As-Rigid-As-Possible Character Deformation Using Point Handles

Zhiping Luo(✉), Remco C. Veltkamp, and Arjan Egges

Utrecht University, Utrecht, The Netherlands
{Z.Luo,R.C.Veltkamp,J.Egges}@uu.nl

Abstract. In this paper, we present a versatile, point handles based character skinning scheme. Point handles are easier to design and fit into an object's volume than a skeleton. Moreover, with a conventional blending technique such as linear blending, point handles have been successfully demonstrated to handle stretching, twisting, and supple deformation, which are difficult to achieve with rigid bones. In the context of only blending, limbs, however, are not bent rigidly with point handles, limiting the space of possible deformations. To address this, we propose a method that automatically recovers the local rigidities of limbs via minimizing a surface-based, nonlinear rigidity energy. The minimization problem is subjected to the positions of a set of point handles' proximal vertices. The positions fitting point transformations are computed by linear blend skinning, leading to speedups of the minimization in particular for large deformations. The use of nonlinear energy also allows versatile posing by intuitively selecting which point handles provide their proximal vertices on-the-fly. The degrees of freedom in modeling user constraints are reduced, and the skinning process is automated by relevant functionalities included in our scheme. The effectiveness of our scheme is demonstrated by a variety of experimental results, showing that the scheme could be an alternative to skeletal skinning.

1 Introduction

To date, skeletal skinning has dominated the practical usages in deformable character animation due to its simplicity and efficiency. In this skinning scheme, the skeleton is a hierarchy of rigid bones referred to as *handles* inside the character, and the skin is moved as an explicit function of handle transformations.

A manual pipeline of skeleton-based character rigging mainly consists of three sequential phrases: first, designing and fitting a skeleton into the character body, and second, painting the influence weights per bone, and finally specifying the transformations of a set of bones. Before painting, usually a method that automatically computes the weights is applied in order to reduce the time, as in this context artists only need to adjust some of the weights. There are several cases where the number of corrections is reduced. First, the computed weights are good enough. Second, artists can easily determine the influencing bone(s) of a body segment so that they will intuitively repaint the regions assigned with

G. Bebis et al. (Eds.): ISVC 2015, Part I, LNCS 9474, pp. 57–70, 2015.
DOI: 10.1007/978-3-319-27857-5_6

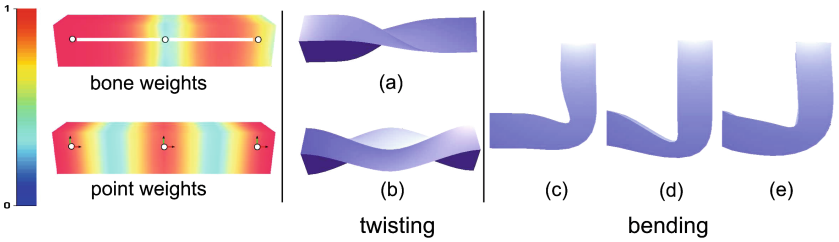

Fig. 1. The weights are visualized using a jet colormap. The white chain denotes the skeleton and the coordinate frames denote point handles. Point weights vary over the region between two point handles, while bone weights only vary near a joint. **Twisting**: twisting the middle joint (point handle), skeleton-based LBS twists the shape only near the middle joint (a), while LBS with point handles produces interesting twisting spread over the entire shape (b). **Bending**: skeleton-based LBS produces a rigid bending (c), while LBS with point handles fails (d). Alternatively, for a rigid bending point handles can be placed at bone centers if one already knew the characteristics of point weights. However, the intuitiveness of specifying transformations is lost. Our scheme aims to approximately recover the local rigidities of a region between two point handles (e).

improper weights. At last but not least, the rotation center of each bone is correct enough, as artists will transform the bones for animation previewing, which is necessary to detect problematic regions and would be repeated. A common solution is to fit a skeleton to the character body and anatomically collocate it with the skin. However, this solution is cumbersome, as transforming a joint often shifts its incident bone(s) outside the body due to the skeleton hierarchy.

