Joint Virtual Reality Conference of EGVE - EuroVR (2013)
B. Mohler, B. Raffin, H. Saito, and O. Staadt (Editors)

Personalized Animatable Avatars from Depth Data

Jai Mashalkar Niket Bagwe Parag Chaudhuri

Department of Computer Science and Engineering, IIT Bombay, India

Abstract

We present a method to create virtual character models of real users from noisy depth data. We use a combination of four depth sensors to capture a point cloud model of the person. Direct meshing of this data often creates meshes with topology that is unsuitable for proper character animation. We develop our mesh model by fitting a single template mesh to the point cloud in a two-stage process. The first stage fitting involves piecewise smooth deformation of the mesh, whereas the second stage does a finer fit using an iterative Laplacian framework. We complete the model by adding properly aligned and blended textures to the final mesh and show that it can be easily animated using motion data from a single depth camera. Our process maintains the topology of the original mesh and the proportions of the final mesh match the proportions of the actual user, thus validating the accuracy of the process. Other than the depth sensor, the process does not require any specialized hardware for creating the mesh. It is efficient, robust and is mostly automatic.

Categories and Subject Descriptors (according to ACM CCS): Computer Graphics [I.3.7]: Three-Dimensional Graphics and Realism—Virtual Reality; Information Interfaces and Presentation [H.5.1]: Multimedia Information Systems—Artificial, augmented, and virtual realities

1. Introduction

Virtual characters are used extensively in virtual, augmented and mixed reality applications. They often serve as a proxy for the real user and the extent of their resemblance to the user in shape, appearance and movement is crucial to the sense of presence that the user feels in MR environments. This has led to a lot of research work in area of creation and animation of virtual avatars. Recent availability of cheap depth sensors like the Microsoft Kinect [Kin13] have led to a large number of techniques to capture the shape and motion of a user. However, the techniques that generate a model of the user, do not discuss the suitability of the model for animation. On the other hand, techniques that capture the motion, generally expect a proper animatable model as input and hence do not discuss it either. Professional animators and modelers are very particular about the kind of topology a character mesh has so that it animates properly [Wil10].

In this paper, we present a system that creates a mesh model of the user by deforming a template mesh to match a point cloud captured from a system of depth cameras. The process maintains the topology of the template mesh in the process thereby producing an avatar mesh of proper topology at the end of the process. We validate the shape of the mesh by comparing with actual anthropological measurements from the real user with measurements on the mesh. We also capture motion data from a single depth camera and use that to animate the created mesh. The entire process is mostly automatic and other than the depth cameras (Microsoft Kinect cameras), no specific hardware is required.

We first introduce the related work in the area in the next section. In subsequent sections we describe our system in more detail with the help of a running example. Finally, we present results to validate our model reconstruction and to show its use in animation.

2. Background

First we discuss work related to reconstruction of humans from scanners and depth sensors. Real-time scanning of static scenes using the Microsoft Kinect has recently been demonstrated by Newcombe et al. [NIH*11]. It is well known that the iterative closest point (ICP) algorithm and its variants can be used to locally align and register overlapping rigid point clouds [BM92]. Scanning of humans offers the challenge of non-rigid registration of overlapping point clouds. Chang and Zwicker [CZ11] present work to globally register dynamic range scans of articulated mod-

DOI: 10.2312/EGVE.JVRC13.025-032

els. This work has been recently improved upon by Cui et al. [CCNS12] by utilizing a probabilistic scan alignment model. Both these techniques require computationally intensive optimizations and produce models with topology unsuitable for animation. Tong et al. [TZL*12] present a system to scan a user using three Kinect cameras with the user standing on rotating turntable. They also utilize a global non-rigid registration algorithm augmented with a rough template constructed from the first depth frame. They even demonstrate that their model is animatable. Even though we share some similarities with this system (use of multiple Kinect cameras), we do not directly mesh the point cloud obtained from scanning but use a single template mesh instead to get better mesh topology. Weiss et al. [WHB11] estimate the model of a user by fitting the parameters of a SCAPE model [ASK*05] to depth data and image silhouettes obtained from a single Kinect. This method differs from us in that it requires the use of a parametrized database of models, whereas our method only requires one template model to work.

