SIGGRAPH: G: Interactive Rigging

Seungbae Bang, Byungkuk Choi, Roger Blanco i Ribera, Meekyoung Kim, Sung-Hee Lee, and Junyong Noh,
Korea Advanced Institute of Science and Technology (KAIST)

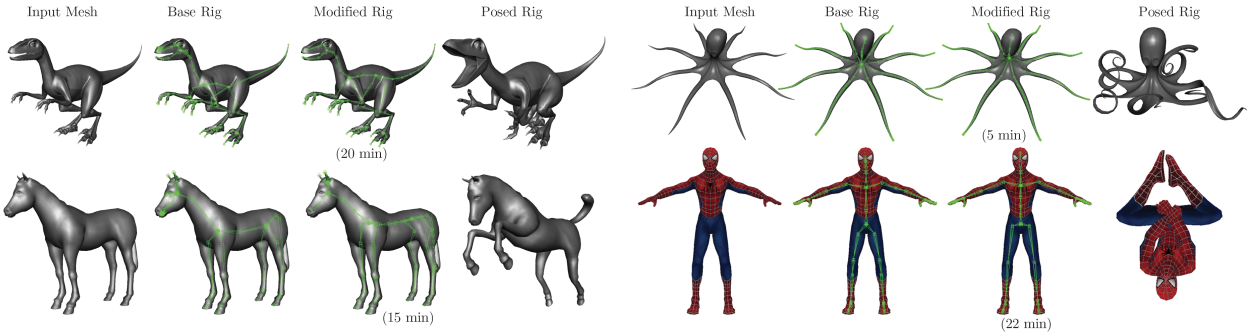


Figure 1: *Rigging results generated by our system.*

Abstract

Rigging is a core element in the process of bringing a 3D character to life. The rig defines and delimits the motions of the character and provides an interface for an animator with which to interact with the 3D character. The quality of the rig has a key impact on the expressiveness of the character. Creating a usable, rich, production ready rig is a laborious task requiring direct intervention by a trained professional because the goal is difficult to achieve with fully automatic methods. We propose a semi-automatic rigging editing framework which eases the need for manual intervention while maintaining an important degree of control over the final rig. Starting by automatically generated base rig, we provide interactive operations which efficiently configure the skeleton structure and mesh skinning.

1 Problem and Motivation

Skeleton-driven animation is a widespread technique which is frequently used in film and video game productions to animate 3D characters. The process of preparing characters for skeletal animation is referred to as character rigging. Commercial applications such as Maya or 3DS Max provide many tools that support this process, including the 'joint tool' and the 'paint skin weights tool'. But most of them are difficult to use for novice users. Even for professional artists, it requires many hours of intensive effort. Consequently, many studies have attempted to solve this problem with respect to the skeleton construction or the skinning process.

2 Background and Related Work

Automatic skeleton construction has received considerable attention. One approach is to extract a skeleton automatically by analyzing the properties of the character mesh[Au et al. 2008; Sharf et al. 2007; Cornea et al. 2007; Pan et al. 2009; Hauser et al. 2003]. Although these methods produce an initial skeleton from an arbitrary mesh, the resulting skeletons may not be readily applicable to animation. The skeletons often contain unnecessarily many joints. Additionally, the placement of the joints often requires anatomical knowledge that is difficult to be exploited by automatic methods. Another approach is to embed an existing rig into the mesh [Baran and Popović 2007; Seo et al. 2010]. These methods obtain animation-ready rigs, often lacking in options for customizing the results.

The skinning process determines the influences of the skeleton bones on the mesh. This process is performed by assigning appropriate weights to the vertices influenced by each bone. In professional environments, the skinning process requires a significant amount of back and forth manual refinement of the skin weights by highly skilled artists in order to ensure a smooth and plausible mesh deformation. Automating this process has been a subject of several studies [Baran and Popović 2007; Wareham and Lasenby 2008; Jacobson et al. 2011; Dionne and de Lasa 2013]. Many of which have mainly focused on delivering reasonable skinning results for novice users.