The hierarchical structure also complicates skeleton-based automatic skinning. On the one hand, automatic skeleton extraction [1] does not guarantee that the resulting line segments lie inside the object's volume. On the other hand, embedding an existing skeleton inside a shape is a difficult problem. Although well embedded skeletons have been produced, they were archived by solving a continuous optimization of joint locations specific to humanoid shapes [2], or a nonlinear optimization where users provide joint location hints [3].

In contrast to the skeleton, point handles are easy to place inside an object's volume, and each point handle is independently positioned without influencing the positioning of other point handles. Moreover, point handles with associated weight functions (e.g. [4]) are highly suitable for handling stretching, twisting [5], and supple skin regions [4], which are difficult to achieve by linear blend skinning (LBS) with only a scalar weight function per bone. However, LBS with point handles are unsuitable to bend limbs rigidly if the point handles are placed at joint locations so that they are rigged as intuitively as joints. In this paper, the influence weights computed for point handles are referred to as *point weights*, and the weights computed for bones are referred to as *bone weights*. As Fig. 1 illustrates, interesting twisting is spread over the entire shape by LBS with point weights, while LBS with bone weights bends a shape rigidly. This is because point

weights by construction vary over the shape more than bone weights which only vary around joints.

Favoring the aforementioned flexibility of point handles, we seek a fully point handles based skinning scheme that also produces rigid bending desired for articulated deformable characters. As a result, this novel scheme will richly expand the space of deformations possible with LBS with only per point handle scalar weight functions.

To address the limitation of bending, we propose a method that automatically recovers the local rigidities of limbs via minimizing a surface-based, as-rigid-as-possible deformation energy. The minimization problem is subjected to deformed positions of a set of point handles' proximal vertices, computed by LBS with point weights. As point handles are placed exactly at joints which are located where they would be in a real world skeleton, a region between two point handles deforms as-rigidly-as-possible as a rigid limb by solving the constrained minimization problem. Point handles are not in a hierarchical structure as rigid bones, hence inverse kinematics or motion capture techniques cannot be applied to facilitate the input of transformations. Instead, their transformations have to be manually specified through a two-dimensional interface common to animation authoring. To reduce the degrees of freedom, a functionality inferring rotations from translations is developed. So only translations, easily and intuitively prescribed by moving the mouse, are required to users. Compared to automatic skeletal skinning, automatic skinning with point handles is easily achieved based on a mesh-segmentation method.

Our scheme has shown advantages by a variety of experimental results:

i. It provides a cheap and robust way of producing rigid bending, and maintaining shape details, in terms of user interactions and computational time.
ii. It allows versatile posing by adjusting the positional constraints of the as-rigid-as-possible deformation energy on-the-fly.
iii. It retains the intuitiveness of rigging a skeletal character by placing point handles at joint locations, and the simplicity of LBS.

2 Previous Work

Linear blend skinning (LBS) with a hierarchy of rigid bones gives rise to *joint-collapse* artifacts near joints such as shoulders, wrists, or even elbows if they exhibit large rotations. To cope with such artifacts, one possibility is to expand the expressive power of LBS by either supplying additional weights per bone [6,7], or adding virtual bones expressed as their transformations and weights [8,9]. The new data has to be automatically computed using example poses [5], which often do not exist. Another solution is to linearly blend unit dual quaternions instead of transformation matrices, forming the method called dual quaternion skinning (DQS) [10] which is more computationally expensive than LBS.

It is also possible to reduce the number of LBS artifacts by replacing rigid bones with other types of handles. Works such as [11,12] use curve skeletons to fix joint-collapse. Curved skeletons need new rigging controls that are inconsistent

with the existing rigging pipeline [13]. A lattice of control points, used in free-form deformation [14], is easy to design, but it is unsuitable for the control of concave objects due to its regular structure. A cage [15], as a general control polyhedron loosely enclosing the target surface, has a better match of degrees of freedom to the enclosed geometry, thereby allowing precise area control such as bulging and thinning in regions of interest. However, a control mesh is tedious to establish and manipulate.

