SkelSeg: Segmentation and Rigging of Raw-Scanned 3D Volume with User-Specified Skeleton

Seung-Tak Noh* The University of Tokyo
Kenichi Takahashi† Kabuku, Inc.
Masahiko Adachi‡ Kabuku, Inc.
Takeo Igarashi§ The University of Tokyo

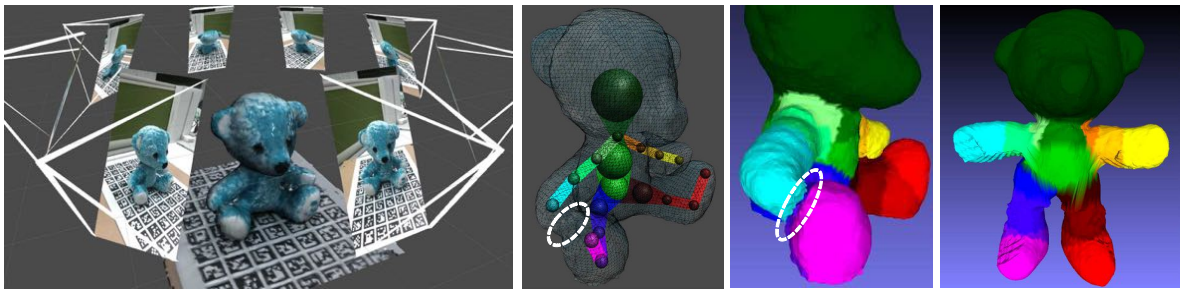


Figure 1: Overview of our approach. (a) Raw-scanned 3D volume with calibrated images. (b) Our tool enables user to annotate 3D skeleton structures in the raw-scanned 3D volume. (c) Our skeleton-based segmentation method enables to clean the geometry issues in the raw-scanned 3D volume. (d) The automatic skinning computation generates the animation-ready 3D skinned mesh.

ABSTRACT

Although RGB-D camera-based scanning has become popular, a raw-scanned 3D model contains several problems that hinder animation such as fused arms and legs. We propose a system that allows a user to generate a rigged 3D mesh from a raw-scanned 3D volume with simple annotations. The user annotates the skeleton structure on the calibrated images captured at the scanning step, and our system automatically segments the raw-scanned volume into parts, generating a skinned 3D mesh based on the user-specified 3D skeleton. We tested our method with several raw-scanned 3D plush toy models, and successfully generated plausible animations.

Keywords: User-specified skeleton, segmentation, rigging

Index Terms: Computing methodologies—Computer graphics—Shape modeling; Computing methodologies—Computer graphics—Animation

1 INTRODUCTION

Three-dimensional scanning method is getting popular as one way to generate 3D models in computer graphics, in addition to polygonal meshing, 3D sculpting (e.g. Zbrush [24]), and sketch-based modeling tool (e.g. Teddy [13]). 3D scanning technologies have originally used by professional to digitize expensive real-world objects as 3D data using highly expensive devices, such as a laser scanner (e.g. [20]). However, recent advances in 3D scanning technology enables a casual user to scan a model using a commodity RGB-D camera [14, 23]. Now that 3D scanning is no longer a special way, we can capture objects in daily life, such as a plush toy, to create 3D character models.

However, a raw-scanned 3D model still has several problems to use it for animation. Ground plane needs to be removed. Invisible areas in scanning sessions generate holes in the scanned model. Nearby parts become fused together. Consequently, intensive editing is required to correct these problems. In addition, one needs to set skeleton and assign bone weights appropriately, which also requires tedious manual operations.

We propose a semi-automatic method for converting a raw-scanned 3D model to an animation-ready model with simple annotations (Fig. 1). The system requires a raw-scanned 3D volume and a few calibrated images as input. Next, the user specifies the predefined skeleton structure and annotates them on the provided calibrated images. Based on the information above, our system segments the raw-scanned volume based on the skeleton and generates a cleaned 3D mesh. Finally, the system automatically computes the bone weights of each vertex and generates a 3D rigged mesh. By using this approach, we can simplify the operations in both geometry cleaning and rigging.

Although there were several methods to clean the geometry and generate animation ready models, we combine these concepts in single system to democratize the scanning for 3D animation. With a small amount of annotation on raw-scanned 3D volume, we clean the ground and fused parts of the scanned object. Most automatic rigging methods assume clean 3D models (e.g. [5]), while we deal with raw-scanned 3D data with dirty geometry.

Our contributions are twofold:

- We present a framework that enables a user to create a rigged 3D character mesh from a raw-scanned 3D model with simple annotations.
- We describe an algorithm to clean a raw-scanned 3D model leveraging user-specified skeleton.

2 RELATED WORK

Since the release of the commodity RGB-D camera [34] and technical advances in scanning methods (e.g. [6, 9, 14, 23]), 3D scanning becomes a popular way to generate 3D models. However, despite of its possibility, scanning method is not so widely used in various context. To democratize this technology, we need to solve common problems in raw-scanned 3D object.

*e-mail: noh@ui.is.s.u-tokyo.ac.jp
†e-mail: kenichi.takahashi@kabuku.co.jp
‡e-mail: masahiko.adachi@kabuku.co.jp
§e-mail: takeo@acm.org

2.1 Segmentation of Scanned 3D Data

One major problem of scanning technology resides in the object segmentation. Since there is no semantic process in typical scanning method [14, 23], target object and surrounding scene are usually not separated.

There have been several ways to segment an object from the scene, but most approaches were done in an automatic way. One representative approach is the plane-based segmentation (e.g. [11]). In this approach, it estimates the main plane(s) in the view and segments each object from the plane. This also has several limitations, so researchers have been started to use deep-learning to segment 3D point cloud (e.g. PointNet [25]). Compared to automatic approach, user-assisted 3D data segmentation has not been widely investigated in a research field. SemanticPaint [32] supported user-assisted segmentation, but it only supported the large-scale object segmentation from a room-scaled 3D scene.

