Auto-rigging 3D Bipedal Characters in Arbitrarily Poses

Jeonghwan Kim, Hyeontae Son, Jinseok Bae, and Young Min Kim

Seoul National University, South Korea

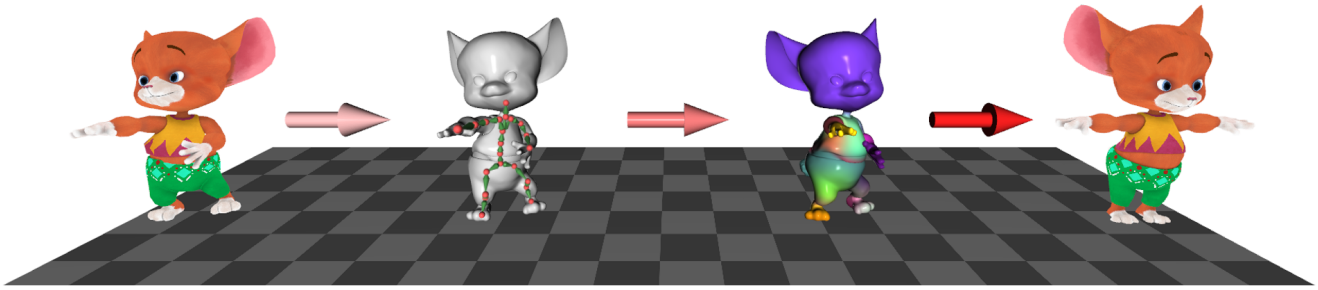

Figure 1: *Auto-rigging a 3D bipedal character in arbitrary pose. Given a 3D character in an arbitrary pose, we find the best embedding of the bipedal skeleton. Using the estimated skin weight, the input mesh is transformed into the canonical pose, from which we can generate 3D animation.*

Abstract

We present an end-to-end algorithm that can automatically rig a given 3D character such that it is ready for 3D animation. The animation of a virtual character requires the skeletal motion defined with bones and joints, and the corresponding deformation of the mesh represented with skin weights. While the conventional animation pipeline requires the initial 3D character to be in the predefined default pose, our pipeline can rig a 3D character in arbitrary pose. We handle the increased ambiguity by fixing the skeletal topology and solving for the full deformation space. After the skeletal positions and orientations are fully discovered, we can deform the provided 3D character into the default pose, from which we can animate the character with the help of recent motion-retargeting techniques. Our results show that we can successfully animate initially deformed characters, which was not possible with previous works.

CCS Concepts

• ***Computing methodologies*** → ***Motion processing; 3D imaging; Neural networks;***

1. Introduction

3D characters used in animations or games are deformed from a canonical pose. The deformation is often defined with a skeleton, which mimics bones and skins of the human body. The movement of the surface mesh of 3D characters is given as a linear combination of transformation parameters of neighboring bones, which follow rigid transform. A typical pipeline starts from a 3D character in a *bind pose*, namely T-pose or A-pose, where the underlying bones and skin weights are manually defined. This technique is successful when the character in the desired pose is available, for example, when they are generated by artists or scanned from a controllable object, for example, a real human body.

However, not all 3D models are available in the bind poses. With the recent advance in 3D modeling and capturing techniques, one can scan and acquire a 3D model of one's favorite figure, for example, a statue bought from a character fair, or one's favorite key chain. To revive the characters in the virtual world, the rigging needs to be extended for the input shape in arbitrary pose.

We propose a method to animate a 3D character in an arbitrary pose. As the input character is not in a canonical pose, we need to solve for the joint locations with the pose parameters. Since the problem is highly under-constrained, we fix the skeleton as one of a bipedal character that resembles human body.

By constraining the skeletal structure, it might appear to be similar to a human-pose estimation problem, which has a large volume