Points are perhaps the easiest handles to establish. Existing automatic bone weights methods such as [2] may be trivially adapted to compute the influence weights for point handles. Jacobson et al. [4] proposed a method well suited for computing point weights. Although the method is slow to compute and requires a volume discretization, its resulting weights produce the highest quality deformations [5]. Once good point weights are available, the transformation of a vertex at runtime is computed by linearly blending point transformations. LBS with point handles is suitable for handling stretching, twisting [5], and supple deformation which requires a skeleton with many bones in skeletal skinning. However, it is much less effective to bend limbs rigidly than skeleton-based LBS. Furthermore, LBS, as a closed-form blending technique, has no shape-preserving property. Variational methods [16,17] compute high-quality shape preserving deformations for arbitrary handles at points, but at a greater runtime cost.

Since each of the aforementioned handle types shows its own advantages, a hybrid of two or more handle types has been demonstrated to expand the space of deformations possible with LBS: a hybrid of point and bones [5], and a hybrid of points, cages and bones [4]. Although the flexibility of point handles has been demonstrated, until now a fully point handles based LBS remains to be developed.

3 Skinning with Point Weights

To achieve rigid bending with point handles is difficult. On the one hand, this is because it is not so clear where the deformation point handles should be placed for rigid bending. On the other hand, rigging the point handles for a rigid bending is not as intuitive as rigging a skeleton, in case intuitively and anatomically the point handles are placed between conjoint near-rigid segments. In contrast, tediously and carefully rotating the point handles is required, as point weights vary over the region between two point handles.

Assuming that good point weights are available, it is no doubt that associated weights of the vertices in some distances to each point handle fall into a rather small range close to 1. These vertices form a small region in the vicinity of each point handle. Here, good weights should at least satisfy the properties of locality and smoothness. So in the context of blend skinning, on the one hand the deformation of the proximal region per point handle is nearly rigid. In other words, the weight map per point handle captures the local rigidities of such a region. On the other hand, the region's deformation accurately and intuitively fits the transformation of point handle since the associated weights are large enough.

Our solution is to impose a rigidness regularization to the vertices between two point handles, by solving an energy minimization problem. The objective function is subjected to the positions of proximal vertices per point handle, which are computed in the spirit of LBS. Thus, our framework keeps the simplicity of modeling the user constraints as blend skinning, where users specify the handle transformations rather than to directly prescribe the vertex positions.

3.1 Modeling Framework

Given a skinning model consisting of H point handles, a skin mesh S containing N vertices $\{v_1, ..., v_N\}$ and M triangles $\{t_1, ..., t_M\}$ at rest pose, each vertex v_i is bound to the handles by influence weights expressed as a vector $\mathbf{w}_i = (w_{i,1}, ..., w_{i,H})$. Given the transformations of point handles $(C_1^f, ..., C_H^f)$ at frame f, optimal deformation is computed by solving the minimization problem

$$\begin{aligned} \min \quad & E_s(S') = \sum_{i=1}^{N} \sum_{j \in \mathcal{N}_1(v_i)} \boldsymbol{\omega}_{i,j} \parallel (v_i' - v_j') - R_i(v_i - v_j) \parallel^2 \\ \text{s.t.} \quad & v_k' = v_k \sum_{j=1}^{H} w_{k,j} C_j^f, \quad k \in \mathscr{H}, \end{aligned} \tag{1}$$

where $\mathscr{H}$ is the set of indices of the proximal vertices, $\boldsymbol{\omega}_{i,j}$ are per-edge weights between v_i and each one-ring neighbor $v_j \in \mathcal{N}_1(v_i)$. R_i of vertex v_i is derived from the singular value decomposition (SVD) of the covariance matrix $S_i = \sum_{j \in \mathcal{N}_1(v_i)} (\boldsymbol{\omega}_{i,j} e_{ij} e_{ij}'^T) = U_i \sum_i V_i^T$, and $R_i = V_i U_i^T$ [16], where $e_{ij} = v_i - v_j$ and e_{ij}' is the edge after reconstruction.

The solution consists of two steps. First, compute an initial guess as $v_i' = v_i \sum_{j=1}^{H} w_{i,j} C_j^f$, and approximate the local rotations R_i based on v_i'. Second, new positions v_i' are obtained by minimizing Eq. (1). We follow the same procedures as [16] to compute the partial derivatives w.r.t. v_i'. By setting the partial derivatives to zero w.r.t. each v_i', a sparse linear system of equations is achieved, which can be solved efficiently by the sparse Cholesky factorization. The minimization is iteratively performed by alternately re-computing local rotations and solving the equation until it reaches the number of iterations specified by users.

