

Frankenrigs: Building Character Rigs From Multiple Sources

Christian Miller*
The University of Texas at Austin

Okan Arıkan†
Animeeple Inc.

Don Fussell
The University of Texas at Austin

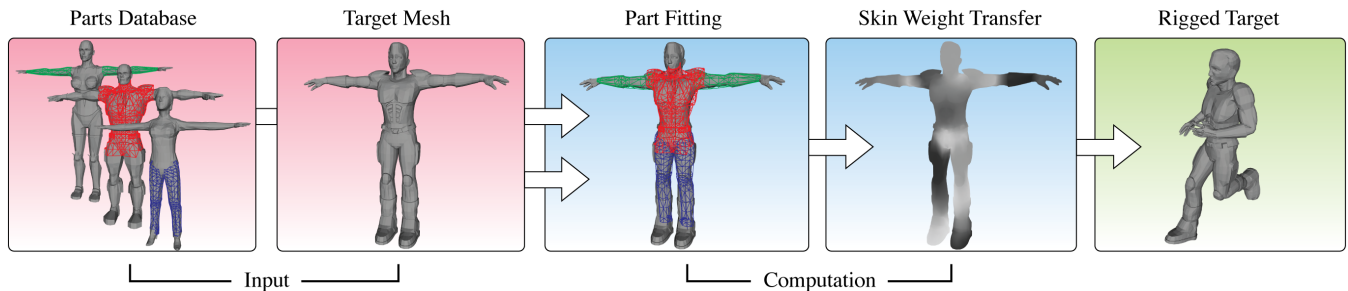


Figure 1: Frankenrigs can use body parts from a database to rig and skin new meshes. Given a database of partial source rigs and a target mesh, the system can automatically match and fit body parts to the target mesh and transfer their skinning information, creating a fully rigged and skinned character. We also include an intuitive set of tools to help the user resolve any remaining artifacts.

Abstract

We present a new rigging and skinning method which uses a database of partial rigs extracted from a set of source characters. Given a target mesh and a set of joint locations, our system can automatically scan through the database to find the best-fitting body parts, tailor them to match the target mesh, and transfer their skinning information onto the new character. For the cases where our automatic procedure fails, we provide an intuitive set of tools to fix the problems. When used fully automatically, the system can generate results of much higher quality than a standard smooth bind, and with some user interaction, it can create rigs approaching the quality of artist-created manual rigs in a small fraction of the time.

CR Categories: I.3.7 [Computer Graphics]: Three-Dimensional Graphics and Realism—Animation

Keywords: Rigging, Skinning, Rig Reuse, Character Animation

1 Introduction

The process of creating an animated 3D character involves several steps: design, mesh creation, rigging, skinning, and animation. Each of these steps is traditionally done by hand, and requires significant time and skill to get right; in this work we focus on the issues of rigging and skinning in particular. Creating a rig involves making an articulated skeleton and a set of controls that are exposed to the animator for the purpose of animating the character. Skinning a character is the process of tying those controls to mesh deformations. This step usually takes the form of painting bone influences on the surface of the mesh, a tedious and unintuitive task.

We present a method that can perform high-quality rigging and

skinning with little to no interaction from the user, thus reducing the time and difficulty involved in character creation. We accomplish this by gathering a database of pre-existing rigs and extracting usable body parts from it. Given an unrigged target mesh, we can construct a new rig by adapting and fitting together pieces from several different source meshes in the database. This “Frankenrig” inherits the local skeletal structure of the source meshes, and a correspondence-driven transfer procedure is used to reconstruct the skinning information on the target mesh. Along the way, we are forced to overcome several undesirable properties of real character meshes, such as the presence of multiple connected components and non-manifold geometry.

The primary contributions of this paper are twofold: First, we present a fitting procedure for adapting and aligning body parts from one mesh to another. Second, we present a skinning transfer method that is capable of blending deformation information from several partial source meshes onto a single target mesh.

Using these tools, we demonstrate the ability to quickly and automatically generate rigs of significantly higher quality than those achieved with standard smooth bind techniques. With a little user help, we demonstrate the ability to create rigs comparable to those created by artists in a small fraction of the time. Unlike many other skinning systems, we preserve editability; the character mesh is not changed in any way, and the output is a standard skeleton and weight map representation of skinning information, which can be further modified in any modeling and animation software.

2 Previous Work

The *de facto* standard for deforming animated characters has long been Linear Blend Skinning (LBS), also referred to as Skeleton Subspace Deformation (SSD), which uses an articulated skeleton and per-vertex weights to specify deformations. As the skeleton is moved, vertices deform as a weighted linear combination of the bones they are attached to. This technique is ubiquitous in films and video games, primarily due to its simplicity and efficiency.

However, these advantages come at the cost of distracting deformation artifacts around joints that are bent or twisted too far. Furthermore, LBS is incapable of representing complex deformations such as muscle bulges. Much research has been done to create efficient skinning methods that mimic more complicated deformations

*e-mail: ckm@cs.utexas.edu

†e-mail: okan@animeeple.com

by training against a set of examples. Mohr and Gleicher [2003] accomplish this by adding additional rotation and scale joints to the skeleton. Several other works define spaces or reduced representations of deformations, often layered on top of LBS, to help represent muscle bulges ([Lewis et al. 2000], [Kry et al. 2002], [Weber et al. 2007], [Wang et al. 2007]).

Creating rigs and skinning them, no matter the deformation technique used, is a difficult and time-consuming task, and evaluation of results is up to artistic interpretation. There has been a significant amount of research done into extracting ([Liu et al. 2003], [Kin-Chung Au et al. 2008]) and embedding ([Aujay et al. 2007]) skeletons into character meshes, but these do not address the skinning problem. Baran and Popović [2007] developed a system that can automatically embed a skeleton and derive skin weights for arbitrary meshes. To date, this is the only published system that can rig and skin meshes without any user intervention.