In this work, we assume small scanning volume which contains a single object on a flat surface. We only consider removing the flat surface from the raw-scanned volume. We do not claim a contribution on this point. However, it is a reasonable assumption because additional objects usually bother the scanning process (i.e. partial scanning by unseen areas), and it also enforces to decrease the scanning resolution in each object, consequently.

2.2 Skeleton-based 3D Modeling and Refinement

Another important factor for democratization of scanning method is shape refinement. There are several issues in a raw-scanned 3D shape, but many users still rely on conventional 3D modeling tool which is mainly designed for modeling-from-scratch. Skeleton is a common structure to segment and refine a raw-scanned 3D data. We refer a recent survey [30] for more comprehensive introduction on this approach.

In a skeleton-based shape processing, the way to generate skeleton has an issue. Most previous methods (e.g. [12, 26]) depend on automatic skeleton extraction, but it is difficult to achieve completely automatic approach, so they usually provide user-defined parameters. Morfit system [33] supports generalized cylinder fitting [3, 8] on point cloud data to complement imperfect 3D scanning. The concept of using the skeleton for geometry improvement of raw-scanned data is quite similar to ours. However, we mainly work on separation of existing shape by removing volume.

3D skeleton with thickness is also used in the context of 3D modeling. B-Mesh system [17] utilizes a user-specified 3D skeleton with key-balls. They generate an initial mesh by sweeping and stitching the balls in skeleton and subdivide it to obtain a higher resolution mesh. Sphere-Meshes [31] has the similar concept of sphere-based 3D skeleton, but it adopts simplices among spheres to create a final shape. By this approach, they can represent a complex shape with a small amount of primitive shapes. These works, however, mainly aim at simplification of well-defined 3D meshes and do not care for raw-scanned 3D data.

In this work, we focus on *separating fused parts* in the raw-scanned 3D volume. We use user-specified 3D skeleton, which is originally used for rigging the 3D model, to separate fused parts in a raw-scanned 3D volume. To the best of our knowledge, there is no previous work to jointly solve problems both separating fused parts and rigging the limbs in a raw-scanned 3D model.

2.3 Rigging for Mesh Animation

Mesh skinning is a common approach for 3D animation, but the essential cost on the manual rigging prevents a non-skilled user from using it. Due to this, the support of the mesh rigging has been an important research issue for decades. The most direct and representative way is a fully automatic approach, like Pinocchio system [1]. Recently, Dionne and de Lasa [5] applied voxel-based discretization to overcome the manifoldness in the automatic rigging process. However, this approach is basically limited to the clean 3D model that has clearly separated limbs with a rest pose (i.e. T- or A-pose).

Recent work is rather to support the manual deformation with a novel optimization technique. Jacobson and colleagues [15, 16] introduced methods that allow to deform 2D or 3D mesh with several user-specified control points by minimizing the energy. It allows an intuitive mesh deformation for novices, but it also need an optimization-based deformation framework, which is not widely supported in an off-the-shelf graphics engine. In addition, those methods need an additional discretization such as tetrahedral meshing [29] that supposes a cleaned 3D surface mesh as an input. By these reasons, we keep using bone-based skinning for mesh deformation, while trying to solve the shape issues in an original raw-scanned 3D voxels.

To reduce such an effort in mesh skinning, there is an alternative method that generates a rigged mesh from scratch. Borosan et al. [2] proposed the RigMesh system, that enables user to create the skinned mesh. This system was also based on the generalized cylinder as a primitive [3, 8]. Recently, Jin et al. proposed AniMesh [18] enabling users to animate 3D models from the RigMesh system [2] by human motion. These approaches, however, have the same shape limitations inherited from the geometric primitives. Our work is inspired by them, but we endeavor to keep using raw-scanned 3D data.

3 Motivation

Before introducing our system, we illustrate the problematic areas in a raw-scanned 3D model captured with a commodity RGB-D camera (Fig. 2). We address the three types of problems in this work: ground plane, internal artifacts, and fused parts.

Ground plane We assume that the target object is placed on a flat ground plane (Fig. 2a). Such a ground plane is included in the raw-scanned volume, so it is necessary to remove before dealing with it as a 3D object.

Internal artifacts Voxel-based scanning methods such as KinectFusion [14, 23] integrate each depth frame into a 3D regular grid and construct a *truncated* signed distance field. It effectively cancels the depth noise in the measurement, but the integration usually causes the artifacts inside of the 3D model (Fig. 2b).

Fused parts Different limb parts, such as an arm and leg, are fused together in the volumetric scan when they touch each other (Fig. 2c). Therefore, it is necessary to separate them.

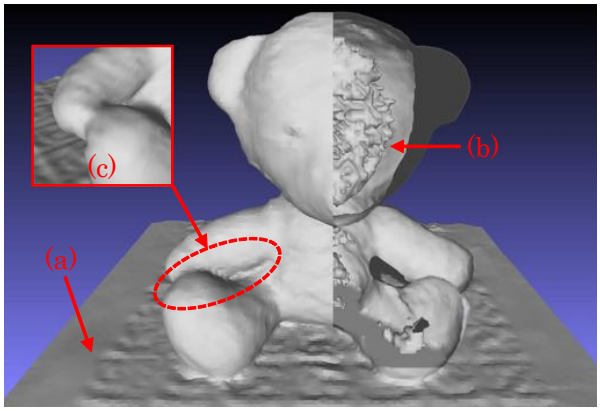


Figure 2: Problematic shapes in a raw-scanned 3D model. (a) Ground plane (b) Internal artifact (c) Fused parts.