Since we deform a template mesh to fit a point cloud, we now look at work related to this area. Koo et al. [KCKC04] and Jeong et al. [JK02] present methods that follow a shrink and smooth approach. Given a vertex on the template mesh, a nearest point on the point cloud is found, and the vertex is moved closer to that point. After doing this for all vertices of template mesh, the result is smoothed. This process is iteratively repeated. We found that these methods do not scale very well for the dense point clouds that we deal with in our system. Another set of methods mentioned in [SCOL*04] take the help of Laplacian coordinates to deform the mesh. Laplacian coordinates capture the local surface detail around any vertex. A system of linear equations is created, to keep the Laplacian coordinates for each point the same, and to satisfy user defined constraints. This is solved using least squares, to get an optimal solution which maintains the Laplacian coordinates and adjusts the mesh to the point cloud. Stoll et al. [SKR*06] improve this basic method by adding an iterative solver. After each solution, correspondences are found between the mesh and the point cloud. These are used to add additional constraints, and the system of equations is solved again. This makes the input mesh match the surface of point cloud closely. Since some parts of the point cloud may be inaccurate, there is a provision to exclude correspondences from certain parts of the mesh. We first divide the template mesh into six parts, namely, torso, neck, head, legs, elbows and hands. Each of these parts is fit to the corresponding segment of the point cloud and blended together. Subsequently we use an iterative Laplacian solver to refine the fit of the template mesh to point cloud. This two stage process of the part based fitting and the Laplacian refinement gives additional robustness to our system as explained later.

There have also been previous attempts at automatic creation of virtual avatars from multi-view video. Ahmed et al. [ADAT*05] fit a template mesh to silhouettes obtained from multi-view video and then texture and animate the avatar. Though the output produced is similar to ours, their setup requires synchronized multi-view video and more extensive processing. Performance capture methods that directly fit a scanned users mesh to multi-view video silhouettes can recover motion of the person without doing explicit kinematic tracking [DAST*08] or by recovering skeleton geometry first and then refining the surface template fit [GSDA*09]. These require more accurate template models to bootstrap the process, require significantly longer computations as they must process multiple video streams and require more expensive hardware.

Our template is a very generic mesh and we have used two template meshes (one for each gender) to generate all the examples in this paper. It requires no additional tuning. Our depth cameras do not have to be synchronized, however, they have to be calibrated initially which can be easily done using standard computer vision algorithms [Zha00]. A pairwise transformation between each pair of depth cameras can be estimated by using ICP. The computation required to fit the template mesh to the depth data is not very intensive. The motion capture in our system is from a single depth camera, though this can be easily replaced by any available motion capture setup.

3. System Overview

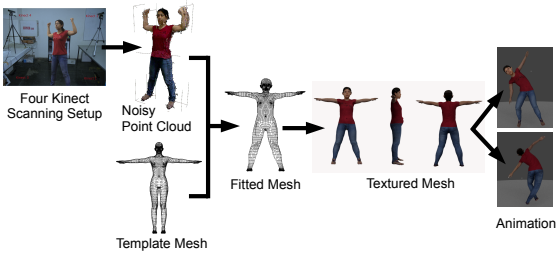
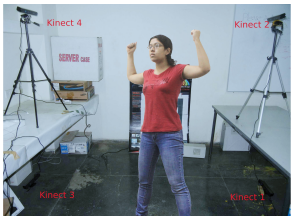


Figure 1: *System Overview*

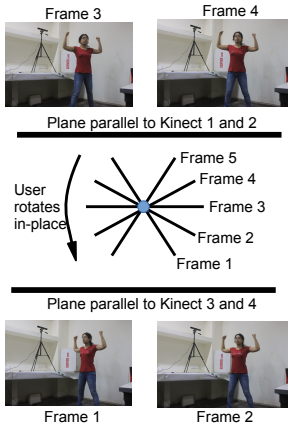
A brief overview of our system pipeline is shown in Figure 1. We use a four Kinect setup to scan the user. This generates a noisy point cloud. We fit our template mesh to this point cloud, to get a fitted mesh with correct topology. This mesh is then textured and rigged for use as a personalized avatar. The template mesh we use are the base female and male meshes taken from the MakeHuman Software [Mak13]. We explain our scanning setup in Section 4, followed by the details of two stages of the mesh fitting process in 5. In Section 6, we explain the details of texture mapping the fitted mesh. We conclude the description of our system by explaining the rigging, motion capture and animation in Section 7.

4. Point Cloud Capture

We use a set of four Microsoft Kinects to capture the point cloud of the user. The position of the four Kinects is shown in Figure 2(a). The Kinect cameras are first calibrated to determine the rigid transformation between their camera coordinate frames [Zha00, BM92].