Another avenue of content creation is the reuse of existing rigging and skinning information. For example, deformation transfer techniques (like [Sumner and Popović 2004]) map deformations from one mesh onto another. Deformation cages ([Joshi et al. 2007], [Ju et al. 2008]) work by embedding a character into a deformable volume, and thus allowing easy transfer among meshes. Poirier and Paquette [2009] employ graph-matching techniques to retarget character rigs into new meshes, though they do not handle the transfer of skinning information. Orvalho [2007] demonstrates a system that can perform full-scale facial rig retargeting, which has since become a commercial product.

Our work falls into this class of research; to the best of our knowledge, we are the first to perform rig retargeting and skin transfer using multiple separate sources. Our system uses LBS as its underlying deformation technique, but can be extended to incorporate any technique that uses a skeleton and weight map, such as Dual Quaternion Skinning [Kavan et al. 2008].

3 Overview

A roadmap of our technique is as follows: First, as a preprocess, a database of partial source rigs is constructed from a set of fully-rigged characters (Section 4). The user then provides a new character mesh to be rigged and tags the approximate locations of a set of important joints within it. The system will scan through the database and tailor each source rig to fit the appropriate part of the target mesh as closely as possible (Section 5). These modified source rigs are automatically scored, and the best fits are used for the rest of the process. The user can optionally override the automatic choices for source rigs, if so desired. Once selected, the source rigs are assembled to form a new skeleton for the target mesh.

Next, skinning information is transferred from the source rigs to the target mesh by a two-phase procedure (Section 6). In the first phase, correspondences between the source rigs and target mesh are generated, both automatically and by optional user input. Skinning information is transferred along each correspondence onto the target mesh by solving a velocity matching problem. The correspondences do not necessarily cover the target mesh, so in the second phase the remaining skinning information is filled in by diffusing the correspondence transfer results over the surface of the mesh. The result is a fully rigged and skinned target mesh.

4 Database Construction

Our system attempts to create new rigs by recycling body parts from a collection of source rigs, thus, the first step is building a database

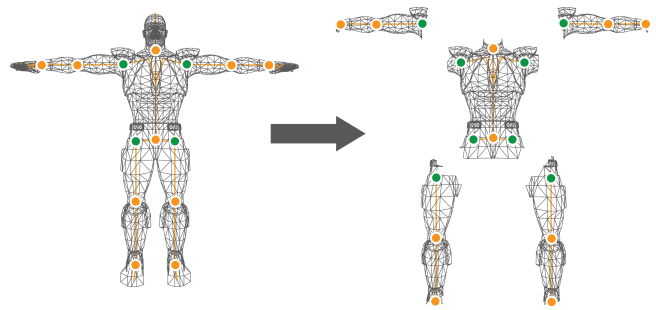


Figure 2: *Annotating and splitting body parts. The dots are user-provided locations of important joints; green dots represent seam joints, where the skeleton is severed to create separate body parts. Note that some parts of the mesh are extracted by more than one body part, but the skeletons do not overlap.*

of such body parts. Given a set of source rigs, the user begins by tagging a set of important joints in each rig. These joints should be common to all the source rigs, and mark locations where major deformations are likely to occur. The set can be chosen depending on the type of creature being rigged; our database consists entirely of bipeds, so we picked the set of 14 joints shown in Figure 2. These tagged joints are later used as cues for body part alignment and resizing.

Next, the user determines how the input rigs are split into body parts by identifying a set of *seam joints* from the collection of tagged joints. The source rig skeleton is severed at each of the seam joints, splitting it into a number of body parts which each contain a subtree of the original skeleton. These too can be chosen depending on the creature being rigged; good seam joints separate the body at the bases of major limbs, leaving body parts which are roughly cylindrical. Since our database consists of bipeds, we chose the shoulders and hips as the seam joints. This generated 5 body parts for each source rig: a torso, two arms, and two legs.

Once the seam joints have been identified, an automated exporter will generate a series of “chunk” files, one for each body part from each source rig. Each chunk contains the subtree of the tagged source rig skeleton corresponding to its particular body part, as well as all vertices and skin weights attached to any bone in that subtree. Note that this means some duplication of the source mesh among body parts; furthermore, blend regions around seam joints will have skin weights “fade out” on one body part and “fade in” on another. This partial information is crucial, because it helps determine how to blend together source rig contributions during the skinning transfer steps later.

It is worth noting that the source rigs are only required to contain the set of tagged joints, and no other constraints are imposed on the structure of the skeleton. In particular, body parts extracted from different source rigs do not need to share the same set of bones or degrees of freedom. This allows the system to propagate the skeletal detail of the source rigs into the rig created for the target mesh. Other rig elements such as IK handles and deformers could be exported as well, although we left this for future work.

Database construction is a preprocessing step; once completed, a database can be used to rig an arbitrary number of targets. More source rigs can be added to enrich the database at any time, as the extraction process operates on each rig individually.

5 Part Fitting and Selection

With the database built, the user can provide a target mesh for rigging, and the next step is to scan through the database for appropriate body parts to scavenge. To summarize, this is done by applying a multi-step fitting procedure to each potential source rig, which squashes and stretches it to match the appropriate part of target mesh as closely as possible. Once the source rigs have been tailored as best as possible, they are scored by a function that estimates how well the fitting procedure worked, and sorted to find the best choices to use for the target mesh’s new rig. The user can optionally interact with some of these steps to improve the quality of the results.

5.1 Required Annotations

Before the source rigs can be fitted to the target mesh, the target must first be annotated with the locations of the same joints marked in the source meshes (Figure 2). These joint location tags can be approximate, but they are used as important hints during the fitting process.