Closely similar to the surface modeling framework [16], our framework also minimizes the changes of edge lengths to obtain as-rigid-as possible (ARAP) deformations. However, we compute positional constraints in the sprit of linear blend skinning. Thus, users only need to move the point handles without manually selecting a set of vertices on the surface. Also, an initial guess is computed by LBS rather than using the Laplacian surface editing (LSE) [18] method. Since LBS, having a closed-form formula, is much faster than LSE, and can produce a reasonable degrees of local rigidities in bending with point weights. Thus, the number of iterations required to reach a desired rigid bending will be reduced.

Our method is depicted in Fig. 2. Since the objective function is subjected to hard constraints that are computed positions using LBS, discontinuities appear

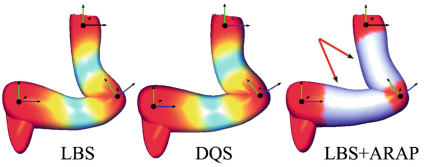

Fig. 2. Point weights are displayed as colors in the two images on the left. Both LBS and DQS with point handles are unsuitable to bend a shape rigidly. Our scheme (LBS + ARAP) automatically refines the region between two point handles to be as-rigid-as-possible via minimizing a rigidity energy constructed over the shape. The minimization problem is subjected to the positions of a set of point handles' proximal vertices (reddish parts of the rightmost image).

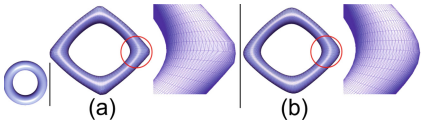

Fig. 3. The torus is deformed. (a) Discontinuities appear around the point handles. (b) With a Laplacian smoothness regularization, such artifacts are suppressed.

in the handle proximity. Such artifacts can be easily suppressed, see Fig. 3, by moving the vertices in the direction of the Laplacian

$$v_{i\_opt}^{'} = v_i^{'} + \frac{1}{\sum_{j\in\mathcal{N}(i)} \boldsymbol{\omega}_{i,j}} \sum_{j\in\mathcal{N}(i)} \boldsymbol{\omega}_{i,j}(v_j^{'} - v_i^{'}), \tag{2}$$

In fact, only the positions of proximal vertices will be re-optimized. For the regions between two point handles, optimal results have been reached due to the rigidness regularization resulting from a solution of Eq. 1.

3.2 Degrees of Freedom

The transformations of point handles are manually specified through a 2D user interface common to animation authoring. Through such an interface, rotations that are necessary to LBS (see Fig. 4), however, are non-trivial and less intuitive to specify relative to translations that are directly proportional to mouse movements. We present methods that automatically infer rotations from user-specified translations.

Given P point handles denoted by $p_i \in \mathbb{R}^3, i \in \{1, ..., P\}$, each point handle has E user-defined incident pseudo-edges, defined as three-dimensional vectors $p_{i,k} = p_i - p_j, k \in \{1, ..., E\}$ at rest pose, and $p_{i,k}^{'} = p_i^{'} - p_j^{'}$ after translations. The rotation between two vectors $p_{i,k}$ and $p_{i,k}^{'}$ is represented using quaternion denoted by $q_{i,k}$. The final rotation denoted by q_i of point handle p_i is an average of all $q_{i,k}$, $k \in \{1, ..., E\}$.

Notice that, unlike the skeleton, the pseudo-edges do not necessarily lie inside the volume and conform to a skeleton structure. In fact, there is a large degree of

freedom on connecting the point handles as long as no one is isolated, such that, for each point handle, the rotation axis is the cross product of the two vectors $p_{i,k}$ and $p'_{i,k}$ defined by the pseudo-edges.

The half-quaternion method is used to calculate the rotation about an axis between two vectors, simply because it saves some square-root calculations and ensures a "shortest arc" solution. For an average of only two quaternions, the well-known spherical linear interpolation (Slerp) can be applied. In the presence of more quaternions, a simple and fast way is to average by a recursion sum algorithm

$$avg(q_{i,E}) = \begin{cases} \frac{1}{E}(q_{i,E} + avg(q_{i,E-1})) & E \geq 2 \\ q_{i,1} & E = 1. \end{cases} \tag{3}$$

This method yields a valid average only if the quaternions are relatively close to each other.