The automatic skeleton construction and skinning process have mainly been addressed independently by the research community. However, observations of artist work-flows reveal that both tasks are performed iteratively until satisfactory results are achieved. There are a few techniques, such as the Pinocchio rigging system [Baran and Popović 2007] or the work of *Pan et al.*[Pan et al. 2009], which fully address the rigging problem by considering the generation of a plausible skeleton and then applying a skinning procedure. In RigMesh [Borosán et al. 2012], the modeling step is included via a sketching interface, providing an end-to-end system for the creation of the character rigging. These methods mainly focus on presenting novice users with readily animatable character, often lacking in providing means for subtle refinement, which is critical for professional skinning process.

3 Approach and Uniqueness

In professional environments, the manual approach still takes a primary role, as it gives full freedom to control the result. In this paper, we approach the rigging process from a semi-automatic point of view, integrating the skeleton creation process with the skinning of the mesh into an interactive rig editing system [Bang et al. 2015]. We also maintain an important degree of control over the final result of the rig, especially in its ability to refine the skinning results. Our method begins by providing an automatically generated, fully skinned rig as a starting point for interactive editing. In this framework, the skeleton structure and the skin weights can be interactively edited with the provided manipulation tools while receiving immediate colorized visualization of the skin weights of the current state of the rig. Real-time visualization, which is one

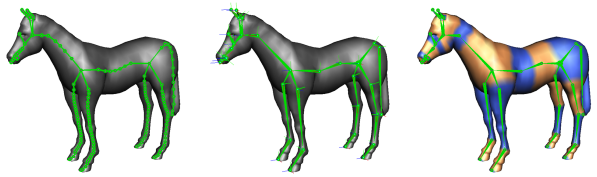


Figure 2: *(Left) The skeleton extraction result. (Middle) The resulting skeleton reduced by the DP algorithm where the hierarchy and the orientation information is specified. (Right) Corresponding regions obtained during the skeleton extraction step.*

of the most important aspects in interactive rigging, is achieved by re-computing the weights locally. These editing operations can be interleaved back and forth until a satisfactory result is achieved.

3.1 Base Rig

In order to obtain a fully skinned base rig, a skeleton is first extracted from the given mesh. Then a hierarchical joint structure is defined after the user specifies the root joint for the skeleton. Next, joint orientation information is determined and finally, the system computes the initial skin weights for the mesh.

3.1.1 Skeleton Extraction

We employ the skeleton extraction technique[Au et al. 2008] for the automatic skeleton generation of a given mesh $\mathbf{G} = (\mathbf{V}, \mathbf{I})$, where $\mathbf{V} = [v_1^T, v_2^T, ..., v_m^T]^T$ and $\mathbf{I}$ is the set of edges, with m being the number of vertices. This method contracts a mesh to approximately a zero volume using Laplacian smoothing with attraction constraints. A 1D graph skeleton is obtained by half-edge collapsing of the contracted mesh. This process produces a curve-skeleton that has node(i.e, joint) positions $\mathbf{U} = [u_1^T, u_2^T, ..., u_n^T]^T$, with n being the number of nodes. A corresponding region, a mapping between the extracted skeleton and the original mesh, is defined through the collapsing step. Specifically, the set of mesh vertices Π_k collapsing to each skeleton node $k(0 < k \leqq n)$ is tracked and the information is stored. The corresponding regions are then utilized during the interactive editing process to speed up the re-computation of the skin weight.

Since the skeleton extraction method[Au et al. 2008] often generates more nodes than necessary, we apply the Douglas-Peucker algorithm(DP)[Douglas and Peucker 1973] to remove extraneous nodes. We divide the skeleton to create a segment from one junction node to the next junction node or from one junction node to a terminal node. We then apply the DP algorithm to each segment. The left image in Fig. 2 shows a result from the skeleton extraction step. The middle image shows the cleaned skeleton after the applying the DP algorithm. It can be observed that only a small number of essential nodes remain. The right side shows the corresponding region of cleaned skeleton.

3.1.2 Hierarchy

A skeleton hierarchy is composed of a series of joint chains with hierarchical relationships. Upon the selection of the root joint, the remaining hierarchy is then established by traversing the nodes in a depth-first manner starting from the root. From the root joint J_1, all of the joints are indexed according to their traversal order with each joint J_k having one parent joint J_{P_k} and one or more children joints J_{C_k}. Because the skeleton extraction method produces an articulated skeleton, the number of bones is always one less than the number of joints. The relationship between bone and joint indices can be described as follows: B_i is the bone corresponding to joint J_{i+1} and its parent: $\overline{J_{i+1}J_{P_{i+1}}}$, $(0 < i < n)$. With the established hierarchical relationship, a transformation applied to a specific joint will propagate through its children joints.