5.2 Stretching Limb Proportions

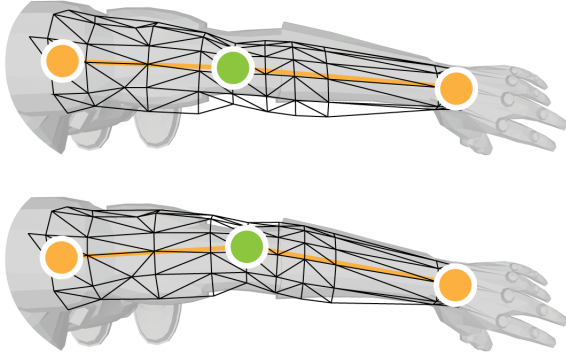


Figure 3: Adjusting limb proportions. The source rig does not originally line up with the target mesh (top), and is repositioned by moving the elbow to match the user’s annotation (bottom).

The first step in the alignment procedure is to transform and uniformly scale every body part in the database such that its tagged joints line up as well as possible with the same joints in the target mesh. Rotations are not performed along a body part’s major axis, to prevent accidentally reorienting cylindrical parts such as arms and legs. This puts the partial source rigs in roughly the right place for further modifications.

Next, the system repositions body parts containing at most two seam joints (such as arms and legs) to better match the joint tags provided by the user in the target mesh. Using an arm as an example, the shoulder and wrist joints are held fixed, and the elbow joint is moved to line up with the user-provided elbow location in the target mesh. The upper and lower arm segments are rotated and scaled to match the edit, and the source mesh vertices are reskinned using the new bone transforms. See Figure 3 for an example.

While many methods can be used to reposition body parts, we have found that this procedure adjusts the source rigs in a manner that best respects their function and preserves their structure.

5.3 Nonuniform Scaling ICP

Once the source rigs have been aligned and repositioned, the final fitting step is to adjust the source meshes to match the target as much as possible. This is accomplished by using a variant of Iterative Closest Point (ICP) matching that allows the solver to deform the source mesh by nonuniform scaling. This has two benefits: it lines up the source and target meshes to aid in transferring skinning information, and refines the location of the joints from the original user estimates (Section 5.1). Our method is an extension of the uniform scaling ICP method described in [Ziřner et al. 2005] and is a simpler version of the system described in [Du et al. 2007].

ICP works by, at each iteration, finding closest point pairs between two meshes and transforming one mesh to minimize the distances. Given the set of closest point pairs $(\mathbf{a}_i, \mathbf{b}_i)$ for one iteration (drawn from the source and target meshes, respectively), we perform nonuniform scaling ICP by solving the following problem:

$$\min_{\mathbf{s}, \mathbf{R}, \mathbf{t}} \sum_i \|\text{diag}(\mathbf{s})\mathbf{R}\mathbf{a}_i + \mathbf{t} - \mathbf{b}_i\|^2 \quad (1)$$

where $\mathbf{s}, \mathbf{R}, \mathbf{t}$ are the optimal scale, rotation, and translation parameters. This problem is separable and can be solved as follows: First, compute the centroids $\bar{\mathbf{a}}$ and $\bar{\mathbf{b}}$ of \mathbf{a}_i and \mathbf{b}_i respectively. \mathbf{R} can then be extracted via computing the singular value decomposition (SVD) of the covariance matrix \mathbf{K} :

$$\mathbf{K} = \sum_i (\mathbf{b}_i - \bar{\mathbf{b}})(\mathbf{a}_i - \bar{\mathbf{a}})^T = \mathbf{U}\mathbf{D}\mathbf{V}^T \quad (2)$$

$$\mathbf{R} = \mathbf{U}\mathbf{V}^T \quad (3)$$

Next, \mathbf{s} can be solved by:

$$\mathbf{s} = \sum_i \tilde{\mathbf{b}}_i \otimes \tilde{\mathbf{a}}_i \oslash \sum_i \tilde{\mathbf{a}}_i \otimes \tilde{\mathbf{a}}_i \quad (4)$$

$$\tilde{\mathbf{b}}_i = (\mathbf{b}_i - \bar{\mathbf{b}}) \quad (5)$$

$$\tilde{\mathbf{a}}_i = \mathbf{R}(\mathbf{a}_i - \bar{\mathbf{a}}) \quad (6)$$

where \otimes and \oslash represent component-wise multiplication and division, respectively. Finally, \mathbf{t} is solved by $\mathbf{t} = \bar{\mathbf{b}} - \text{diag}(\mathbf{s})\mathbf{R}\bar{\mathbf{a}}$. The transform described by \mathbf{s}, \mathbf{R} , and \mathbf{t} is applied to the source mesh, then a new set of closest points is found for the next iteration. The process is repeated until convergence, at which point the source mesh has been tightly fitted to the target mesh.

5.4 Ranking Fits

Once the source rigs have been fitted to the target mesh, we need to evaluate the quality of the fit for each, in order to identify the best choices from the database. Our scoring function is a weighted combination of the following:

Normalized ICP distance: Sum of the distance between all closest-point pairs between the source and target mesh, divided by the number of such pairs. That is, where N is the number of closest-point pairs:

$$\sum_i \frac{\|\mathbf{a}_i - \mathbf{b}_i\|}{N} \quad (7)$$

Joint locations: Squared distance between tagged joint locations in the source mesh and target mesh

Automatic correspondences: Number of automatic correspondences generated after alignment, see section 6.1

Since the above calculations do not have the same units, we determined the weights empirically. Weights were chosen once for the database, and reused for every target mesh; ours were 1.0, 0.01, and 0.05 respectively, assuming meters as distance units. If desired, symmetrical body parts can be scored as a pair to ensure that both are taken from the same source rig.