(a)



(b)

(c)

Figure 2: *Capturing the point cloud of the user using the four Kinect setup. (a) shows the four Kinect setup, (b) shows the point cloud capture protocol and (c) the resulting noisy point cloud.*

The user stands in the space between the four Kinects and rotates in place in five steps of approximately 30 degrees. The order of user poses that make up the capture sequence protocol are shown in Figure 2(b). The user has to stand in the pose shown to avoid capture errors due to self occlusion. This capture sequence generates four partial point clouds of the user per frame. These 20 clouds are then registered and merged [TZL*12, CZ11] to generate the point cloud of the user, as shown in Figure 2(c). As can be seen this point cloud is noisy. We can filter the noise and perform meshing of the point cloud [KBH06] at this point to directly generate a model. As can be seen in Figure 3, this model has a very random distribution of triangles of all sizes which makes it unsuitable for animation. It can also be seen that facial features are completely lost in this reconstruction.

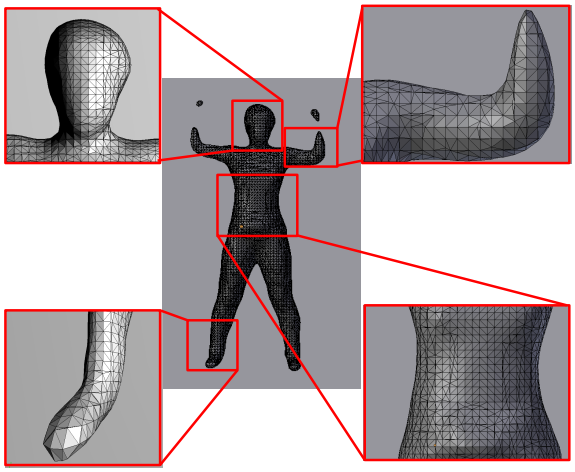


Figure 3: *Direct meshing of the point cloud using Poisson surface reconstruction produces a mesh with an arbitrary distribution of triangles as can be seen in many parts of the mesh. This makes the mesh topology unsuitable for animation.*

5. Fitting the Template Mesh

We fit the template mesh to the point cloud in the previous section in two stages.

5.1. Stage 1: Piecewise Mesh Fitting

In the first stage we divide the mesh and the point cloud into sections using pre-defined clipping planes. The sections correspond to disjoint body parts like torso, head, legs, upper and lower arms (see Figure 4). The 6 planes used to section the template mesh are fixed. The position of planes used to section the point cloud can be adjusted by the user. This is a simple $1D$ adjustment and consists for moving the plane along its normal to the appropriate location on the point cloud.

Each section of the mesh and point cloud is then automatically subdivided into equally spaced smaller segments. We divide the torso and legs into 6 segments each, arms into 5 segments and the head into 2 segments. Each segment of the mesh is then transformed to the corresponding segment of the point cloud by computing the rigid transformation that aligns the bounding box of the segments (see Figure 5). We have arrived at number of segments experimentally and the same number of segments were used for all examples. It should be noted that if the segments are made too small, due to noise there is a large variation in the transformation of adjacent segments, and the final mesh is not appropriately smooth. Vertex blending across segments ensures continuity and smoothness is maintained as the mesh is piecewise deformed to follow the point cloud. Excessive noise can lead to unreal dimensions for some segments. To prevent this,

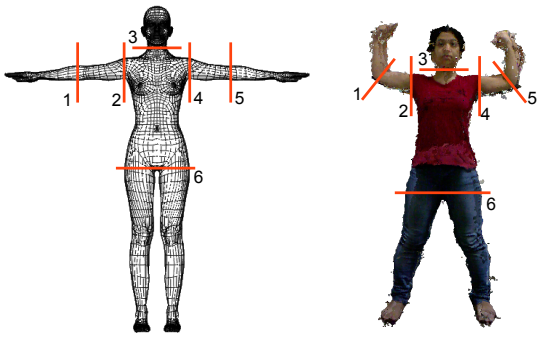


Figure 4: *Corresponding section planes on the template mesh and the point cloud divide them into corresponding sections.*

automatic checks for normal body ratios are added as constraints to the process. The final mesh that is obtained from