These are quite common problems in a raw-scanned 3D model, but there is no general solution yet. Conventional 3D sculpting software, such as Zbrush [24], is used to address these issues in

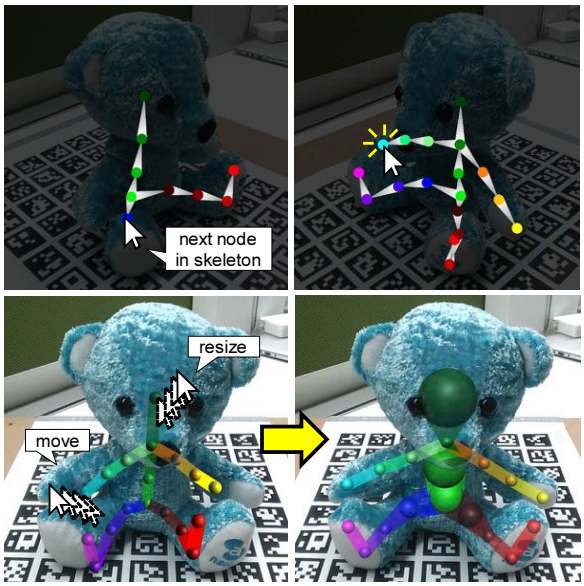


Figure 3: Annotation tool for the skeleton specification. (top) User can put the 2D projected points of 3D node in each reference view. (bottom) Based on 2D points in multiple views, the system computes initial 3D nodes. User can check and adjust the position and size of 3D nodes.

practice, but this kind of tool and operation are difficult and time-consuming for novice users. Instead, we fix these problems by asking the user to give simple annotations.

4 User Interface

4.1 Workflow

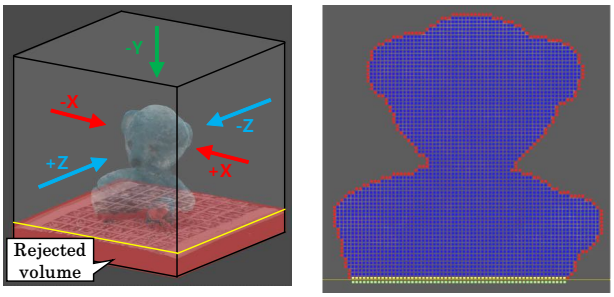
Our system consists of three parts: RGB-D based scanning, user-annotation, and geometry processing (Fig. 1). We built our scanning platform with an Intel® RealSense™ SR300 near-range depth camera and the KinectFusion algorithm [14, 23] for depth integration. Although KinectFusion supports camera tracking by using the reconstructed 3D volume, our current implementation uses more stable and accurate marker-based camera pose estimation [27]. Our scanning platform also captures the color views specified by the user.

Once the scanned 3D volume with color images are acquired, the user switches to the annotation tool. The captured color images are associated with their corresponding camera poses. Because a novice user has a difficulty in 3D rotation [8], we do not include such an operation in our system. Instead, we provide registered multi-view images and let user do the 2D-based operations to handle the annotations. The number of images and its distribution are not limited, but we suppose less than 20 side views for the user interface.

With the registered multi-view references on the raw-scanned volume, the user first annotates the ground plane points. This step seems unnecessary because we are using fiducial marker in our current implementation. However, it is a tentative feature for prototype, and we cannot expect that information in a casual scanning situation. Hence, we do not rely on the marker information as possible. Removing the ground is done, then the user annotates the skeleton structure in raw-scanned volume. After setting the skeleton, the system runs an automatic geometry processing algorithm to generate a skinned mesh from the raw-scanned 3D volume.

4.2 Annotation Tool

We implemented a simple annotation tool for the user to provide the essential information to clean and rig the raw-scanned 3D volume.



(a) (b)

Figure 4: Removal of ground plane and internal artifacts. (a) We remove the ground plane from user-specified points. (b) From the visibility test in five directions, we acquire the inside (blue) and exterior (red) voxels. Based on this information provided by the ground equation, we fill the hole at the bottom (green).

The process is largely divided into two parts: ground plane specification and skeleton structure annotation. The first part allows the user to specify the points on the ground plane. Clicked points are projected into a 3D domain and used for ground plane estimation.

The second part is skeleton structure annotation (Fig. 3). This part has several steps. First, user chooses the skeleton structure. We currently provide humanoid and quadruped types (Fig. 5), which are compatible with a common motion database such as [4]. Next, our system moves to the 2D annotation step (Fig. 3a). Our system currently requests the user to specify skeletons on two views to get 3D skeleton. After the annotations at two-views are done, we compute the 3D position of each node with epipolar geometry [10]. Once the initial 3D points are computed, our system then moves to the adjustment step (Fig. 3b). The user can check and adjust the position and size of each 3D node with simple drag-and-drop operations. More advanced technology such as inferring 3D skeleton from multi-view 2D skeletons [19] is also available, but our raw-scanned 3D volume may have fused parts which make skeleton extraction difficult. We believe that editing the predefined 3D skeleton with about 20 bones is not so tedious for a user, so we completely rely on the 3D skeleton specified by user.

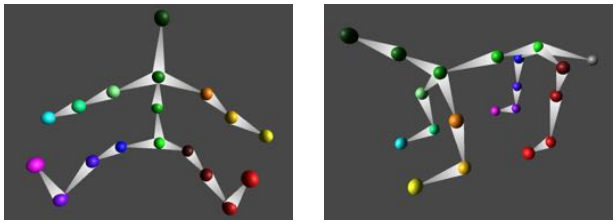
5 Algorithm

Based on the user-specified skeleton given in the annotation tool, our system generates a cleaned mesh from the 3D volume and computes its skinning. Our geometry processing is mainly inspired by Dionne and de Lasa [5]. In their method, they first voxelize the character's 3D mesh before computing the vertex to bone distances. Our scanned 3D data is also contained in 3D voxel space, so we adopt their main workflow. However, the dirty geometry in a raw-scanned 3D data prevent us from using this method directly. We therefore added extra steps to clean the dirty geometry in raw-scanned 3D volume. Our geometry processing consists of three parts: removal of ground plane and internal artifacts from the volume, separation of fused parts, and mesh skinning.