5.5 User Interface

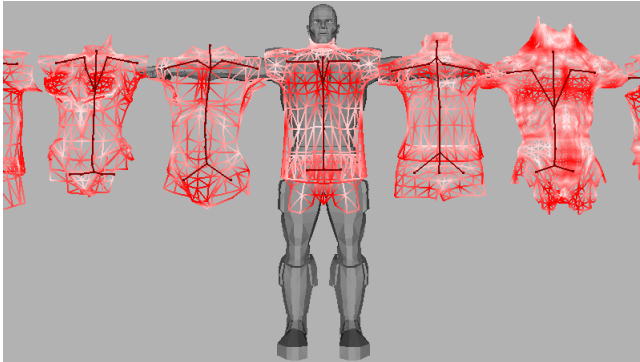


Figure 4: Selecting source meshes from the database. Here, the user is presented with a ribbon of choices for the torso, which are colored according to their distance from the target mesh.

After the partial source rigs from the database have been assigned scores, a sorted list is generated for each type of body part, and the user can select which sources they want to transfer onto the target mesh. This is presented to the user as a ribbon of possible selections, with each source mesh color-coded by mesh distance to help visually identify the best fits (Figure 4). The best ranked fit for each body part is selected by default, but the user is free to flip through the list and pick whatever source rig they like, with the recognition that lower ranked sources will likely not perform as well during the skin transfer phase.

If the user places a joint in the wrong location while specifying target mesh tags (Section 5.1), or if the deformation simply doesn't look the way it was intended, we also give the user the ability to manually reposition joints. This can help in cases where proper joint locations are difficult to determine due to the lack of distinguishing features of the mesh, such as in characters wearing long, straight pants. This repositioning is done using the same method described in Section 5.2, and allows the user to quickly correct misplaced joints.

6 Skin Transfer

With the source rigs selected and fitted, we build a full skeleton for the target mesh by attaching the source skeletons at the seam joints. If two body parts disagree on the location of a joint, they are averaged to minimize discrepancies. Then, we can move on to transferring the skinning information from the sources onto the target mesh.

Transferring skin weights can be viewed from a variety of perspectives, such as a scattered data interpolation problem [Wendland

2004], a coating transfer problem [Sorkine et al. 2004], or a deformation transfer problem [Sumner and Popović 2004]. After trying several strategies, we found that a two-phase approach works best. To summarize, the skin transfer process begins by defining a set of correspondences (not necessarily dense) between the target mesh and the source meshes. These correspondences are used to set up a system of equations that attempts to make the target mesh respond to joint actuations the same way that the source meshes do. Solving this system reconstructs the skinning information on the target mesh, but only at correspondence locations; the rest is filled in by diffusing the results over the surface of the mesh. This method allows deformations to be transferred in select regions, then have the result nicely blended together over the whole target mesh.

Since we are working with real character data, the meshes we use have several pathologies. Each one is typically made of several separate connected components, which are expected to move together despite not being attached. Holes, self-intersection, degenerate triangles, non-watertightness, and other such mesh issues are common. Most problematic is non-manifold geometry: some regions of our meshes simply cannot be mapped to \mathbb{R}^2 . To preserve the user's ability to edit the character, we cannot fix the mesh or reextract one from an implicit surface [Shen et al. 2004]; we are forced to keep the mesh as is.

6.1 Defining Correspondences

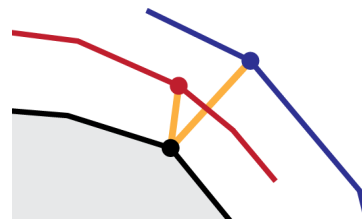


Figure 5: One-to-many correspondences. Each vertex on the target mesh (black) can have a correspondence a location on each source mesh (red and blue). Each correspondence is an instruction for the target mesh vertex to “move like” the corresponding points on the source mesh.

Skin transfer begins by picking the correspondences. These can be one-to-many: each vertex on the target mesh can have a correspondence to a location on every source mesh (Figure 5). Multiple correspondences frequently occur near seam joints, where the fitted body parts are likely to overlap. Source mesh locations need not be vertices; if they occur in a triangle, their properties can be interpolated from the vertices.

Correspondences define regions where the skinning information from the source meshes will be transferred to the target mesh. This does not mean that correspondences need to be dense; leaving out correspondences in a region of the mesh is effectively an instruction to “fill in the blank” with diffusion later. Manually defining correspondences would take far too much time, so we provide some methods of quickly specifying them *en masse*.

The first method is totally automatic. Each vertex on the target mesh finds its closest point on every source mesh, and if the distance between the two points is less than a small threshold (1 cm for a 1.75 m tall character), a correspondence is automatically created. This will generate many correspondences in regions where the target and source meshes line up well or pass through each other. In our database, the density of vertices in the mesh is typically greatest around the joints, which naturally leads to a large population of automatic correspondences in these regions.

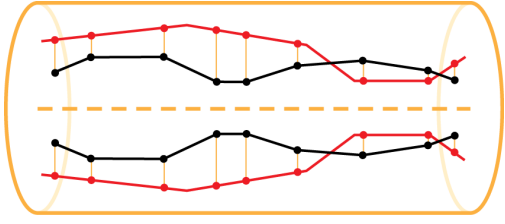


Figure 6: Cross-section of specifying correlations via cylinder. Rays are cast from the centerline of the cylinder (orange) through vertices on the target mesh (black) to find correspondences on the source mesh (red). Source mesh skin weights are interpolated from the nearest vertices.

The next method is semiautomatic. The user drags a cylinder around a portion of the character, and every target mesh vertex within that cylinder has a ray cast through it from the centerline (see Figure 6) onto the source meshes to specify correspondences. To help the user specify these cylinders, the user interface will automatically use the closest bone as a centerline. This method allows the user to quickly specify large numbers of correspondences in simple regions of the target mesh, covering the areas missed by automatic correspondences.