Code available at https://github.com/whitealex95/autorigging-bipedal

DOI: 10.2312/egs.20211023

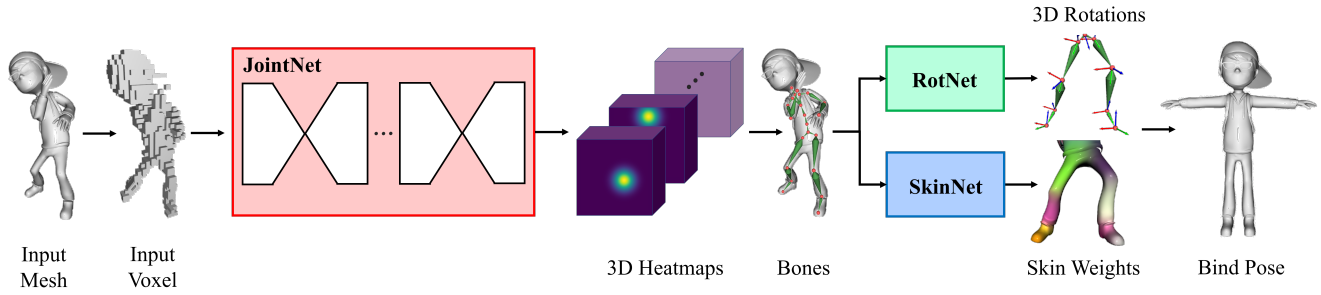

Figure 2: *The overview of our auto-rigging process. We first convert the input mesh into a 3D voxel representation to regress the 3D positions of joints (JointNet). From the joint locations, we find the full orientation of joints (RotNet) and also assign weights corresponding to individual bones (SkinNet). By combining the information, we can animate the input character to the bind pose.*

of literature. Our problem is different from the human-pose estimation in three aspects. First, the relative sizes between different body parts distribute in wide ranges than those of the human body, and the approach for 3D characters require separate training. Second, we need to find the rotations of individual joints, whereas the human pose estimation often only regresses for the joint positions as heat maps and solves for inverse kinematics when necessary. Third, the 3D animation of the character requires deformation of individual mesh vertices with skin weights deduced from the full skeletal transformation [XZK*20].

Our algorithm aims to create lively animation of a character of your choice in arbitrary pose. After we detect the joint locations and solve for initial transformation, we can create a nice animation of unique virtual characters using existing motion re-targeting methods.

2. Method

Given a triangular mesh of a 3D character in a random initial pose, our goal is to find the character in the canonical pose (T-pose or A-pose), and rig a humanoid skeleton to apply bipedal animation. We adapt one of the most widely used skeletal animation formulation, based on linear blend skinning (LBS):

$$v_i^t = \sum_j w_{ij} J_j^t B_j^{-1} v_i, \ \sum_j w_{ij} = 1. \tag{1}$$

where the i-th vertex v_i of the input mesh is transformed to v_i^t with respect to j-th bone transformation matrices $J_j^t B_j^{-1}$ and skin weight w_{ij} . The bone transformation $J_j^t B_j^{-1}$ is composed of two matrices: J_j^t is the usual transform matrix for animation frame at time t, and B_j, which we call as the *bind-pose matrix*, whose inverse undoes the input pose of the given character to generate the default *bind pose*.

Finding the initial pose of the unknown skeleton and unknown 3D shape result in an abundant degrees of freedom (DoF), and we fix the skeleton topology to hand the ambiguity of the problem. We adapt Bio-Vision Hierarchy (BVH) [MM01] to hierarchically represent the movement of the character's body as used in a motion capture system. The positions of bones in the skeleton are encoded as a sequence of relative transformations of the child bone with respect to its parent. The motion is encoded using the joint positions at the bind pose p_j^b and the relative joint orientations $\bar{R}_j$, which is set to identity at the default pose. Specifically, the transformation of individual bones under BVH can be written as

$$J_j B_j^{-1} = \begin{bmatrix} R_j & p_j \\ 0 & 1 \end{bmatrix} \begin{bmatrix} I_{3\times 3} & p_j^b \\ 0 & 1 \end{bmatrix}^{-1}, \ \bar{p}_j = R_{parent(j)} \bar{p}_j^b, \tag{2}$$

where $R_j = I_{3\times 3}$ and $p_j = p_j^b$ at the bind pose.

In our set-up, the initial pose is not the bind pose of the animation. Rather, we can consider the initial mesh as the output of an animated mesh under motion. With this representation, our goal is finding the bind pose B_j^{-1} and the skin weights w_{ij} from which we can animate our character with motion re-targeting algorithms [VYCL18, ALL*20]. The pipeline is outlined in Figure 2. We first find the joint position p_j (Sec. 2.1) followed by the joint orientation R_j (Sec. 2.2). Then the skinning weights for individual vertices are obtained (Sec. 2.3) to reconfigure the given character into the bind pose following Eq.(2). Each module is trained separately and combined to form a full pipeline.