5.1 Removal of Ground Plane and Internal Artifacts

We first separate the ground plane from the user-specified 3D points. Due to the lack of semantics, the ground plane is mixed with the object in the raw-scanned volume. Using the 3D ground plane points identified by users, we compute the equation of the ground plane. We then remove all the voxel under (Fig. 4a).

Next, we remove the internal artifacts in the raw-scanned volume by modifying the visibility checking approach in Dionne and de Lasa [5]. Fig. 4 illustrates our filtering process. We check the visibility of each voxel from five orthogonal view directions (we



(a) (b)

Figure 5: Predefined skeleton set in our prototype. (a) Humanoid and (b) Quadruped model.

do not consider bottom-to-up direction because there is a hole). The internal and exterior voxels can be determined based on these visibilities. After we conclude that the visible voxels are exterior and invisible voxels are interiors, we can then remove all the internal voxels.

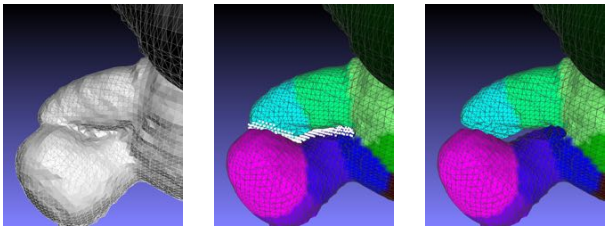
Finally, we fill the hole at the bottom of the object caused by the removal of the ground plane (Fig. 4b). We check the right above layers on the rejected volume, which is specified by the ground plane equation. If it is internal, then we attach the exterior under this voxel. Using this approach, we can acquire the water-tight 3D volume without the internal artifacts.

5.2 Separation of Fused Parts

To leverage the user-specified skeleton to separate fused parts (e.g. Fig. 2c), we first compute the distance fields from each skeleton bone to each internal voxels, like [5]. We then associate each voxel with the nearest bone. For each pair of adjacent voxels, we check the precomputed graph geodesic in the skeleton (Fig. 5) between the associated bones. To compute this geodesic, we consider each bone as a graph vertex and each node (bone connection) as a graph edge. If the graph geodesic is smaller than the threshold (we used 3), then we keep them connected. If the geodesic is larger, then we separate (disconnect) the two voxels. Since the raw-scanned volume contain truncated signed distances, the result mesh keeps the water-tightness whether voxels are separated or not. Finally, we obtain the surface mesh by applying the iso-surface extraction method [7, 21]. The associated bone is visualized with color on this extracted mesh, so the user can easily modify the skeleton (Fig. 6).

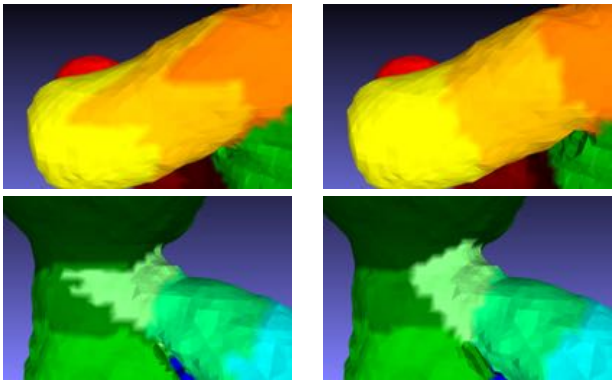
In a computation of distance fields, we apply the fast marching method [28] (FMM) instead of the Dijkstra algorithm [5]. The main reason is the difference in voxel resolution. We directly employ the raw-scanned 3D volume in the voxelization, so our resolution is coarser than Dionne and de Lasa's. In this situation, the distance field calculated with the Dijkstra method suffers from artifacts by Manhattan distance in the parts separation and mesh skinning (Fig. 7). By substituting the metric computation with FMM, we could simply solve this issue.

Compared to standard human models, our target models (plush toys) have characteristic shapes (e.g. large head). For this reason, the naïve distance computation does not work well without considering the bone volume (Fig. 10 and 11). To overcome this issue, we allow the user to specify node size in the annotation tool. Bone shape is also considered as a capsule-like 3D volume. Specifically, this volume is a convex hull of two spheres those are possibly having different radii. We describe how to compute this in Appendix A. After determining the voxels which are included in the node, we place these voxels as the initial seeds which have 0-distance, and then run the FMM to compute the distance field. The computation of the entire distance fields consumes a lot of time, so we only update the distance fields of modified bones in the recomputation step. We also interactively visualize the intermediate results after each computation for a distance field.



(a) (b) (c)

Figure 6: Our separation method for fused parts. Colors are associated to bone. (a) Mesh from raw-scanned volume. Near parts are wrongly fused together. (b) We identify the wrong connection in the voxel domain (white dots denote the voxel points). (c) Mesh after the separation.



(a) (b)

Figure 7: The impact of metric differences. (a) Dijkstra-based metric causes the discrete artifacts in the final shape due to the Manhattan distance. (b) Our FMM-based metric generates relatively smoother result, in spite of the coarse voxel resolution.

5.3 Mesh Skinning

For the last step, we compute the skinning weights for the 3D mesh. We apply the automatic skinning computation by Dionne and de Lasa [5] for vertex weights. Since we already acquired the cleaned 3D mesh and distance fields associated with the bones, we can reuse them in our computation as much as possible. However, distance

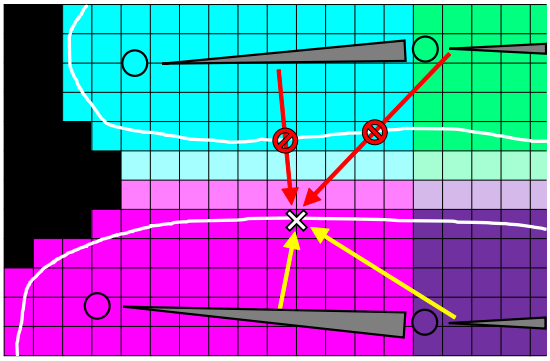


Figure 8: Illustration of the inconsistent distance values. Although we disposed the voxels in fused parts (white-toned), the wrongly propagated distances still remain (red arrows). To avoid recomputation of distance fields, we only bind the associated and adjacent bones to the vertex (yellow arrows).

fields may have incorrect distances (Fig. 8) because the voxel separation is determined *after* computing the distance fields. Naïve approach to solve this issue is recomputing the distance except for disconnected voxels, but it is a time-consuming process that is not desirable for the interactive system. To avoid this without recomputation, we only bind the associated and its adjacent bones in each vertex.