Finally, if all else fails, correspondences can be made manually. It was not necessary to resort to manual correspondences for any of the examples in this paper, but the option is there should the need arise. Users are free to delete any correspondences as well.

6.2 Solving Correspondence Transfer

Each correspondence is effectively an instruction to make a vertex of the target mesh “move like” the associated locations on the source meshes. Unfortunately, copying skin weights directly can introduce noticeable shear in the resulting target mesh deformation, due to differences in the positions of the two corresponding points. Instead, we use this intuition to build an optimization problem that, for each correspondence, matches the response of the target mesh vertex to those of the source mesh locations when any relevant joint is actuated. Each correspondence is treated independently, leading to very small problems that can be solved efficiently.

Formally, for each correspondence, we would like to solve the following:

$$\min \sum_i \left\| \frac{\partial \mathbf{t}(\mathbf{q})}{\partial q_i} - \sum_k \eta_k \frac{\partial \mathbf{s}_k(\mathbf{q})}{\partial q_i} \right\|^2 \quad (8)$$

where \mathbf{q} are the rotational degrees of freedom of the skeleton, $\mathbf{t}(\mathbf{q})$ is the target vertex skinning function, $\mathbf{s}_k(\mathbf{q})$ is the source location skinning function for source mesh k , and η_k is a weighting factor for the k th source location on the target vertex. The η_k factors (described shortly) are used to balance the influences of multiple source meshes on the single target vertex. When using LBS for skinning, the skinning function for the target mesh vertex is:

$$\mathbf{t}(\mathbf{q}) = \sum_j w_j^t \mathbf{T}_j(\mathbf{q}) \mathbf{v}^t \quad (9)$$

where $\mathbf{T}_j(\mathbf{q})$ transforms a bind pose vertex into the frame attached to bone j , w_j^t is the skin weight for bone j on the target mesh, and \mathbf{v}^t is the corresponding vertex in the target mesh. The same skinning function is used for the source meshes \mathbf{s}_k , with the appropriate

weights and locations. Substituting LBS into the optimization problem above:

$$\min \sum_j^t \sum_i \left\| \sum_j w_j^t \frac{\partial \mathbf{T}_j(\mathbf{q})}{\partial q_i} \mathbf{v}^t - \sum_{j,k} \eta_k w_j^{s_k} \frac{\partial \mathbf{T}_j(\mathbf{q})}{\partial q_i} \mathbf{v}^{s_k} \right\|^2 \quad (10)$$

$$\sum_j w_j^t = 1 \quad (11)$$

$$w_j^t \geq 0 \quad (12)$$

We balance the influences of the various source meshes by means of η_k , which we define as the relative skin weight contribution of a particular source mesh k against the contributions of all attached source meshes (see equation 13). Recall that the skin weights on a body part fade out as they approach a seam joint (Section 4). Since the input source rigs are partial, this weights the source meshes or each correspondence by how much deformation information they can contribute to the target vertex, giving us nice blending in the regions around seam joints.

$$\eta_k = \frac{\sum_j w_j^{s_k}}{\sum_{j,k} w_j^{s_k}} \quad (13)$$

By softening the weight normalization constraint (11), the optimization problem described above becomes a non-negative linear least squares (NNLS) problem [Lawson and Hanson 1974]. NNLS shows up commonly in skinning (see [Kavan et al. 2009] and [James and Twigg 2005]), and can be efficiently solved [Cantarella and Piatek 2007].

6.3 Diffusion

Solving for correspondences transfers skinning information from the source rigs to the target rig, but only for a subset of the vertices in the target mesh. To fill in the rest, we treat the known vertices as boundary conditions and diffuse them over the rest of the mesh. This technique has been used several times in the past to fill in unknown skinning information, such as [Baran and Popović 2007] and [Weber et al. 2007].

The main idea is to use a discretization of the Laplacian operator over the surface of the mesh (Δ) to set up a Poisson equation (14) for the scalar field of skin weight values for each bone (w_k). With known boundary conditions, these systems are linear and sparse, and can be efficiently solved. The solutions are nice, smooth harmonic functions that interpolate the boundary conditions across the mesh.

$$\Delta w_k = 0 \quad (14)$$

Unfortunately, the mesh quality problems described at the beginning of this section complicate matters. Holes and degenerate triangles preclude the use of standard cotangent weights [Meyer et al. 2002] for Δ , and non-manifold geometry makes it impossible to define a proper two-dimensional Laplacian operator over the surface in the first place. Furthermore, the separate connected components which make up the target mesh are usually *intended* to move as if they were attached, despite the lack of any actual connection.

$$\Delta_{ij} = \begin{cases} 1 & \text{if } i = j \\ -d_{ij}/D_i & \text{if } j \in \mathcal{N}_i \\ 0 & \text{otherwise} \end{cases} \quad (15)$$

$$d_{ij} = (\|\mathbf{v}_i - \mathbf{v}_j\| + \epsilon)^{-1} \quad (16)$$

$$D_i = \sum_{j \in \mathcal{N}_i} d_{ij} \quad (17)$$

We handle these problems by defining Δ using only adjacency information (15), where \mathcal{N}_i is the 1-ring neighborhood around vertex i . The weights are determined by inverse distance between adjacent vertices, with a small mollification constant in the denominator to avoid singularities. To stitch together connected components, vertices that are on two separate components but within a small distance (1 cm for a 1.75 m tall character) are connected to each other’s neighbors for the duration of the diffusion. This allows skinning information to flow between the surfaces, closing any gaps that would occur if the components were solved separately.