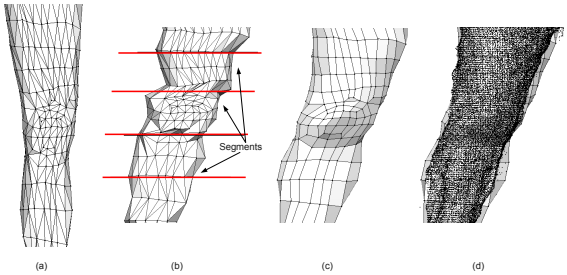


Figure 5: *A segment on the mesh deforms to fit the corresponding segment on the point cloud. (a) show the original mesh, (b) shows the segments after they have been transformed, (c) shows the blended transformed segments and (d) shows the overlap with the point cloud.*

this stage can be seen in Figure 6. Notice that though this mesh matches the point cloud at a gross level, finer level matches like the proportions of the face do not match the point cloud. In order to obtain a finer fit, we use this mesh to bootstrap the stage 2 fitting that is based on a Laplacian deformation framework.

5.2. Stage 2: Laplacian Mesh Fitting

In this stage, we deform the mesh obtained from the previous stage in a manner that moves every mesh vertex closer to the points on the point cloud in its vicinity, resulting in a mesh that closely fits the point cloud. We use a Laplacian formulation [SCOL*04] to achieve this. Laplacian coordinates, d are calculated as $d = Lp$, where p defines the vertex positions in the template mesh, and L is its corresponding Laplacian matrix. In order to get a close fit of the point cloud, vertices v_i are constrained to positions q_i, which are found by calculating an average position of the points within a certain radius

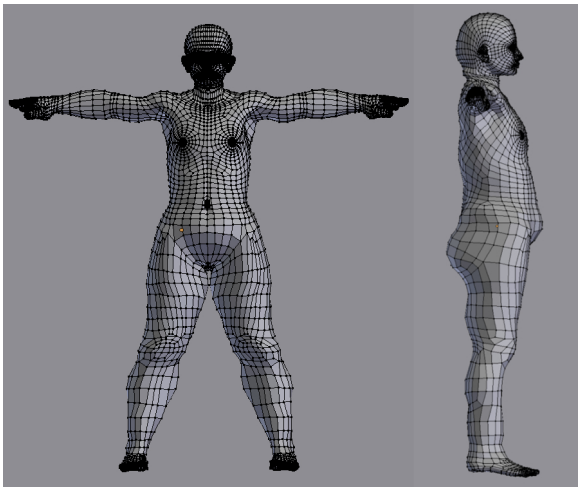
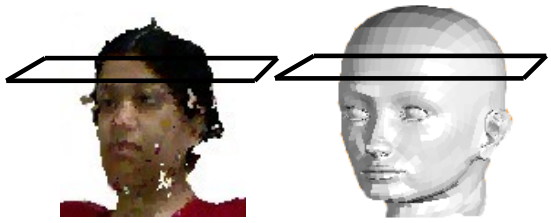


Figure 6: *Final mesh after Stage 1 piecewise mesh fitting.*

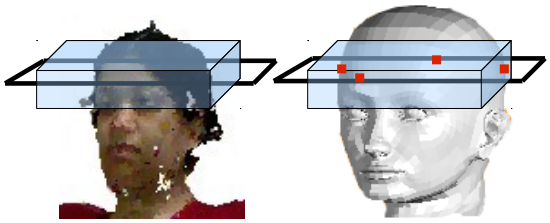
of the vertex. Displacement projected onto the vertex normal is used to obtain the new position q_i of the vertex [SKR*06]. These constraints are added to the Laplacian system as constraints of the form, $c_i v_i = c_i q_i$, where c_i is the weight of the constraint.

Since Laplacian coordinates are not rotation invariant, parts of the template mesh can be rotated to match the rotations of the mesh obtained from stage 1. Scaling of Laplacian coordinates is calculated as $s = (x_p/x_m + y_p/y_m + z_p/z_m)/3$, where x_p, y_p and z_p are the dimensions of the point cloud and x_m, y_m and z_m are the dimensions of the template mesh. Scaling of specific parts of the mesh such as hands and legs can also be adjusted interactively by the user, using planes as shown in Figure 7(a) to identify corresponding regions. A bounding box of the vertices within a specific radius of the planes is used to determine the scaling. The Laplacian equation $d = Lp$ is then modified as $d = S \cdot Lp$, where $S = (s_i)$, where s_i is the scaling of the vertex v_i. This system of equations is solved using least squares to get final positions of vertices.