2.1. Joint Position Estimation

The JointNet in Figure 2 extracts the probabilistic distribution of individual joint positions of an input 3D character. The heatmap based representation is widely used for human-pose estimation [TJLB14, NYD16, MYCML18] in computer vision. While the human-pose estimation in computer vision regresses for the 2D coordinates of joints on pixel space, our JointNet finds the 3D positions in 3D voxel. The input to the network is voxelized signed distance field of the given model and the output is 3D heat maps. Using the stacked hourglass architecture [NYD16] with 3D convolutions and skip connections, our network outputs the likelihood heat map for each joint. The network is trained using the cross-entropy loss compared against the Gaussian distribution centered at the ground-truth joint positions. The final joint positions are regressed to the maximum likelihood estimate from the heat map within the 3D voxel grid as in [XZKS19].

2.2. Joint Orientation Estimation

Given the skeletal topology of BVH, we estimate the bone lengths from the detected joints and define the bind pose of the

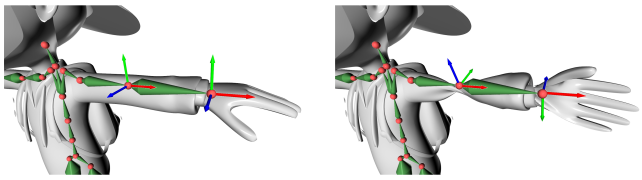

Figure 3: *If the template mesh in the bind pose (left) is not provided as in our setting, we need to use the correct bone transformation to deform the input posed mesh into the bind pose. Without correct orientation, candy wrapper effects can deteriorate the quality of the recovered template mesh in the bind pose (right).*

skeleton. However we cannot formulate the inverse kinematics (IK) which minimizes either the discrepancy obtained from projected template mesh [KBJM18] or mapping between animated rigs and the underlying skeleton [HSK15]. The main difference is that we do not have the template mesh at the bind pose to define the loss functions. Even if we attempt to solve to fit the skeletal joint positions without template, simple iterative IK solver results in awkward rotation as presented in Figure 3 (right). Instead, we adapted a learning-based module that learns the joint rotation in the form of BVH. Our module, denoted as RotNet in Figure 2, is composed of fully connected layers with batch norm. Given the joint positions relative to the root, our network regresses the joint rotation matrix in a 6D representation [ZBJ*19] using geodesic distance of rotation matrices as a loss function: $L_r = \sum_i \| \log(R_i^T \hat{R}_i) \|_F$, where R_i is the rotation matrix computed from the output 6D representation. With the predicted joint position and orientations, the joint position at the bind pose can be directly calculated using Eq. (2).

2.3. Skinning Weight Estimation

After we find the initial skeleton with their transformations B_j^{-1}, we need the skin weights w_{ij} to deform the input mesh into the bind pose (SkinNet in Figure 2). After computing volumetric geodesic distance from joints to vertices, we run graph convolution to find the skinning weights of individual vertices as in [LZT*19, XZK*20]. Treating the skin weights as a distribution over joints, we use soft cross entropy as a loss function $L_s = -\sum_{i,j} w_{ij} \log(\text{softmax}(\hat{w}_{ij}))$, measuring the distance between ground truth and predicted skin weights. For the final output, we filter up to 3 most relevant bones over the threshold of 0.2 and normalize them such that the skin weights for each vertex sum up to 1.

3. Experiments

We collect 64 characters in bind pose from the mixamo dataset [Ado20] where trainining, validation, and test splits contain 55-3-6 characters respectively. We first simplify the mesh to have less than 8K vertices, resulting in 2K to 8K vertices per mesh. We rig them to have the same skeletal structure with 22 joints and generate 7 motion sequences (585 frames) for each character. To create the SDF for JointNet, we voxelize the mesh and compute SDF based on morphological transform.