The solved correspondences are added to the system as boundary conditions (as in [Sorkine 2006]), producing an overconstrained problem which we solve using least squares. The skin weight map for each bone is solved in this fashion, then the results are normalized to ensure validity.

7 Results and Discussion

We assembled a set of 14 character rigs for testing our system, with a wide variety of proportions and features. Each character was rigged and skinned by an artist, then automatically broken down into 5 body parts as described in Section 4, forming a database of 70 partial source rigs. The same set of character rigs was stripped of all skinning information and used as target meshes for input into our rigging system. To prevent trivial solutions, we never allow a target mesh to be rigged using body parts extracted from its original rig.

We have combined the entire system workflow into an interactive program that allows the user to experiment with rigging the target meshes. When the user loads a new target mesh, the system scans through the database and fits all the body parts to it (Section 5), a linear-time process which takes about 10 seconds for our database. After this point, all the operations in the program are virtually instantaneous; the skinning transfer process (Section 6) takes a fraction of a second. All timing information was obtained on a 2.2 GHz Intel Core 2 Duo based laptop.

7.1 Automatic Results

The standard way for a character artist to begin skinning a character is to create a skeleton, then perform a “smooth bind” operation. The software will use a set of distance and blending heuristics to generate a weight map for that skeleton on the target mesh, which the artist will then tweak and adjust until satisfied with the result.

Our system can be used in a fashion similar to smooth bind to produce a baseline weight map for further editing. Instead of specifying the entire skeleton, the user specifies joints in the target mesh as described in Section 5.1, then lets the rest of the procedure happen automatically. Part selection, scaling and stretching, correspondence definition and solution, and diffusion are all completed with no user input, and the whole process takes about 10 seconds (as mentioned above). While not precisely the same, this procedure is roughly equivalent in time and effort to performing a smooth bind.

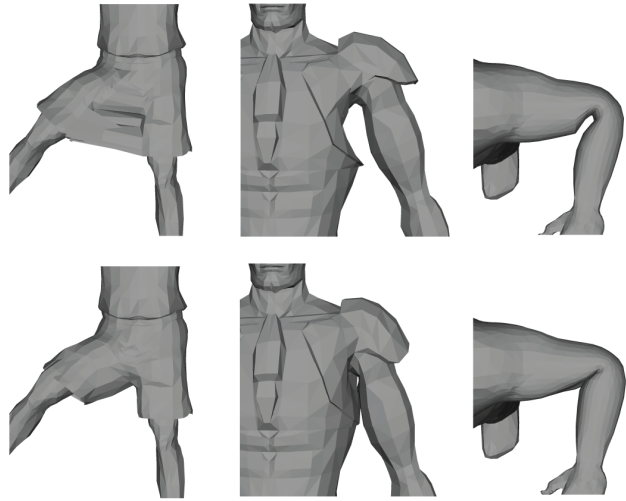


Figure 7: *Smooth bind comparison. Top is Maya’s smooth bind, bottom is our automatic method. Smooth bind has difficulty separating layers of cloth and causes serious deformation artifacts around joints.*

The results of the above procedure are compared against Maya’s smooth bind feature in Figure 7. Our system matches or exceeds the quality of smooth bind in all cases, demonstrating less undesired deformation around joints and better handling of layered surfaces such as clothing. An artist using this tool would spend less time fixing errors introduced in the initial skinning estimation, and thus would spend less time on the overall skinning process.

Whenever LBS is used, deformation artifacts are inevitable; much of the time an artist takes skinning a character is spent hiding these artifacts and making them less distracting. Figure 7 clearly shows the benefit of reusing existing skinning information: the artist’s hard work removing displeasing deformations carries over to the target mesh. Furthermore, none of the target meshes we tried were automatically rigged using arms, legs, and a torso from the same source rig. This indicates that giving the system the flexibility to mix and match pieces from several source rigs allows it to achieve better fitting and transfer overall.

7.2 Manual Results

Once a baseline weight map is in place, the artist must modify it such that the model deforms in exactly the way it is intended to. The traditional tool used for this process is a skin weight brush, which allows the user to select a bone and paint skin weights for that bone directly onto the target mesh. As intuitive as the painting metaphor is, a visible map of skin weights gives little insight into how a particular joint will deform while moving, making the skin weight brush a difficult tool to master. Artists typically spend hours toggling between painting weight maps and flexing joints to test the deformation before converging on a final result.

Our system sidesteps the necessity for most of this tweaking by adapting skin weights from professionally-created rigs. However, no two characters will match up perfectly, and the output from the automatic procedure will sometimes not be exactly what the user had in mind. As described in the previous sections, we have included a suite of tools (Sections 5.5 and 6.1) that can help the user refine the transfer by picking body parts, modifying joint locations, and specifying correspondences.

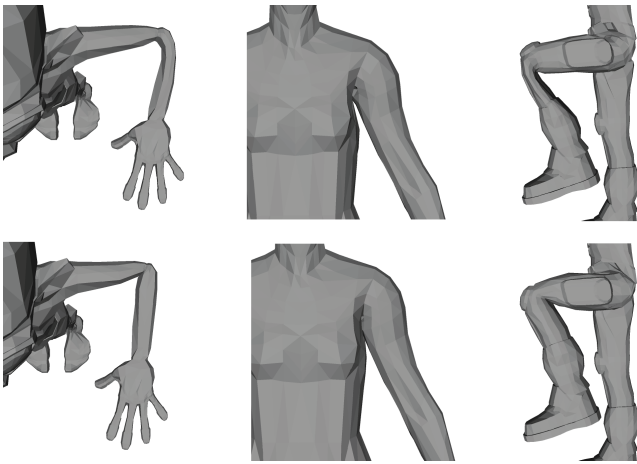


Figure 8: Fixing transfer problems by user interaction. Top is the automatic result, and bottom is the same after a user edit. The character on the left was corrected by moving the elbow joint further up the arm. The middle character had slight deformations around the shoulder, which were corrected by adding more correspondences. The character on the right was fixed by selecting a better leg from the source rig database.