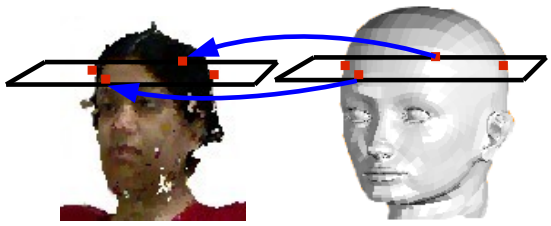
Before proceeding to the iterative step, correspondences need to be specified for finer features such as eyes, nose which are not matched in the first stage. A set of planes as shown in Figure 7 is used to automatically detect corresponding points, for features that are not clearly visible or very noisy in the point cloud. Points within a certain distance of the planes (see Figure 7(b)) are selected on the point cloud and template mesh. For four extreme points from selected points on the mesh, points at similar relative position are found amongst selected points on the cloud, as shown in Figure 7(c). If there are no points near the expected positions, virtual points are assumed. Since the dimensions of the plane are flexible, noisy areas near important features can be avoided, by adjusting the size of the planes. The step of determining new position based on closest points is omitted



(a) Corresponding section planes on the point cloud and mesh.



(b) Region around corresponding section planes selected and bounding box extents on the plane marked on the mesh.



(c) Corresponding points identified on the point cloud.

Figure 7: *Process of determining correspondences for the Laplacian iterative refinement.*

for areas of point cloud that are noisy, and areas of mesh that have specific details such as toes, fingers. Such features get lost in the iterative step and are maintained by adding constraints of the form $v_i = q_i$, where q_i is the position of the vertex v_i in the mesh obtained from the first stage. Note that this avoids direct clicking of correspondences between the mesh and point cloud and instead only relies on placement of 2 or 3 section planes, which can move only along their normals.

The iterative step is repeated around 3 to 6 times to get a proper fit of the point cloud. If this step is not bootstrapped with the first stage, around 30 to 40 correspondence pairs need to be specified to get a reasonable mesh. In spite of that, finer features like toes, fingers and areas where point cloud is noisy often get distorted. Since input of the first stage is given to the Laplacian, much fewer correspondences are required as input, which obtained as explained above. We have intuitive interfaces to move the section planes and provide all such input.

It can be seen in Figure 8 that the final mesh obtained after this stage has perfectly maintained the topology of the template mesh and is suitable for animation.

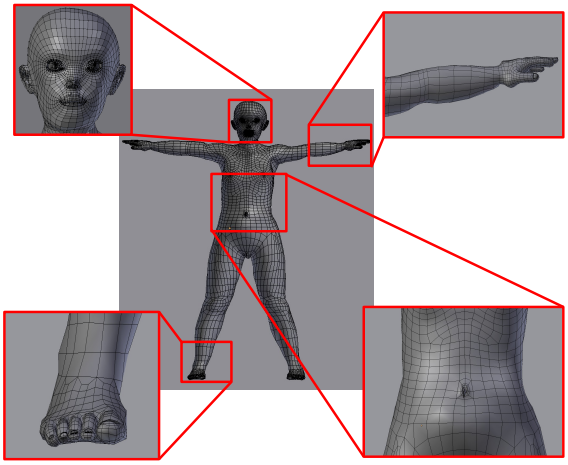


Figure 8: *Our two stage fitting process produces a mesh that maintains the topology of the original template. This ensures that the fitted mesh topology suitable for animation.*

6. Texturing

Proper texturing is needed to make the avatar resemble the user and heighten the sense of presence caused by the avatar. We define line guides on the texture image to correspond to the section planes on the mesh, as created during the stage 1 fitting (see 5.1). These are used to automatically align the texture to the mesh. We then generate the u-v parametrization for the mesh by projecting the mesh onto the textures. For this to work, the texture images need to be of the person standing in a T-pose (see Figure 9a and 9b).

The face texture is separately assembled from front and side view images of the user's face so that it can be wrapped around the mesh face (see Figure 9c). The approach followed in [LMT98] is used to create a blended texture image from front and side view images. The side view images are deformed to be connected to the front view image along user defined feature points. The images are then blended using the multi-resolution spline technique from [BA83]. To map this texture on the face, similar procedure is followed with the u-v coordinates of the face. Projections of vertices in front and side view are used to initialize the u-v coordinates. The side views are joined to the front view along user defined feature points, as done for the images. To achieve exact mapping of facial features, user defined feature points are used to adjust the u-v coordinates using bilinear interpolation.