6 Results

6.1 Quality and Performance

We tested our method with 10 plush toy models. All of them were scanned in 128^3 voxel resolution, and each length of axis is 25 [cm]. Eight of them were human-like biped model, though the proportions were quite different. The remaining two were horse-like quadruped models. Fig. 10 and 11 shows our skeleton annotation, parts separation and skinning results.

We measured the performance of our current implementation: a desktop computer consisting of an Intel® Core™ i7-7700K CPU with 16GB RAM. Table 1 shows our experiment results. Most computation time was spent on the distance fields computation, because they needed to compute a 3D distance field per bone. Note that the total time refers the worst-case situation, which computes all distance fields. In a casual usage, 2-5 bones were recomputed, so the wait time usually took 10-20 seconds, except of the initial computation.

6.2 Comparison

We compared our animation result with the Pinocchio system [1]. However, since Pinocchio had no mesh cleaning process, simply applying the raw-scanned 3D model could not generate a meaningful result. For this reason, we fed the cleaned mesh from our separation method and switched the mesh skinning to Pinocchio.

Fig. 9 shows a screenshot of the mesh animation generated by Pinocchio and us. In case of Pinocchio, it initially assumed the A- or T-pose, so the skinned results were not correct. In our method, however, there is no such an issue because we can support non-standard initial pose from the user specified skeleton. In other words, it implies that our system can cover a wider range of 3D character models than Pinocchio.

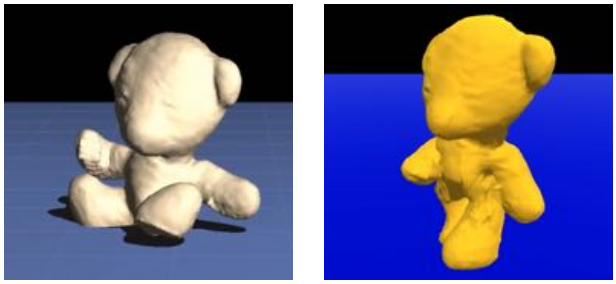


Figure 9: The screenshot of animation by Pinocchio [1] and ours. Because Pinocchio assumes A- or T-pose at the initial status, it occurred false rigging (left). In contrast, we can support the arbitrary initial pose from the user-specified skeleton (right).

6.3 Pilot Study

We conducted a pilot user study to evaluate our annotation system with the separation algorithm. We scanned 3D models shown in Fig. 10 in advance and captured equally-distributed side views (e.g. Fig. 1a) per 3D model. After the scanning session, we invited five users who had a knowledge on 3D computer graphics. One of them was an expert and remaining four had an intermediate knowledge on 3D computer graphics. The session took 10-15 minutes per

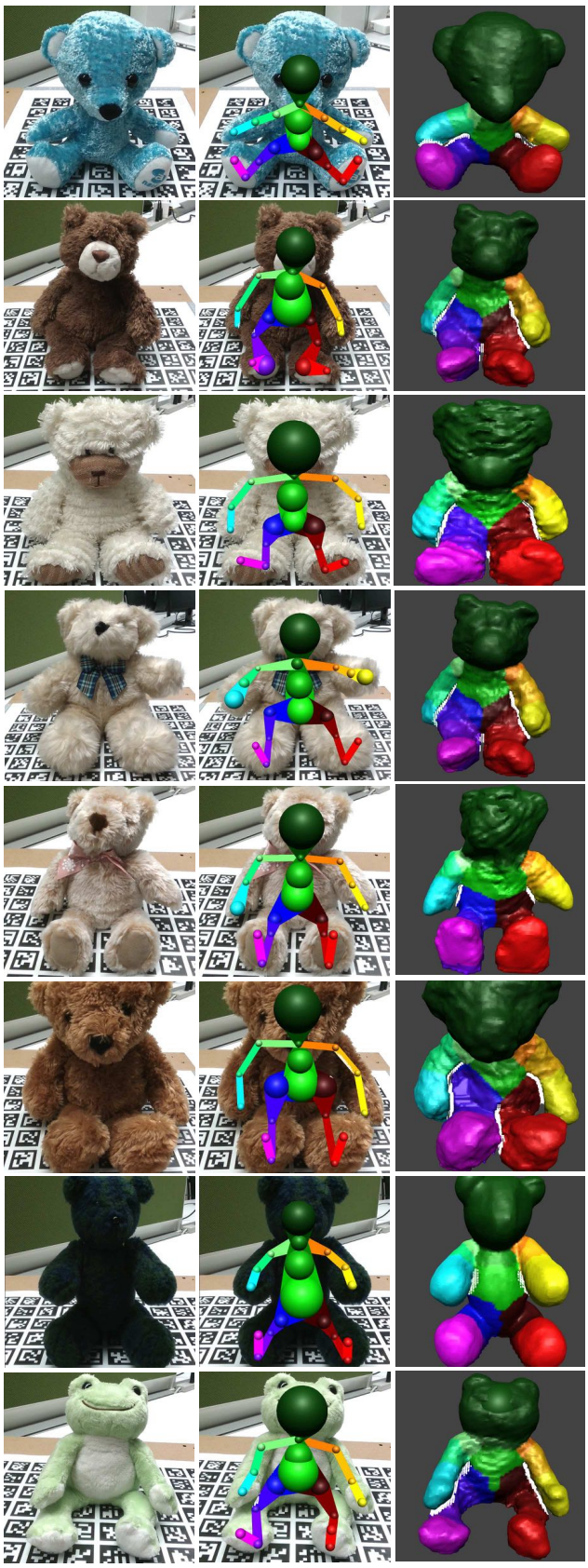


Figure 10: The results of humanoid shape. (left) Frontal image (middle) User-specified skeleton (right) Mesh and removed voxels.