Figure 8 shows several examples of using these tools to improve upon the automatic results. On the left, the location for the elbow joint is slightly off, leading to artifacts in the elbow. This is fixed by manually moving the elbow joint to the proper location. In the middle, artifacts around the shoulder are fixed by specifying more correspondences in the torso and shoulder region. On the right, the strange leg deformations are fixed by picking an alternate source leg from the database. By using these tools in combination, users can quickly improve the quality of the skin transfer. In our experience, roughly two minutes of user interaction in our system is enough to make most rigs visually indistinguishable from an artist-created version.

Figure 9 shows some failure cases, comparing characters rigged in our system against the artist-created originals. In these cases, after about two minutes of work the user was unable to remove the remaining deformation artifacts using the tools in our system. The primary cause for this problem is a lack of appropriate body parts in the database, and can be mitigated by adding a greater variety of source rigs to the database.

If this is unfeasible, the user still has recourse. Unlike many other skinning methods, the output of our system is a standard skeleton and skin weight map representation, which can be edited in any modeling and animation software. If the user can not quite achieve the desired deformations within our system, it is still possible to export the results and use them as a starting point for further tweaking.

7.3 Limitations

It is possible to create situations in which our system will fail to create a satisfactory rig, for a variety of reasons. First, the system is only as good as its database. It is capable of performing a number of adaptations to make the sources fit the target, but if no good source rig exists for a particular character, then the results will always be poor. Second, the skin transfer results are highly dependent on a good set of correlations. If correlations are too sparse, then the diffusion step will fill in large sections of the skin weight map automatically, leading to deformations that tend to look rub-

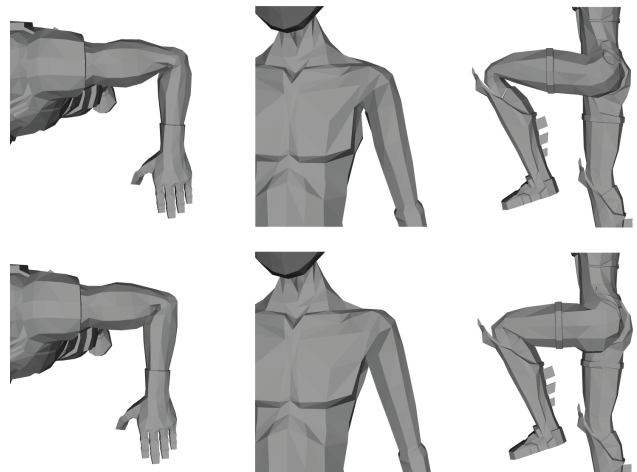


Figure 9: Failure cases. Top are results from our system after about two minutes of user interaction, bottom are the original artist-created skins.

bery. However, we provide a number of tools to help define correspondences (Section 6.1) and mitigate this problem. Third, since we do not retessellate the target mesh, the skin weight transfer is susceptible to undersampling. We reproduce detail from the source mesh deformation as faithfully as possible on the target mesh, but low resolution targets will inevitably suffer from aliasing artifacts.

8 Conclusion

In this work, we have presented a system capable of transferring rigging and skinning information from disparate sources onto a single target rig. We have described a fitting procedure for tailoring source meshes to match a target, and a skinning transfer method capable of blending the influences from several source meshes at once. Using these tools, we have shown significant improvement over standard smooth bind results, and the ability to produce high-quality rigged and skinned characters in a matter of minutes.

Looking forward, one immediate improvement that can be made to our system is transferring rigging information beyond just the skeleton. Character rigs often include elaborate controls, FK and IK handles, deformers, collision volumes and more. Our system could be extended to handle this extra data as well, but we have not yet written exporters and importers to do so. In the future, our system may also be combined with facial rig retargeting work (like [Orvalho 2007]) to provide a complete rigging system.

As a final note, the fact that our relatively small database was able to successfully rig virtually all of our characters may suggest that the space of useful character rigs is not very large, at least within the domain of a single type of character (e.g. bipeds). It may be possible in the future to use a database of source rigs to extract a parameterized representation of skinning information that could be applied to any character.

Acknowledgements

We thank Autodesk and the artists who made their work available on <http://www.creativecrash.com> and <http://www.turbosquid.com> for the source rig data. We also owe thanks to Andrew Floren, for writing our 3ds max exporter. This work was supported by an NDSEG Graduate Research Fellowship.