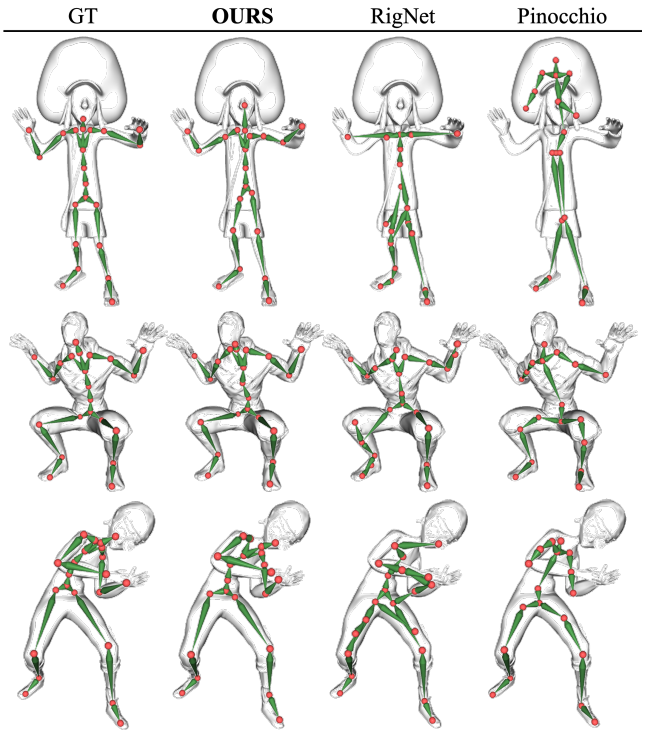

Figure 4: *Result of joint skeleton extraction. Other methods fail to create right topology or miss out important joints.*

Table 1: *Comparison with other skeleton prediction methods*

	MPJPE	S-EMD	IoU	Prec.	Rec.
Pinocchio	0.0913	0.146	50.9%	60.4%	50.9%
RigNet	0.0889	0.117	60.1%	64.9%	**69.5%**
Ours	**0.0766**	**0.0592**	**67.0%**	**70.5%**	69.4%

3.1. Skeleton Estimation

We compare our method with RigNet [XZK*20] and Pinocchio [BP07]. For fair comparison, RigNet's parameter is tuned to have the same average number of output joints. As can be seen in Figure 4, methods without a template create inconsistent skeletal structure and often fails to form plausible skeleton. However, our method does not suffer from inconsistent hierarchy and correctly finds the symmetric structure by enforcing the same skeletal topology.

The accuracy is quantitatively measured in Table 1. As discovered from the qualitative results, our result achieves the best accuracy for the joint position estimation. MPJPE (Mean Per Joint Position Error) is adapted from conventional pose estimation to measure the accuracy of the joint position, and computed by averaging the distance of each matching joints.

With the help of the skeletal topology, the structural recovery is more prominent using our method. The skeletal structure is provided as the collection of line segments of bones, and it is not trivial to compare the line segments as it is. Instead, we sample 300 points

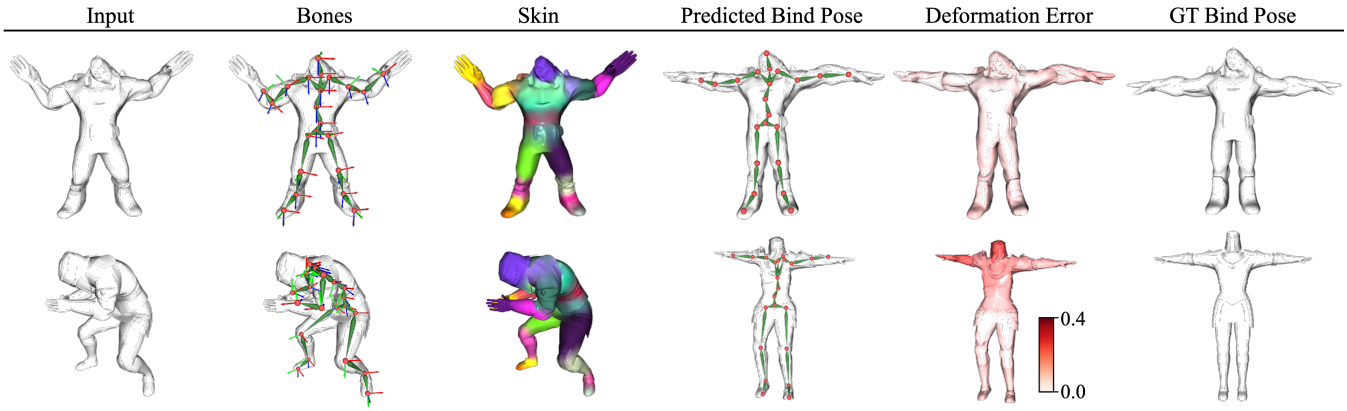

Figure 5: *Complete result of our pipeline. With the predicted joint and skin information, our method can animate the model back to the bind pose. Deformation error shows per vertex distance from the original bind pose mesh.*

such that each bone has about the same number of points, and S-EMD (Skeletal EM Distance) measures the Earth Mover Distance between the point distributions of the proposed skeleton and the ground truth. The metric measures the structural similarity such as symmetry, and our method shows superior result.