These texture images can be captured during the point cloud capture phase. In order to get seamless textures, the texture border regions are extended from the edge of the texture image and blended by using Gaussian pyramids [BA83].

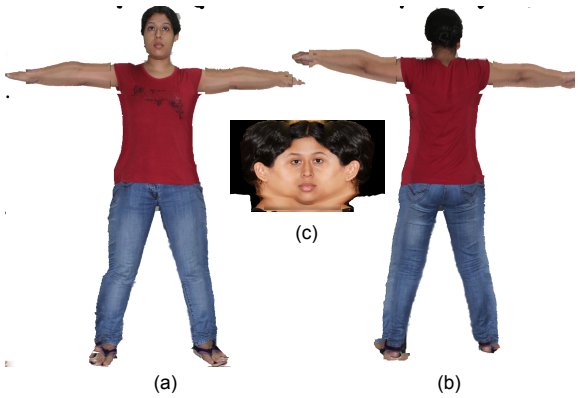


Figure 9: *(a) and (b) show textures for the mesh body. (c) shows the texture for the mesh face.*

7. Motion Capture

To make the meshes suitable for animation they must be rigged. Rigging requires the computation of skin weights, which can be computed automatically [BP07,Ble13]. We use the OpenNI drivers and the NiTE framework [Ope] to get the tracked 2D skeleton from the Kinect sensor. The track data is streamed to our real-time motion retargeting plugin in Blender that applies the motion in 3D to the rigged mesh in order to animate it. The motion can be saved to a standard BVH file if required, for later use. Figure 10 shows a frame

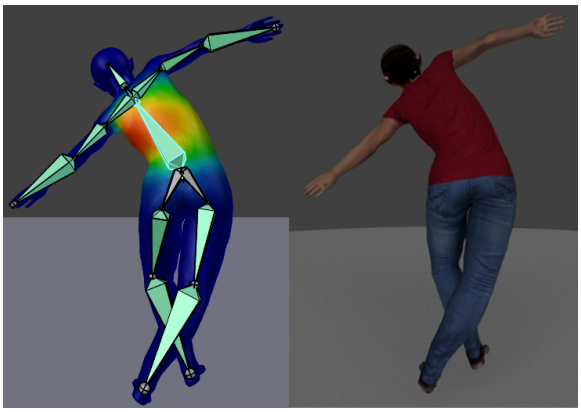


Figure 10: *Rigging and skinning the mesh allows us to animate it in real-time using motion captured from a single depth camera.*

from an animation sequence. The left image shows the rig and the skin weights for the torso on the mesh whereas the right image shows the actual rendered frame.

8. Results

We first compare our meshes with those generated by direct meshing of the point cloud data using Poisson surface reconstruction. The Poisson mesh reconstruction was tried with 8 to 10 octree levels. It can be seen in Figure 11(a) and 11(b) that the Poisson mesh has improper deformation during animation as the mesh has many irregular folds. Automatic skinning methods like bone heat skinning [Ble13] fail to skin such meshes properly. These problems do not occur in the meshes generated using our method (as can be seen in Figures 11(c) and 11(d)). Animation systems using Poisson meshes have to rely on extensive manual skinning to get them to work properly.

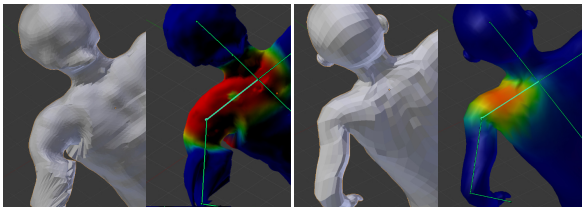


Figure 11: *Difference between animating a (a, b) Poisson reconstructed mesh and (c, d) mesh recovered using our system*

We have created personalized avatars for multiple users as can be seen in Figure 12. We have also animated these

Figure 12: *Models of users 1, 2 and 3 created using our system.*

models using the same motion stream captured from a single Kinect camera using our motion retargeting system as shown in Figure 13. We measured various anthropomorphic dimensions on the created mesh and the actual user. The lengths we