Table 1: Our experiment result. Timing is seconds.

Model name	Model type (#nodes)	Artifacts removal	Distance comp.	Voxel segment.	Mesh skinning	Total time	#valid voxels in the raw-scan	#vertices in the final mesh
Bear_blue	Humanoid(18)	2.371	24.375	5.254	3.876	31.371	86,508	20,513
Bear_brown	Humanoid(18)	2.361	35.075	5.687	9.856	52.498	154,306	33,297
Bear_white1	Humanoid(18)	2.386	32.405	5.690	8.312	48.459	138,517	30,434
Bear_white2	Humanoid(18)	2.427	53.363	6.364	17.841	78.962	264,961	47,778
Bear_beige	Humanoid(18)	2.358	28.347	5.602	6.209	41.823	112,096	26,197
Bear_orange	Humanoid(18)	2.358	45.340	6.261	14.672	67.817	220,157	43,274
Bear_green	Humanoid(18)	2.350	25.986	5.465	4.784	38.226	98,052	22,943
Frog	Humanoid(18)	2.373	21.838	5.165	3.353	32.226	69,425	19,603
Skunk	Quadruped(20)	2.409	31.219	5.457	5.439	43.932	106,620	46,366
Cat	Quadruped(20)	2.372	21.434	5.340	2.260	31.257	52,443	15,182

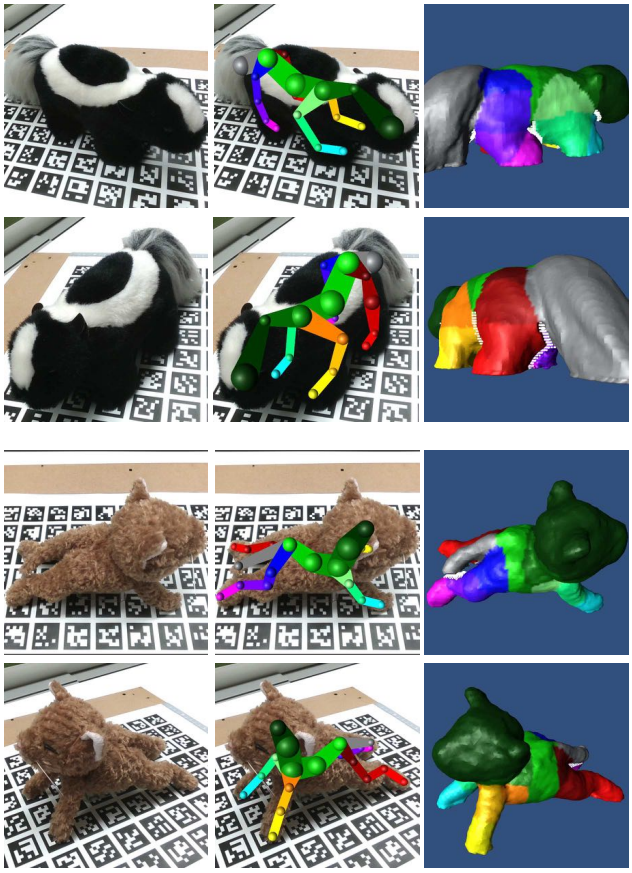


Figure 11: The results of quadruped shape. We provide two different views for each model. Columns are consistent with Fig. 10.

model. All participants succeeded in generating similar positioning of skeleton to that of Fig. 10. However, the size of nodes varied to the participant, so partitioning results also varied.

In the post study interviews, they mentioned that they did not have difficulty understanding the concept or the manual operations of our annotation system. Two participants mentioned on the unique experience of our 2D-based interface and wanted to use a 3D-based interface, but they also adapted to using our system in a short time. However, all participants mentioned that it is difficult to anticipate the actual impact of the size of nodes in the final mesh. Because of this, they needed to adjust the node size to recompute the distance fields several times. We thought that the wait time after each operation might cause frustrations, but no participants complained about it owing to our interactive visualization. However, they also implied that it will be obtrusive if they need to work on the task for a long period. Additionally, two of them said that they would like to know the impact of their operations on animated 3D mesh, which is not included in our current implementation.

7 Discussion

Our method successfully cleaned and created a skinned 3D mesh from the raw-scanned 3D volume with simple manual annotations. However, we still have several issues to be solved. The most pressing problem is the shape completion at the invisible parts. For instance, we simply created the flat surface on the ground where is invisible at the scanning session. It made generating the water-tight 3D shape possible. However, the generated shape was not so natural. Similarly, we also have a shape problem in the separated parts described in Sect. 5.2. Although our voxel resolution is adequate so there is no critical issue yet, the shape in the originally fused parts was not so natural, neither. However, we clearly identify disconnected voxels in our framework, so it will be also possible to improve these areas. The possible solution to this problem is to utilize the skeleton's shape. Based on the size of the bones, we can create a plausible 3D shape at the invisible parts. Generalized cylinder based modeling and fitting method [3, 8] might be helpful for this process. Additionally, mesh subdivision or remeshing at those area is also helpful for smoother results.

Another significant drawback is the mesh quality in the animation. In our animation results, we can still find visual artifacts not only the cleaned parts mentioned above but also the other parts. It comes from various reasons, but the main reason is the low mesh resolution. Mesh subdivision at those area or remeshing will be helpful for resolving these artifacts. Combining those mesh operations with user interface will be necessary for generating more natural animation.

