRAPH)*, 30(4):78:1–78:8, 2011. 1

[4] Michael Kazhdan, Matthew Bolitho, and Hugues Hoppe. Poisson surface reconstruction. In *Proc. SGP*, pages 61–70, 2006. 1

[5] Stephen Lombardi, Jason Saragih, Tomas Simon, and Yaser Sheikh. Deep appearance models for face rendering. *ACM Trans. Graphics (Proc. SIGGRAPH)*, 37(4):68:1–68:13, 2018. 1

Figure 8. Before-and-after improved SMPL fitting comparison of reconstructed avatar mesh movement. *Left:* Initial attempt at animation of the refined avatar mesh. Of note, while the animation is a great start, it is not completely physically accurate; for instance, the arms bend in unnatural ways and move through the figure's torso. *Right:* Final animation playback generated through the SMPL-based animation pipeline. After we refined the SMPL fitting process, the animation looks more natural, including the arms, which behave much more along the lines one would expect.

[6] Matthew Loper, Naureen Mahmood, Javier Romero, Gerard Pons-Moll, and Michael J. Black. SMPL: A skinned multi-person linear model. *ACM Trans. Graphics (Proc. SIGGRAPH Asia)*, 34(6):248:1–248:16, 2015. 1

[7] Richard A. Newcombe, Shahram Izadi, Otmar Hilliges, David Molyneaux, David Kim, Andrew J. Davison, Pushmeet Kohli, Jamie Shotton, Steve Hodges, and Andrew Fitzgibbon. KinectFusion: Real-time dense surface mapping and tracking. In *ISMAR*, pages 127–136, 2011. 1

[8] Georgios Pavlakos, Vasileios Choutas, Nima Ghorbani, Timo Bolkart, Ahmed A. A. Osman, Dimitrios Tzionas, and Michael J. Black. Expressive body capture: 3D hands, face, and body from a single image. In *CVPR*, 2019. 1

[9] 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

[10] Zhan Xu, Yang Zhou, Evangelos Kalogerakis, Chris Landreth, and Karan Singh. RigNet: Neural rigging for articulated characters. *ACM Trans. Graphics (Proc. SIGGRAPH)*, 39(4):58:1–58:14, 2020. 1

Appendix

Team contributions

Please describe in one paragraph per team member what each of you contributed to the project.

Muhiim Ali I was solely responsible for implementing the animation pipeline from end to end. A significant portion of my time was spent researching state-of-the-art methods in 3D animation and experimenting with various animation tools, many of which failed but were necessary for building something that worked in the end. I also contributed to the data collection process, assisting in capturing images using the camera, and created a 37-second demo video to assist with the presentation.

David Eskilson I took primary responsibility for the second phase of our pipeline, combining point clouds from different angles into a single point cloud of the scene model. This included manual segmentation and alignment as well as tuning and use of the built-in ICP algorithm to improve results. I also had dominion over the poster, including the content on the poster—text, structure, graphics, and more—as well as the overall design and aesthetic. Finally, I played a key role in data collection, contributing to the design of the scene's layout, model positioning, and lighting (both initially and using insights I gleaned from my work with CloudCompare).

Ziheng Huang Independently handled ZED Mini setup, depth capture, and multi-angle data acquisition for 3D reconstruction experiments. Led the end-to-end 3D reconstruction pipeline, including point cloud preprocessing, mesh refinement, and reconstruction artifact correction. Performed extensive manual mesh cleanup and topology refinement in Blender, improving geometry quality and surface continuity beyond automated reconstruction outputs. Conducted detailed evaluation of Blender remeshing techniques and mesh modifiers to

generate cleaner avatar meshes with improved visual quality and structural consistency. Explored manual rigging workflows on reconstructed meshes and validated animation compatibility through Mixamo integration and deformation testing on upper-body motion sequences.

Charumathi Narayanan I developed a Python interface for the Azure Kinect to automate .ply data capture and was part of the multi-angle data collection process. I also executed the surface reconstruction via BPA and conducted mesh refinement experiments in Blender. Additionally, I specifically evaluated the efficacy of Remesh and Shrinkwrap modifiers for topological optimization and utilized Mixamo to evaluate the mesh's compatibility with standard rigging pipelines.

SRC Response

We received an SRC critique of our project proposal, linked here. Below, we respond to the critique our group received; we discuss what feedback we incorporated, what we decided not to change, and why.

Response to Criticism 1

We concur with the criticism that our initial stakeholder analysis in our proposal is too narrow. To account for this, we formally decided to include scene actors, animation laborers, and cultural communities connected to the framing of *Avatar: The Last Airbender* as additional stakeholders. We recognize that scene actors whose likeness is used should have clear rights such as data deletion, while animation laborers' pipelines may be altered and they may need to focus on alternative work that composes our pipeline rather than standard processes. And, we will ensure that all media generated from our pipeline that is publicly released and associated with a cultural community such as ATLA is explicitly noted as unofficial and synthetic.

Response to Criticism 2

We agree that our original proposal did not clearly define consent and data governance procedures for 3D biometric capture. To address this concern, all image and depth data will only be collected with explicit participant consent, and participants will be informed about how their reconstructed assets will be used within the course project. All captured data, point clouds, meshes, and animation outputs will be stored on private devices or restricted-access repositories and will not be publicly distributed or reused outside the project without additional permission. Participants may also request deletion of their captured data and generated assets at any time during and after the project.

Response to Criticism 3

To address cultural concerns, we used the SMPL model as a geometric guardrail. By mapping capture data to a neutral, anatomically standard rig, we prevent stylistic caricature and keep the focus on biomechanical motion rather than cultural appropriation. Additionally, we restricted input to local Azure Kinect captures. This establishes a "Physical Proof of Ownership," ensuring the pipeline only processes data from a user-present capture session rather than unverified web images.

Response to Criticism 4

We acknowledge that our current success criteria were primarily technical. While the project successfully established a 3D reconstruction baseline, it lacked integrated social impact metrics. To address this, future work could include frameworks such as Physical Handshake Protocols to ensure hardware-level consent during capture and Local-Only Serialization to resolve ownership ambiguity by keeping identity data under the user's physical control.

Response to Criticism 5

We acknowledge the fact that some of our initial assumptions had introduced exclusion and bias. The adjustment to our initial proposal of using depth cameras, and specifically the light-based Azure Kinect, means that inequitable access to optimal stereo conditions or lack of textured clothing is not a barrier to someone being modeled effectively. Furthermore, mobility needs can be addressed given that studio conditions are no longer required, simply an open space is. Finally, while the A-pose and T-pose are ideal for animation, we still anticipate being able to get very solid results with our pipeline without the pose as no step explicitly requires it (although it may be somewhat more labor intensive), meaning our pipeline overall has been made relatively accessible.