Additionally we measure IoU, precision, and recall of the set of joints as in [XZK*20] and other pose estimation problems. For matching distance, we compute the tolerance being the local shape diameter evaluated at corresponding joints. Our approach achieves the best IoU and precision. However, we sometimes suffer from symmetric ambiguity in extreme poses which results in performance degradation for recall.

3.2. Bind Pose Mesh

Since we are one of the first to find the bind pose mesh from arbitrary pose, there is no method to compare against. Instead, the bind pose template is compared against the ground truth in Figure 5. With the help of the rotational compensation, our approach greatly reduces possible artifacts and finds the reasonable bind-pose mesh to apply animation. The animated sequences are available in the accompanying video.

4. Conclusion

We propose an algorithm that successfully rigs a 3D character in arbitrary pose. Our method fixes the skeletal structure and regresses for joint rotation. From the full transformation information, we can rig the input mesh and deform it into the canonical pose. With the rigged bind-pose character in hand, we can combine our result with motion re-targeting to animate a wide variety of characters obtained online or through scanning. Future works include rigging arbitrary posed characters that can adapt changes in the skeletal structure.

References

[Ado20] ADOBE: Adobe's mixamo, 2020. URL: http://www.mixamo.com. 3

[ALL*20] ABERMAN K., LI P., LISCHINSKI D., SORKINE-HORNUNG O., COHEN-OR D., CHEN B.: Skeleton-aware networks for deep motion retargeting. *arXiv preprint arXiv:2005.05732* (2020). 2

[BP07] BARAN I., POPOVIĆ J.: Automatic rigging and animation of 3d characters. *ACM Transactions on graphics (TOG) 26*, 3 (2007), 72–es. 3

[HSK15] HOLDEN D., SAITO J., KOMURA T.: Learning an inverse rig mapping for character animation. In *Proceedings of the 14th ACM SIGGRAPH/Eurographics Symposium on Computer Animation* (2015), pp. 165–173. 3

[KBJM18] KANAZAWA A., BLACK M. J., JACOBS D. W., MALIK J.: End-to-end recovery of human shape and pose. In *Computer Vision and Pattern Regognition (CVPR)* (2018). 3

[LZT*19] LIU L., ZHENG Y., TANG D., YUAN Y., FAN C., ZHOU K.: Neuroskinning: Automatic skin binding for production characters with deep graph networks. *ACM Transactions on Graphics (TOG) 38*, 4 (2019), 1–12. 3

[MM01] MEREDITH M., MADDOCK S. C.: Motion capture file formats explained. 2

[MYCML18] MOON G., YONG CHANG J., MU LEE K.: V2v-posenet: Voxel-to-voxel prediction network for accurate 3d hand and human pose estimation from a single depth map. In *Proceedings of the IEEE conference on computer vision and pattern Recognition* (2018), pp. 5079–5088. 2

[NYD16] NEWELL A., YANG K., DENG J.: Stacked hourglass networks for human pose estimation. In *European conference on computer vision* (2016), Springer, pp. 483–499. 2

[TJLB14] TOMPSON J. J., JAIN A., LECUN Y., BREGLER C.: Joint training of a convolutional network and a graphical model for human pose estimation. In *Advances in neural information processing systems* (2014), pp. 1799–1807. 2

[VYCL18] VILLEGAS R., YANG J., CEYLAN D., LEE H.: Neural kinematic networks for unsupervised motion retargetting. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (2018), pp. 8639–8648. 2

[XZK*20] XU Z., ZHOU Y., KALOGERAKIS E., LANDRETH C., SINGH K.: Rignet: Neural rigging for articulated characters. *ACM Trans. on Graphics 39* (2020). 2, 3, 4

[XZKS19] XU Z., ZHOU Y., KALOGERAKIS E., SINGH K.: Predicting animation skeletons for 3d articulated models via volumetric nets. In *2019 International Conference on 3D Vision (3DV)* (2019). 2

[ZBJ*19] ZHOU Y., BARNES C., JINGWAN L., JIMEI Y., HAO L.: On the continuity of rotation representations in neural networks. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (June 2019). 3