In this paper, we only evaluated our system with a small number of participants. We need a more intensive study to evaluate our annotation system. Prior to that, we need to improve our prototype system to provide a smoother user experience. The main problem of our current prototype is the lack of the rapid user feedback. For the functionalities mentioned in our pilot study, such as the quick voxel visualization and mesh animation, we need a fast computation of the distance field. To achieve this, computing distance field transform in linear time [22] instead of geodesics-based measurement [5, 28] might be helpful. Because we dispose computed distances due to the wrong propagation for parts separation, such a heavy computation for this will not be necessary. After removing fused parts in a raw-scanned 3D volume by distance transforms, recomputing voxel-based geodesics for mesh rigging can provide more acceptable experience to user.

We also need to support variable skeleton structures. The two representative skeleton structures we possessed (Fig. 5) were not enough to handle the 3D shapes in a real-world. For example, we

cannot handle a fish-like shape in our implementation. To solve this issue, we allow users to create novel skeleton structures. An intuitive and easy-to-use tool is needed to achieve this goal.

8 Conclusion

We proposed a system allowing users to generate a rigged 3D mesh from a raw-scanned 3D volume with simple annotations. It asks the user to annotate the skeleton structure on calibrated images captured at the scanning step. Our system then segments the raw-scanned volume and generates a skinned 3D mesh based on the user-specified 3D skeleton. We tested our system with 10 raw-scanned 3D plush toy models, and succeeded generating clean, skinned 3D meshes and animating them. However, we still need improvements on the final shape, especially on the separated limbs where were originally fused parts.

A Convex Hull of Two Spheres

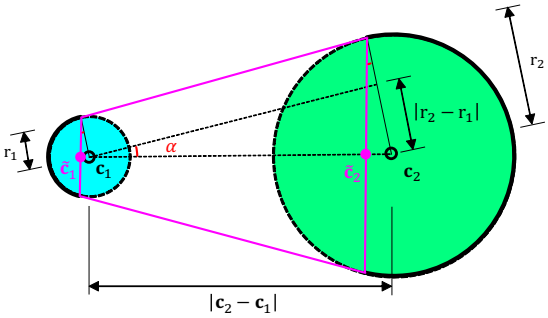


Figure 12: Our bone shape as a convex hull of two spheres.

Here we describe how to compute the convex hull of our bone shape (Fig. 12). Since we have two spheres with different radii in our convex hull, the main issue is basically computing the center position and radius at the cut surface (trapezoid in Fig. 12). We compute the angle α by:

$$\sin\alpha = \left(\frac{r_2 - r_1}{|\mathbf{c}_2 - \mathbf{c}_1|}\right) \tag{1}$$

Note that a numerator becomes a negative value when $r_1 > r_2$, but it still works. By using this value, we can compute the center of a cut surface $\tilde{\mathbf{c}}_i$ and its radius $\tilde{r}_i$, respectively:

$$\tilde{\mathbf{c}}_i = \mathbf{c}_i - \hat{\mathbf{v}} \cdot r_i \sin\alpha \tag{2}$$

$$\tilde{r}_i = r_i \cos\alpha \tag{3}$$

Algorithm 1 Determine a point in a bone

```
if |p − c_i| < r_i where i = 1,2 then          ▷ 1) point to spheres
    return inside
else
    ṽ = c̃_2 − c̃_1, w = p − c̃_1
    t_1 = |ṽ · w|, t_2 = |ṽ · ṽ|
    if t_1 < 0.0 or t_2 < t_1 then             ▷ 2) check ortho projection
        return outside
    else
        t = t_1/t_2, p̃ = c̃_1 + t · ṽ
        dist = |p − p̃|
        size = (1.0 − t) · r̃_1 + t · r̃_2
        if dist ≤ size then                    ▷ 3) distance to line segment
            return inside
return outside
```

where $\hat{\mathbf{v}}$ denotes the normalized directional vector $\frac{\mathbf{c}_2 - \mathbf{c}_1}{|\mathbf{c}_2 - \mathbf{c}_1|}$. Based on the relationship above, we compute Algorithm 1 to determine whether the voxel point is inside of this convex hull.

Acknowledgments

This work was partially supported by JSPS KAKENHI Grant Number JP17H00752.

References

[1] I. Baran and J. Popović. Automatic rigging and animation of 3d characters. *ACM Trans. Graph.*, 26(3), July 2007.

[2] P. Borosán, M. Jin, D. DeCarlo, Y. Gingold, and A. Nealen. Rigmesh: Automatic rigging for part-based shape modeling and deformation. *ACM Trans. Graph.*, 31(6):198:1–198:9, Nov. 2012.

[3] T. Chen, Z. Zhu, A. Shamir, S.-M. Hu, and D. Cohen-Or. 3sweepp: Extracting editable objects from a single photo. *ACM Trans. Graph.*, 32(6):195:1–195:10, Nov. 2013.

[4] CMU. CMU graphics lab motion capture database. `http://mocap.cs.cmu.edu/`.

[5] O. Dionne and M. de Lasa. Geodesic voxel binding for production character meshes. In *Proceedings of the 12th ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, SCA '13, pages 173–180, New York, NY, USA, 2013. ACM.

[6] M. Dou, J. Taylor, H. Fuchs, A. Fitzgibbon, and S. Izadi. 3d scanning deformable objects with a single rgbd sensor. In *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, volume 00, pages 493–501, June 2015.

[7] S. Fuhrmann, M. Kazhdan, and M. Goesele. Accurate isosurface interpolation with hermite data. In *2015 International Conference on 3D Vision (3DV)*, volume 00, pages 256–263, Oct. 2015.

[8] Y. Gingold, T. Igarashi, and D. Zorin. Structured annotations for 2d-to-3d modeling. *ACM Trans. Graph.*, 28(5):148:1–148:9, Dec. 2009.

[9] K. Guo, F. Xu, T. Yu, X. Liu, Q. Dai, and Y. Liu. Real-time geometry, albedo, and motion reconstruction using a single rgb-d camera. *ACM Trans. Graph.*, 36(4), June 2017.

[10] R. Hartley and A. Zisserman. *Multiple View Geometry in Computer Vision*. Cambridge University Press, New York, NY, USA, 2 edition, 2003.