Length	User 1			User 2			User 3			User 4		
	User (cm)	Model (cm)	Error %	User (cm)	Model (cm)	Error %	User (cm)	Model (cm)	Error %	User (cm)	Model (cm)	Error %
L_1	35	33	5.7	40	40	0	37	35	5.4	44	47	6.8
L_2	61	54	11.4	73	76	4.2	70	72	2.7	70	77	10
L_3	22	21	4.6	22	23	4.5	24	23	4.1	28	28	0
L_4	42	42	0.0	46	47	2.1	42	43	2.3	44	46	4.5
L_5	49	50	2.0	55	60	9	56	55	1.7	60	59	1.66
L_6	37	38	2.7	48	46	4.1	47	45	4.2	50	48	4

Table 1: *Measurements of lengths on the actual user, their model reconstructed using our system and the percentage error between the two measurements.*

Figure 13: *Frames from an animation for models of users 1, 2 and 3. The motion was captured and applied using our single Kinect motion capture and retargeting system.*

measured are shown in Figure 14. It can be seen from the results given Table 1 that our system produces models with accurate dimensions. This makes the models suitable not only for animations, but also for other augmented and mixed reality applications like virtual try-on of simulated garments. We present this data in Table 1 as validation of the accuracy

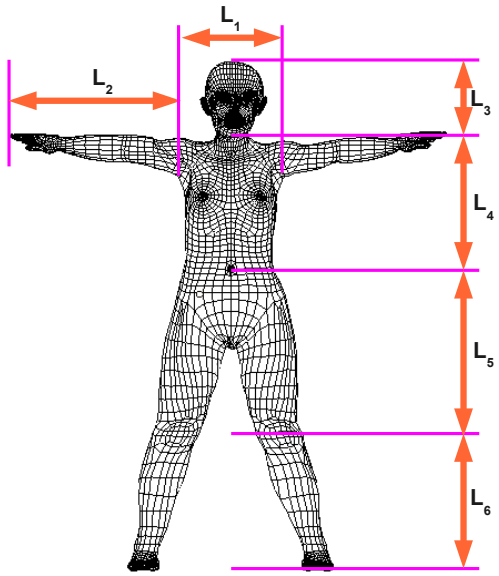


Figure 14: *Lengths measured on the mesh and user for validation.*

of our modeling process.

For a typical user, on a Intel Core i7 machine with 8Gb RAM, stage 1 meshing takes 18 seconds, stage 2 meshing takes 114 seconds, texturing takes 8 seconds. All the user interaction takes about 10 to 15 minutes to complete. Motion capture works in real time. The point cloud capture from the 4-Kinect system and all associated processing to get the point cloud takes about 20 minutes. It should be noted that no special GPU techniques are being used to accelerate the process yet. Such methods are obvious candidates for future improvements.

9. Conclusions

We have presented a system for creating personalized user avatars from depth data. In contrast to the methods in exist-

ing state of the art, the mesh model of the avatar maintains its topology throughout the process, thus making it suitable for animation. This is achieved by fitting a single template mesh having the desired topology to the captured point cloud of the user. The fitting ensures that mesh resembles the user in shape. We perform texture mapping of the body and face to make the model resemble the user in appearance. We validate our claims by providing comparisons anthropomorphic measurements on the created model and the real user. We also show that the obtained model can be readily animated using a single depth camera.

One of the main limitations of the system is that it still requires some user interaction for specifying a few correspondences. We would like to eliminate this and make the process fully automatic. As future work, we would also like to make our process near real-time and more robust to noise.

10. Acknowledgements

We would like to thank the MakeHuman [Mak13] project for the human template models and the Blender Foundation for the open source Blender [Ble13] 3D content creation software. This research was supported by the Immersive Digital Heritage project (NRDMS/11/1586/2009) under the Digital Hampi initiative of the Department of Science and Technology, Government of India.

References

[ADAT*05] AHMED N., DE AGUIAR E., THEOBALT C., MAGNOR M., SEIDEL H.-P.: Automatic generation of personalized human avatars from multi-view video. In *Proceedings of the ACM symposium on Virtual reality software and technology* (2005), pp. 257–260. 2

[ASK*05] ANGUELOV D., SRINIVASAN P., KOLLER D., THRUN S., RODGERS J., DAVIS J.: Scape: shape completion and animation of people. *ACM Transactions on Graphics 24*, 3 (July 2005), 408–416. 2

[BA83] BURT P. J., ADELSON E. H.: A multiresolution spline with application to image mosaics. *ACM Transactions on Graphics 2*, 4 (Oct. 1983), 217–236. 5

[Ble13] BLENDER:. http://www.blender.org, June 2013. 6, 8

[BM92] BESL P. J., MCKAY N. D.: A method for registration of 3-d shapes. *IEEE Transactions on Pattern Analysis Machine Intelligence 14*, 2 (Feb. 1992), 239–256. 1, 3