3.1.3 Orientation

The orientation of a joint is important for the manipulation of the joint. It is a common convention to assign the primary axis of a local joint to the direction pointing to its child joint. It is also usual to align the joint's secondary axis, also referred to as the up-vector, to the bending direction of the joint. Animators sometimes manipulate several joints, such as finger joints, at the same time. Keeping a coherent aligned up-vector of joint setup leads to easily keyable bending motions with just one axis.

Our system automatically defines up-vectors for branches with three joints. Considering three consecutive joints J_a, J_b, and J_c of such a branch, the up-vector is defined as the vector pointing outward in the direction of the projection of J_b onto $\overrightarrow{J_aJ_c}$.

3.1.4 Initial Skin Weights

The skin weight $w_j^i(0 < i < n, 0 < j \leqq m)$ denotes the influence of B_i over the vertex j. Similar to [Baran and Popović 2007], we employ the heat diffusion weight. This method assumes that each bone radiates heat onto the nearby surface. Solving the heat equilibrium over the surface leads to the resulting skin weight vector w^i for bone B_i. The equation can be written as follows:

$$(\mathbf{H} - \mathbf{L})w^i = \mathbf{H}p^i \tag{1}$$

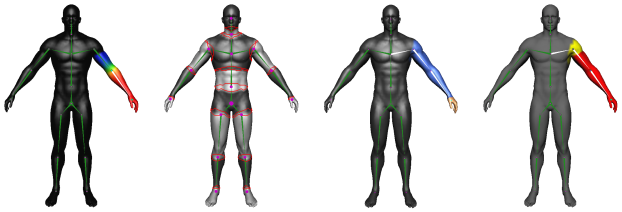
Here, $\mathbf{H}$ is the diagonal matrix with $H_{jj} = c/d^2(j)$ with $d(j)$ being the distance from vertex j to the nearest bone. If the nearest bone is not visible from vertex j, then $H_{jj} = 0$. Similarly to the original method, c is set to one in this step initially. $p^i(\in \mathbb{R}^m)$ is a vector with $p_j^i = 1$, if the bone closest to vertex j is i and $p_j^i = 0$ otherwise. $\mathbf{L}$ is the discrete surface Laplacian matrix obtained from the mesh contraction step mentioned in Sec. 3.1.1 Stacking the weight vectors creates the matrix $\mathbf{W} = [w^0, w^1, ..., w^{n-1}](\in \mathbb{R}^{m \times n-1})$. Fig. 4(a) shows the resulting initial skin weights corresponding to spine bone of a horse character.

3.2 Interactive Editing

Once the base rig is created, the user is presented with several tools to modify and the rig to his/her needs. We provide two different types of manipulations of the rig. The user can manipulate the skeleton to refine the position and orientation of the joints or to modify the skeleton structure by inserting and deleting joints. The user can also edit the character skinning and receive immediate visual feedback on the modifications with a color gradient visualization of the skin weights of a selected bone. We provide intuitive skin weight handles to delimit the influence and area of influence of the bones over the mesh. To afford plausible interactivity, we re-compute the skin weights locally by exploiting the information about the corresponding regions obtained in Sec. 3.1.1. This local re-computation is important, as global re-computation hardly achieves real-time performance depending on input mesh resolutions.

3.2.1 Local Skin Weight Re-computation

Local Computing Region For any changes occurring to a bone, we want the users to receive instantaneous visual feedback on the updated weights while they operate on the rig. Real-time computation is difficult to achieve by solving Eq. 1 naively, as it computes



(a) (b) (c) (d)

Figure 3: *(a) Resulting skin weights for the white bone on the lower left arm. (b) Regions corresponding(grey) to each joint(purple). Boundary vertices(red) between corresponding regions. (c) Combined corresponding regions (orange and blue) and bones to be recomputed (white). (d) The final local computing region included in our skin weight re-computation algorithm (red and yellow).*

weights over all the vertices in the mesh for every bone. Our approach is to compute the heat equation on a subset of the mesh and skeleton and then update the resulting weights. This raises the question of which region and which bones should be involved in the weight re-computation. Fig. 3(a) shows the initial heat weights of a human character for the bone in the lower left arm. The region colored in black indicates a zero-valued weight. It can be observed that large portions of the mesh received a zero-valued weight. The figure shows that the region and the bones to be recomputed are only those in the vicinity of the manipulated joint.