A more accurate method comes at the price of reduced speed. However, because the number of incident edges per point handle is small, the overload is minimal. This method first converts the quaternions $q_{i,k}(w, x, y, z)$ as four-dimensional column vectors $q_{i,k}(x, y, z, w)$ and then build a matrix as

$$\mathbf{M} = \frac{1}{E} \sum_{k=1}^{E} q_{i,k} {q_{i,k}}^T. \tag{4}$$

The final quaternion is the (convert back) eigenvector corresponding to the largest eigenvalue of the matrix. Figure 4 depicts a visual comparison between two methods.

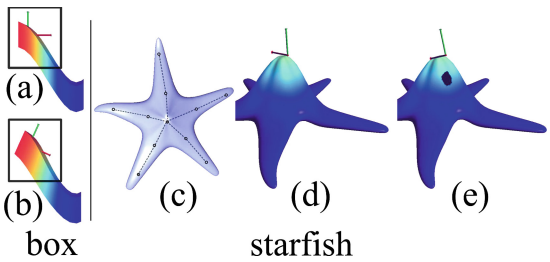

Fig. 4. Box: With the presence of only translation of the transforming point handle (coordinate frame), LBS gives rise to shape distortion (a). Additionally providing a rotation removes the distortion artifact (b). **Starfish**: given the pseudo-edges (dash lines) (c), accompanying rotations are inferred from user-prescribed translations. For the middle point handle, the fast recursion sum method taking about 6 ms yields self-intersection, rendered into a black hole (e). The method based on computing eigenvector of a matrix addresses this limitation (d), though it takes about 16.6 ms.

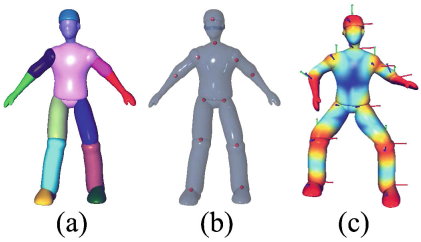

Fig. 5. Segments with different thicknesses, indicated by different colors, are obtained based on SDF values (a). Point handles are automatically placed between conjoint segments (b), and their influence weights, displayed as colors in (c), are computed. The character is posed by translating point handles (coordinate frames). Note that rotations are inferred.

3.3 Automatic Skinning

Point handles are easily placed within the character volume with the help of ray-triangle intersection for instance, and their positions can be anatomically guided to users. For example, the positions of point handles tend to be joint locations in a humanoid character. Targeting a fully automatic skinning scheme, a method is developed to automatically determine the placements of deformation point handles.

In articulated deformable characters, the thickness of a body segment is often different to its conjoint segments, and the skin of such a segment deforms as-rigidly-as-possible. Recall that in our method the region between two point handles is as-rigid-as-possible, thus, for a polygonal mesh, it is natural to place the point handles between conjoint segments with different thickness. The shape-diameter-function (SDF) values [1] measure the diameter of an object's volume in the neighborhood of each vertex on the surface, providing a direct way to detect near-rigid segments. Our method is based on SDF values.

First, a hard-clustering is performed on the raw SDF values to obtain pairwise facet and cluster-id. Second, candidate vertices occupying multiple clusters are picked out if their hosting triangles have a different cluster-id. For each candidate: a ray along the inverse normal is cast from the vertex position, and then a point as an intermediate result is picked on the line of the ray according to half of the associated SDF value. This ensures that all picked points lie inside the volume. Finally, the position of a point handle is the center of a spatially grouped intermediate points, guaranteeing that all point handles are inside the volume.

Once the skin is bound to deformation point handles by influence weights computed using an automatic weights method such as [4], the character is ready for animation. Figure 5 depicts our method applied to a humanoid character.

4 Results and Discussion

Our method was implemented in C++ on a laptop with an Intel Core i7 2.4 Ghz processor and 6 GB memory. The sparse Cholesky solver and two-sided Jacobi

SVD decomposition, provided by the Eigen library [19], were used to solve the minimization formula and to compute rotations R_i, respectively. The cotangent-weight formula [20] instead of uniform weights is used to compute per-edge weights $\boldsymbol{\omega}_{i,j}$, avoiding deformation artifacts [16].