[BP07] BARAN I., POPOVIC J.: Automatic rigging and animation of 3d characters. *ACM Transactions on Graphics 26*, 3 (July 2007). 6

[CCNS12] CUI Y., CHANG W., NÃűLL T., STRICKER D.: Kinectavatar: Fully automatic body capture using a single kinect. In *ACCV Workshop on Color Depth Fusion in Computer Vision 2012* (2012). 2

[CZ11] CHANG W., ZWICKER M.: Global registration of dynamic range scans for articulated model reconstruction. *ACM Transactions on Graphics 30*, 3 (May 2011), 26:1–26:15. 1, 3

[DAST*08] DE AGUIAR E., STOLL C., THEOBALT C., AHMED N., SEIDEL H.-P., THRUN S.: Performance capture from sparse multi-view video. In *ACM Transactions on Graphics* (2008), vol. 27, p. 98. 2

[GSDA*09] GALL J., STOLL C., DE AGUIAR E., THEOBALT C., ROSENHAHN B., SEIDEL H.-P.: Motion capture using joint skeleton tracking and surface estimation. In *IEEE Conference on Computer Vision and Pattern Recognition, 2009. CVPR 2009* (2009), pp. 1746–1753. 2

[JK02] JEONG W.-K., KIM C.-H.: Direct reconstruction of a displaced subdivision surface from unorganized points. *Graphical Models 64*, 2 (Mar. 2002), 78–93. 2

[KBH06] KAZHDAN M., BOLITHO M., HOPPE H.: Poisson surface reconstruction. In *Proceedings of the fourth Eurographics symposium on Geometry processing* (2006), SGP '06, pp. 61–70. 3

[KCKC04] KOO B. K., CHU C. W., KIM J. C., CHOI Y. K.: Srink-wrapped boundary face algorithm for surface reconstruction from unorganized 3d points. In *Proceedings of the 4th WSEAS International Conference on Signal Processing, Computational Geometry & Artificial Vision* (2004), ISCGAV'04, pp. 17:1–17:5. 2

[Kin13] KINECT M.:. http://www.xbox.com/en-US/kinect, Feb. 2013. 1

[LMT98] LEE W.-S., MAGNENAT-THALMANN N.: Head modeling from pictures and morphing in 3d with image metamorphosis based on triangulation. In *Proceedings of the International Workshop on Modelling and Motion Capture Techniques for Virtual Environments* (1998), CAPTECH '98, pp. 254–267. 5

[Mak13] MAKEHUMAN:. http://www.makehuman.org/, May 2013. 2, 8

[NIH*11] NEWCOMBE R. A., IZADI S., HILLIGES O., MOLYNEAUX D., KIM D., DAVISON A. J., KOHLI P., SHOTTON J., HODGES S., FITZGIBBON A.: Kinectfusion: Real-time dense surface mapping and tracking. In *Proceedings of the 2011 10th IEEE International Symposium on Mixed and Augmented Reality* (2011), ISMAR '11, pp. 127–136. 1

[Ope] OPENNI NITE 2:. http://www.openni.org/files/nite/, Dec. 6

[SCOL*04] 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* (2004), SGP '04. 2, 4

[SKR*06] STOLL C., KARNI Z., RÃűSSL C., YAMAUCHI H., SEIDEL H.-P.: Template deformation for point cloud fitting. In *Proceedings of the 3rd Eurographics / IEEE VGTC conference on Point-Based Graphics* (2006), pp. 27–35. 2, 4

[TZL*12] TONG J., ZHOU J., LIU L., PAN Z., YAN H.: Scanning 3d full human bodies using kinects. *IEEE Transactions on Visualization and Computer Graphics 18*, 4 (2012). 2, 3

[WHB11] WEISS A., HIRSHBERG D., BLACK M. J.: Home 3d body scans from noisy image and range data. In *Proceedings of the 2011 International Conference on Computer Vision* (2011), ICCV '11, pp. 1951–1958. 2

[Wil10] WILLIAMSON J.: Topology in theory and practice - blender conference 2010. In *Proceedings of the Blender Conference* (2010). 1

[Zha00] ZHANG Z.: A flexible new technique for camera calibration. *IEEE Transactions on Pattern Analysis Machine Intelligence 22*, 11 (Nov. 2000), 1330–1334. 2, 3