We utilize the information about corresponding regions obtained during the skeleton extraction process in Sec. 3.1 to determine an appropriate local computing region and the bones to be recomputed. Fig. 3(b) shows the region corresponding to each joint with a segmented color of dark grey and light grey. Boundary vertices between corresponding regions are colored in red. When the user edits joint J_k, the local computing region Γ_k is first defined as the union of several corresponding regions, i.e, $\Gamma_k = \bigcup_{t \in A_k} \Pi_t$, where A_k is a subset of joint indices. The bones to be recomputed are defined as the union of the bones connected to joints $J_t (t \in A_k)$. Starting with only the corresponding region Π_k associated with joint J_k, the local computing region is expanded by including adjacent regions until the weight of every vertex in the boundaries of the local computing region is zero for bone B_{k-1}. Fig. 3(c) shows the combined corresponding regions and bones to be recomputed. The corresponding region associated with the joint on the left wrist is colored in orange. The regions that were added to the local computing region are colored in blue. Bones to be recomputed are colored in white. To narrow down the range of the local computing region of the mesh further, we exclude the vertices with zero-valued weights for bone B_{k-1} that are farther than c-neighbor distance from the vertices with non-zero weights from Γ_k. The zero-weighted vertices within c-neighbor distance are included to ensure weight continuity between the local computing region and the rest. Empirically, we found that $c = 4$ worked well for our purpose. Fig. 3(d) shows the local computing region in red, in which zero-valued weights are excluded. The region covering the c-neighbour distance from the boundary vertices is also colored in yellow. Combining these two colored regions results in the final computing region.

The corresponding region is only a starting point to determine a local computing region. Thus, it need not be defined precisely as long as there exists one associated with every joint. For this reason, the corresponding region is updated only when there is a change in the hierarchy of the skeleton, such as an insertion or deletion of joints.

Local Computation of Weight With the help of the local computing region, the system effectively reduces the size of the region involved in the weight re-computation. As the mesh structure remains same, we re-use the Laplacian matrix $\mathbf{L}$ constructed in Sec. 3.1, but this time only a subset for the local computing region is used. As the skeleton structure changes during the editing process, the $\mathbf{H}$ matrix needs to be rebuilt for the local set. If joint J_k is modified, the following equation is newly computed for the bones with index $i((i-1) \in A_k)$.

$$\mathbf{B}(\mathbf{H}-\mathbf{L})\mathbf{A}w^i + \mathbf{B}(\mathbf{H}-\mathbf{L})\mathbf{B}^T w^i = \mathbf{B}\mathbf{H}\mathbf{B}^T p^i \qquad (2)$$

Here, $\mathbf{B} \in \mathbb{R}^{d \times m}$ and $\mathbf{A} \in \mathbb{R}^{m \times (m-d)}$ are the selection matrices such that $\mathbf{A}w^i \in \mathbb{R}^{m-d}$ is the skin weight vector for the vertices in non-computing region and $\mathbf{B}^T w^i \in \mathbb{R}^d$ is that for the vertices in local computing region. The first term in Eq. 2 serves as the boundary condition. This ensures continuity across the boundary vertices of the local set. This local weight re-computation process occurs according to changes in the joint position or joint hierarchy arising from the manipulations explained in the following sections.

The idea of local re-computation originated from RigMesh[Borosán et al. 2012]. In their method, local computation occurs when two shapes' surfaces and skeletons merge. They determine the local computing region by searching from the merge boundary for vertices of which their closest and visible bone is assigned with under average weight. In contrast, our method utilizes the segmented mesh obtained during the skeleton extraction operation to determine the local computing region.

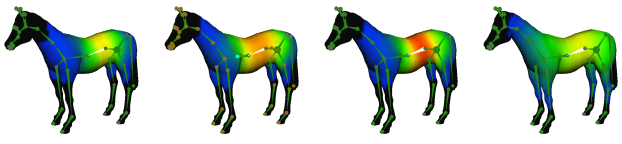
3.2.2 Skeleton Manipulations

We provide tools that intuitively and easily edit the skeleton. During all of the following operations, the system internally re-computes the skin weights according to the changes applied. We then provide interactive color visualization of the skin weights to help users choose the final positions of the bones.