References

- AUJAY, G., HÉTROUY, F., LAZARUS, F., AND DEPRAZ, C. 2007. Harmonic skeleton for realistic character animation. In *Proceedings of Eurographics/ACM SIGGRAPH Symposium on Computer Animation 2007*, ACM Press, 151–160.
- BARAN, I., AND POPOVIĆ, J. 2007. Automatic rigging and animation of 3d characters. In *Proceedings of ACM SIGGRAPH 2007*, ACM Press.
- CANTARELLA, J., AND PIATEK, M., 2007. tsnnls: A solver for large sparse least squares problems with non-negative variables.
- CGCHARACTER, 2005. Absolute character tools 1.6. http://www.cgcharacter.com/act_pro16.html.
- DU, S., ZHENG, N., YING, S., YOU, Q., AND WU, Y. 2007. An extension of the icp algorithm considering scale factor. In *Proceedings of the IEEE International Conference on Image Processing, ICIP 2007*, vol. 5, 193–196.
- JAMES, D. L., AND TWIGG, C. D. 2005. Skinning mesh animations. *ACM Transactions on Graphics (SIGGRAPH 2005)* 24, 3.
- JOSHI, P., MEYER, M., DEROSE, T., GREEN, B., AND SANOCKI, T. 2007. Harmonic coordinates for character articulation. In *Proceedings of ACM SIGGRAPH 2007*, ACM Press.
- JU, T., ZHOU, Q., VAN DE PANNE, M., COHEN-OR, D., AND NEUMANN, U. 2008. Reusable skinning templates using cage-based deformations. In *Proceedings of ACM SIGGRAPH Asia 2008*, ACM Press.
- KAVAN, L., AND ZARA, J. 2005. Spherical blend skinning: A real-time deformation of articulated models. In *2005 ACM SIGGRAPH Symposium on Interactive 3D Graphics and Games*, ACM Press, 9–16.
- KAVAN, L., COLLINS, S., ZARA, J., AND O’SULLIVAN, C. 2008. Geometric skinning with approximate dual quaternion blending. ACM Press, New York, NY, USA, vol. 27, 105.
- KAVAN, L., COLLINS, S., AND O’SULLIVAN, C. 2009. Automatic linearization of nonlinear skinning. In *2009 ACM SIGGRAPH Symposium on Interactive 3D Graphics and Games*, ACM Press, 49–56.
- KIN-CHUNG AU, O., TAI, C., CHU, H., COHEN-OR, D., AND LEE, T. 2008. Skeleton extraction by mesh contraction. In *Proceedings of ACM SIGGRAPH 2008*, ACM Press.
- KRY, P., JAMES, D., AND PAI, D. 2002. Eigenskin: real time large deformation character skinning in hardware. In *Proceedings of Eurographics/ACM SIGGRAPH Symposium on Computer Animation 2002*, ACM Press, 153–159.
- LAWSON, C., AND HANSON, R. 1974. *Solving Least Squares Problems*. Prentice Hall.
- LEWIS, J., CORDNER, M., AND FONG, N. 2000. Pose space deformation: A unified approach to shape interpolation and skeleton-driven deformation. In *Proceedings of SIGGRAPH 2000*, ACM Press / ACM SIGGRAPH, New York, Computer Graphics Proceedings, Annual Conference Series, ACM, 165–172.
- LIU, P., FU-CHE WU, MA, W., LIANG, R., AND OUHYOUNG, M. 2003. Automatic animation skeleton construction using repulsive force field. *Pacific Conference on Computer Graphics and Applications* 0, 409.
- MEYER, M., DESBRUN, M., SCHRÖDER, P., AND BARR, A. H. 2002. Discrete differential-geometry operators for triangulated 2-manifolds. In *Proceedings of the 2002 International Workshop on Visualization and Mathematics (VisMath)*.
- MOHR, A., AND GLEICHER, M. 2003. Building efficient, accurate character skins from examples. In *Proceedings of ACM SIGGRAPH 2003*, ACM Press.
- ORVALHO, V. 2007. *Reusable Facial Rigging and Animation: Create Once, Use Many*. PhD thesis, Universitat Politècnica de Catalunya.
- POIRIER, M., AND PAQUETTE, E. 2009. Rig retargeting for 3d animation. In *Graphics Interface 2009 Conference Proceedings*, 103–110.
- SCHEEPERS, F., PARENT, R., CARLSON, W., AND MAY, S. 1997. Anatomy-based modeling of the human musculature. In *Proceedings of SIGGRAPH 1997*, ACM Press / ACM SIGGRAPH, New York, T. Whitted, Ed., Computer Graphics Proceedings, Annual Conference Series, ACM, 163–172.
- SHEN, C., O’BRIEN, J. F., AND SHEWCHUK, J. R. 2004. Interpolating and approximating implicit surfaces from polygon soup. In *Proceedings of ACM SIGGRAPH 2004*, ACM Press, 896–904.
- SORKINE, O., LIPMAN, Y., COHEN-OR, D., ALEXA, M., RÖSSL, C., AND SEIDEL, H.-P. 2004. Laplacian surface editing. In *Proceedings of the Eurographics/ACM SIGGRAPH Symposium on Geometry Processing*, Eurographics Association, 179–188.
- SORKINE, O. 2006. Differential representations for mesh processing. *Computer Graphics Forum* 25, 4, 789–807.
- SUMNER, R., AND POPOVIĆ, J. 2004. Deformation transfer for triangle meshes. In *Proceedings of SIGGRAPH 2004*, ACM Press / ACM SIGGRAPH, Computer Graphics Proceedings, Annual Conference Series, ACM, 399–405.
- TERAN, J., SIFAKIS, E., IRVING, G., AND FEDKIW, R. 2005. Robust quasistatic finite elements and flesh simulation. In *Proceedings of Eurographics/ACM SIGGRAPH Symposium on Computer Animation 2005*, ACM Press.
- WANG, R., PULLI, K., AND POPOVIĆ, J. 2007. Real-time enveloping with rotational regression. In *Proceedings of ACM SIGGRAPH 2007*, ACM Press.
- WEBER, O., SORKINE, O., LIPMAN, Y., AND GOTSMAN, C. 2007. Context-aware skeletal shape deformation. *Computer Graphics Forum (Proceedings of Eurographics)* 26, 3.
- WENDLAND, H. 2004. *Scattered Data Approximation*. Cambridge Monographs on Applied and Computational Mathematics.
- WILHELMS, J., AND GELDER, A. 1997. Anatomically based modeling. In *Proceedings of SIGGRAPH 1997*, ACM Press / ACM SIGGRAPH, New York, T. Whitted, Ed., Computer Graphics Proceedings, Annual Conference Series, ACM, 173–180.
- ZISSNER, T., SCHMIDT, J., AND NIEMANN, H., 2005. Point set registration with integrated scale estimation.