[11] D. Holz, S. Holzer, R. B. Rusu, and S. Behnke. Real-time plane segmentation using rgb-d cameras. In *RoboCup 2011: Robot Soccer World Cup XV*, pages 306–317, Berlin, Heidelberg, 2012. Springer Berlin Heidelberg.

[12] H. Huang, S. Wu, D. Cohen-Or, M. Gong, H. Zhang, G. Li, and B. Chen. L1-medial skeleton of point cloud. *ACM Trans. Graph.*, 32(4):65:1–65:8, July 2013.

[13] T. Igarashi, S. Matsuoka, and H. Tanaka. Teddy: A sketching interface for 3d freeform design. In *Proceedings of the 26th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '99, pages 409–416, New York, NY, USA, 1999. ACM Press/Addison-Wesley Publishing Co.

[14] S. Izadi, D. Kim, O. Hilliges, D. Molyneaux, R. Newcombe, P. Kohli, J. Shotton, S. Hodges, D. Freeman, A. Davison, and A. Fitzgibbon. Kinectfusion: Real-time 3d reconstruction and interaction using a moving depth camera. In *Proceedings of the 24th Annual ACM Symposium on User Interface Software and Technology*, UIST '11, pages 559–568, New York, NY, USA, 2011. ACM.

[15] A. Jacobson, I. Baran, L. Kavan, J. Popović, and O. Sorkine. Fast automatic skinning transformations. *ACM Trans. Graph.*, 31(4):77:1–77:10, July 2012.

[16] A. Jacobson, I. Baran, J. Popović, and O. Sorkine. Bounded biharmonic weights for real-time deformation. *ACM Trans. Graph.*, 30(4):78:1–78:8, July 2011.

[17] Z. Ji, L. Liu, and Y. Wang. B-mesh: A modeling system for base meshes of 3d articulated shapes. *Computer Graphics Forum*, 29(7):2169–2177, 2010.

[18] M. Jin, D. Gopstein, Y. Gingold, and A. Nealen. Animesh: Interleaved animation, modeling, and editing. *ACM Trans. Graph.*, 34(6):207:1–207:8, Oct. 2015.

[19] J. Kustra, A. Jalba, and A. Telea. Probabilistic view-based 3d curve skeleton computation on the gpu. In *VISAPP 2013 : Proceedings of the International Conference on Computer Vision Theory and Applications, Barcelona, Spain, February 21-24, 2013. Volume 2*, pages 237–246. INSTICC Press, 2013.

[20] M. Levoy, K. Pulli, B. Curless, S. Rusinkiewicz, D. Koller, L. Pereira, M. Ginzton, S. Anderson, J. Davis, J. Ginsberg, J. Shade, and D. Fulk. The digital michelangelo project: 3d scanning of large statues. In *Proceedings of the 27th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '00, pages 131–144, New York, NY, USA, 2000. ACM Press/Addison-Wesley Publishing Co.

[21] W. E. Lorensen and H. E. Cline. Marching cubes: A high resolution 3d surface construction algorithm. *SIGGRAPH Comput. Graph.*, 21(4):163–169, Aug. 1987.

[22] A. Meijster, J. B. T. M. Roerdink, and W. H. Hesselink. *A General Algorithm for Computing Distance Transforms in Linear Time*, pages 331–340. Springer US, Boston, MA, 2000.

[23] R. A. Newcombe, S. Izadi, O. Hilliges, D. Molyneaux, D. Kim, A. J. Davison, P. Kohli, J. Shotton, S. Hodges, and A. Fitzgibbon. Kinectfusion: Real-time dense surface mapping and tracking. In *Proceedings of the 2011 10th IEEE International Symposium on Mixed and Augmented Reality*, ISMAR '11, pages 127–136, Washington, DC, USA, 2011. IEEE Computer Society.

[24] Pixologic Inc. Zbrush. `http://pixologic.com/zbrush`.

[25] C. R. Qi, H. Su, K. Mo, and L. J. Guibas. Pointnet: Deep learning on point sets for 3d classification and segmentation. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, July 2017.

[26] D. Reniers and A. Telea. Skeleton-based hierarchical shape segmentation. In *IEEE International Conference on Shape Modeling and Applications 2007 (SMI '07)*, pages 179–188, June 2007.

[27] F. J. Romero-Ramirez, R. Muñoz-Salinas, and R. Medina-Carnicer. Speeded up detection of squared fiducial markers. *Image and Vision Computing*, 76:38 – 47, 2018.

[28] J. A. Sethian. A fast marching level set method for monotonically advancing fronts. In *PROC. NAT. ACAD. SCI*, pages 1591–1595, 1995.

[29] H. Si. Tetgen, a delaunay-based quality tetrahedral mesh generator. *ACM Trans. Math. Softw.*, 41(2):11:1–11:36, Feb. 2015.

[30] A. Tagliasacchi, T. Delame, M. Spagnuolo, N. Amenta, and A. Telea. 3d skeletons: A state-of-the-art report. *Computer Graphics Forum*, 35(2):573–597, 2016.

[31] J.-M. Thiery, É. Guy, and T. Boubekeur. Sphere-meshes: Shape approximation using spherical quadric error metrics. *ACM Trans. Graph.*, 32(6):178:1–178:12, Nov. 2013.

[32] J. Valentin, V. Vineet, M.-M. Cheng, D. Kim, J. Shotton, P. Kohli, M. Nießner, A. Criminisi, S. Izadi, and P. Torr. Semanticpaint: Interactive 3d labeling and learning at your fingertips. *ACM Trans. Graph.*, 34(5):154:1–154:17, Nov. 2015.

[33] K. Yin, H. Huang, H. Zhang, M. Gong, D. Cohen-Or, and B. Chen. Morfit: Interactive surface reconstruction from incomplete point clouds with curve-driven topology and geometry control. *ACM Trans. Graph.*, 33(6):202:1–202:12, Nov. 2014.

[34] Z. Zhang. Microsoft kinect sensor and its effect, Feb 2012.