Joint Positioning The user can re-adjust the joint position continuously in space. The system re-computes the skin weights interactively with the user's modifications. For this reason, this operation requires especially fast computation. With the help of the local re-computation of the skin weights introduced in Sec. 3.2.1, a user can perform this operation without delay of the system caused by heavy computations.

Joint Insertion The user can insert new joints into the skeleton. This insertion operation adds a new joint between the selected joint J_k and its parent joint J_{P_k}. Index $k+1$ is assigned to the new joint and the indices larger than $k+1$ are incremented by one. The new joint J_{k+1} extracts the boundary vertices from $\Pi_{P_{k+1}}$ and set them as its corresponding region Π_{k+1}. Because there are no previous skin weights for the newly generated bone, we compute the skin weights using Eq. 1 for this operation but with only bones with index $i((i-1) \in A_k)$. The new weight vector w^k is inserted into the kth column of weight matrix $\mathbf{W}$.

Joint Deletion Except for the root joint, the user is allowed to delete any joint in the hierarchy. If a joint J_k is deleted, joint indices larger than k are decremented by one. The local computing region is set to Γ_{P_k} and the bones with index $i((i-1) \in A_{P_k})$ are set as re-computing bones. The $(k-1)$th column of weight matrix $\mathbf{W}$ is removed. The corresponding region of deleted joint Π_k is transferred to Π_{P_k}.



(a) (b) (c) (d)

Figure 4: *(a) Initial Skin Weights. (b) Editing Influence Range. (c) Scaling Influence Magnitude. (d) Editing Influence Smoothness.*

Joints Alignment There are several cases, such as in fingers or spine bone chains, for which the joints need to be aligned on a particular plane while maintaining up-vectors arranged in the same plane in order to have a common control axis for bending them. We provide an alignment operation to align more than three joints. The user can select the joints that should be placed together on one plane. The system then automatically finds the optimal up-vector for those joints. Denoting *X* as a set of indices selected by the user, we can find the hierarchically highest joint J_h, and the lowest joint J_l from the selected joints ($h, l \in X$). Using the same method presented in Sec. 3.1.3, we iterate over every three sequential joints resulting in a set of vectors $U = \{\overrightarrow{U_1}, \overrightarrow{U_2}, ..., \overrightarrow{U_n}\}$ with $n = |X|-2$, where $|X|$ is the number of joints selected by the user. We set the up-vector for the selected joints as the average vector of $\overrightarrow{U_{avg}}$. Finally, all of the selected joints are projected onto the plane defined by $\overrightarrow{J_h J_l}$ and $\overrightarrow{U_{avg}}$.

3.2.3 Skin Weight Manipulations

The skinning weights control the deformation of the mesh. We define three key operations to control the influence of a bone over the mesh. First, the range of influence of a bone affects the extent of the area to be deformed according to the bone's transformation. Second, the magnitude of the influence of a bone determines how strongly the area will be affected by the bone's transformation. Third, the smoothness of the influence of a bone determines overall smoothness of weight propagation. Fig. 4(b-d) shows the three types of skin weights manipulation.

Editing Influence Range We provide a means of controlling the spread of a bone influence by associating each joint with a skin weight handle. This handle controls the boundary between the influence region of a bone and that of its neighbour. This handle is originally placed at each joint position. By moving the handle, the user can increase or reduce the spread of the bone's influence in that direction. According to Eq. 2, the area of influence of a bone is mainly determined by the shape of the bone as it directly influences the distance term in $H_{jj} = c/d(j)^2$. By controlling a joint, the boundary of the area of influence of its associated bones can be controlled. It is, however, important to separate the skeleton manipulations and the editing of the skinning weights. We thus augment our skeleton by adding a skin weight handle to each joint. This skin weight handle is parented to the joint. We use the position of this handle to compute the distance term of the matrix H. This can be seen as computing the weights for a set of virtual bones associated with the skin weight handle positions.

This operation works similar to 'Interactive Skin Bind Tool' in Maya, which provides a sphere-like controller that determines the influence of skin weights. While the Skin Bind Tool can spread the influence only on a sphere-shaped domain. Our method can adjust the weights with shape-awareness.

Scaling Influence Magnitude We allow the user to manipulate the radius of a bone as a means of controlling the importance of a bone during the weight assigning step. We attach the radius scale to the heat radiation of the bone: A larger bone would radiate larger amounts of heat compared to a smaller bones. We use the same Eq. 2 but for $H_{jj} = c/d^2(j)$, we define c as the $scale\ factor$ of the radius of B_k instead of using 1.

Editing Influence Smoothness We provide an intuitive operation to control the smoothness of skin weights. By adopting the idea of Euler-Lagrange equation[Botsch and Sorkine 2008] minimizing thin shell energy[Terzopoulos et al. 1987], we modify the original equation of heat diffusion weight by adding bi-Laplacian operator matrix $\mathbf{L}^2$.

$$(\mathbf{H} - (a\mathbf{L} + b\mathbf{L}^2))w^i = \mathbf{H}p^i, \begin{pmatrix} a > 0 \\ b > 0 \\ a + b = 1 \end{pmatrix} \quad (3)$$

Here, a and b are the weight for Laplacian matrix and bi-Laplacian matrix, respectively. We set $a = 1, b = 0$ for initial setup. Interpolating between a and b with sum of unity gives control for smoothness. By scaling the manipulator, a weight for a gets smaller and a weight for b gets bigger, which result in smoother propagation of skin weights. The surface-oriented bilaplacian has some advantages over BBW [Jacobson et al. 2011]. We don't need a volumetric discretization and the computations are much faster.

Some tools in commercial application provide supports similar function. A tool like 'paint skin weights tool' in Maya provides a function for smoothing the transition of weights with a brush-like controller. However, it can only smooth transition in the bound of the brush size. It takes quite a time to modify overall smoothness of skin weights.

4 Results and Contributions

Implementation We implemented the system as a plugin for Autodesk Maya. Thus we followed the weight coloring convention in Maya to maintain consistency within the application. The skinning process is usually a back-and-forth process between modifying skin weight values and checking the deformation results for different poses of the skeleton. Visualizing the deformations obtained from the skin weights is an important part of the skinning process. Setting a pose and checking the deformation for the current weight values provide cues about how to improve the skin weights. We thus provide a feasible working framework within Maya with regard to this work-flow. From a given mesh, our system generates a distinct custom mesh node with the rest pose of the original mesh for which the rig is manipulated and the skin weights are visualized. On the original mesh, our system generates a fully Maya-compliant rig. With this setup, the user can pose and key-frame the rig on the original mesh and obtain instantaneous visual feedback about the manipulations of the skeleton and skin weights performed on the custom mesh.

Rigging Results Our system can handle any mesh that is compatible with the skeleton extraction process. Fig. 1 shows various characters rigged by our method and deformation results with LBS. These results verify the generality of our method. Rigging of one character required nearly up to 20 minutes by a novice user. Standard linear blend skinning(LBS) was used to generate a pose.

			#Vertices involved	#Bones involved	time (ms)
Armadillo	Global		43243	82	743
	Local	foot	7663	17	61
		knee	10911	16	73
		chest	14312	20	131
		arm	14403	19	101
Lizard	Global		32964	191	1185
	Local	fore-leg	3948	35	46
		hind-leg	3731	100	30
		neck	2648	53	28
		tail	1953	148	30
Human	Global		9832	16	66
	Local	hand	1758	3	22
		shoulder	2404	8	22
		chest	2622	8	19
		head	3538	8	20

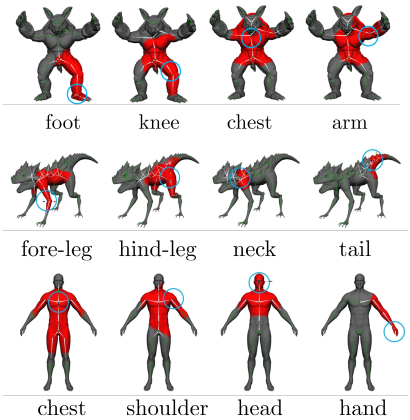


Table 1: *Comparison of the computation time of the skin weights between the global computation and local computation of our method.*

Local Skin Weight Re-Computation The computation of the skin weights in the previous method[Baran and Popović 2007] consists of two steps: preparing data for the linear solver and the solving it. Our local skin weight re-computation method requires one additional step of searching through vertices to define the local computing region. Although this takes extra time to solve the system, it drastically reduces the overall compute time, as the solving process is the bottleneck of the computation. Table. 1 shows a performance comparison between Pinocchio[Baran and Popović 2007] and our method. Our system achieves 3∼40 times the performance gain compared to the global weight computation depending on the model. All of the experiments were performed on a 2.4GHz Intel Xeon with six gigabytes of memory under Ubuntu 12.04.

Contributions We proposed a new method that allows very intuitive and efficient character rigging. While our method grants a higher degree of control compared to previous methods, it is still not possible to achieve full control of skin weights comparable to painting the skin weights manually onto the vertices. Our method, however, provides enough flexibility to be applied to secondary characters or to be used as a rapid way to produce a functional rig for pre-visualization purposes. Given that our system is implemented as a Maya plugin, a user can also utilize the skin weights tool in Maya to refine fine details on top of our result.

References

AU, O. K.-C., TAI, C.-L., CHU, H.-K., COHEN-OR, D., AND LEE, T.-Y. 2008. Skeleton extraction by mesh contraction. In *ACM Transactions on Graphics (TOG)*, vol. 27, ACM, 44.

BANG, S., CHOI, B., BLANCO I RIBERA, R., KIM, M., LEE, S.-H., AND NOH, J. 2015. Interactive rigging with intuitive tools. In *Computer Graphics Forum*, vol. 34, Wiley Online Library, 123–132.

BARAN, I., AND POPOVIĆ, J. 2007. Automatic rigging and animation of 3d characters. In *ACM Transactions on Graphics (TOG)*, vol. 26, ACM, 72.

BOROSÁN, P., JIN, M., DECARLO, D., GINGOLD, Y., AND NEALEN, A. 2012. Rigmesh: automatic rigging for part-based shape modeling and deformation. *ACM Transactions on Graphics (TOG) 31*, 6, 198.

BOTSCH, M., AND SORKINE, O. 2008. On linear variational surface deformation methods. *Visualization and Computer Graphics, IEEE Transactions on 14*, 1, 213–230.

CORNEA, N. D., SILVER, D., AND MIN, P. 2007. Curve-skeleton properties, applications, and algorithms. *Visualization and Computer Graphics, IEEE Transactions on 13*, 3, 530–548.

DIONNE, O., AND DE LASA, M. 2013. Geodesic voxel binding for production character meshes. In *Proceedings of the 12th ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, ACM, 173–180.

DOUGLAS, D. H., AND PEUCKER, T. K. 1973. Algorithms for the reduction of the number of points required to represent a digitized line or its caricature. *Cartographica: The International Journal for Geographic Information and Geovisualization 10*, 2, 112–122.

HAUSER, K. K., SHEN, C., AND O'BRIEN, J. F. 2003. Interactive deformation using modal analysis with constraints. In *Graphics Interface*, vol. 3, 16–17.

JACOBSON, A., BARAN, I., POPOVIC, J., AND SORKINE, O. 2011. Bounded biharmonic weights for real-time deformation. *ACM Trans. Graph. 30*, 4, 78.

PAN, J., YANG, X., XIE, X., WILLIS, P., AND ZHANG, J. J. 2009. Automatic rigging for animation characters with 3d silhouette. *Computer Animation and Virtual Worlds 20*, 2-3, 121–131.

SEO, J., SEOL, Y., WI, D., KIM, Y., AND NOH, J. 2010. Rigging transfer. *Computer Animation and Virtual Worlds 21*, 3-4, 375–386.

SHARF, A., LEWINER, T., SHAMIR, A., AND KOBBELT, L. 2007. On-the-fly curve-skeleton computation for 3d shapes. In *Computer Graphics Forum*, vol. 26, Wiley Online Library, 323–328.

TERZOPOULOS, D., PLATT, J., BARR, A., AND FLEISCHER, K. 1987. Elastically deformable models. In *ACM Siggraph Computer Graphics*, vol. 21, ACM, 205–214.

WAREHAM, R., AND LASENBY, J. 2008. Bone glow: An improved method for the assignment of weights for mesh deformation. In *Articulated Motion and Deformable Objects*. Springer, 63–71.