Bone weights are much more intuitive to paint by artists than point weights. However, seeking an automatic scheme is a primary goal in skinning. Thus, to obtain fair comparisons, the influence weights of bones and point handles are computed. Bounded biharmonic weights (BBW) [4] are computed as influence weights for point handles and bones, respectively, because they are smooth, local and shape-aware, and have experimentally produced high-quality deformations [4,5]. Since BBW requires a priori volume discretization, we obtain it by voxelization instead of tetrahedral meshing, avoiding dealing with self-intersection and non-manifoldness. The computation then could be largely increased as usually a voxel grid is denser than a tetrahedral representation, but it is a tolerable, one-time precomputation taking dozens of seconds for a polygonal mesh with reasonable geometric complexity.

Two animation models mainly containing deformations of rigid bending are used to demonstrate the effectiveness of our method. They include a hand model with 33 frames and 1,926 vertices, a human model with 54 frame and 3,557 vertices. Deformations resulting from skeleton-based LBS (LBS + bones) are considered as the ground-truth, as bones are naturally effective for rigid bending. Joint locations are directly used as point handle positions. For rendering of the deformed mesh, vertex normals are recomputed on CPU.

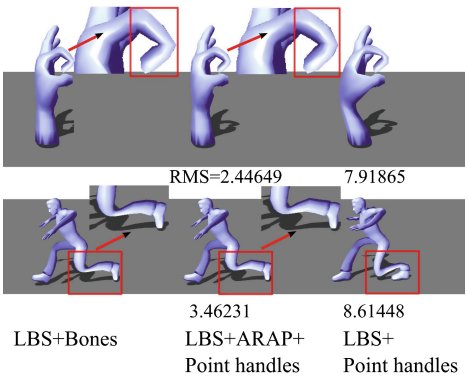

Fig. 6. $l = 0.98$ for the handle model, $l = 0.95$ for the human model, and the number of iterations is 3, giving rise to the results. The numbers below are RMS errors. LBS with point handles fails to bend limbs rigidly. Our scheme gives rise to comparable, visual and quantitative results w.r.t. skeleton-based LBS.

For each point handle, our method needs to determine $v_k, k \in \mathscr{H}$ (recall Sect. 3.1) as positional constraints. A vertex is selected if its largest weight is larger than a threshold l. Note that the associated weights of a vertex consist

of a set of per-point weights, as mostly a vertex is influenced by multiple point handles. Due to the locality offered by BBW, such vertices are well located in the vicinity of point handles.

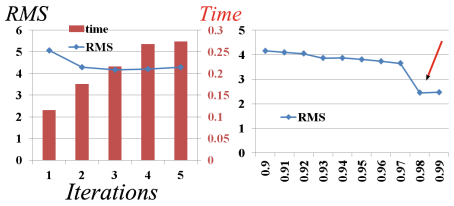

Fig. 7. Left ($l = 0.9$): Computational time (in seconds) increases along with the number of ARAP iterations. The shape is bent more rigidly by increasing iterations, hence mainly the error is reduced along with iterations. By specifying 3 iterations, the lowest error is obtained. **Right** (*iterations* $= 3$): The pattern of the discrete curve shows that mainly the error is reduced by increasing l. In the handle model, mostly, the largest weights of the vertices in the vicinity of a point handle lie in the range from 0.97 to 0.98. Specifying l as a value in this range yields positional constraints, in turn deformations, different to the results generated by other values also in that range. As a result, a sudden drop happens.

An error metric w.r.t. the ground truth is formulated to quantitatively evaluate the methods. The lower the computed value is, the more rigidly the limbs are bent. We first resize the models so that their rest poses are tightly enclosed by a unit cube. Then, the metric is calculated as

$$E_{RMS} = \frac{\sum_{f=1}^{F} \sum_{i=1}^{N} \parallel bv_i^f - pv_i^f \parallel^2}{\sqrt{3NF}}, \tag{5}$$

where F denotes the number of frames, N is the number of vertices, bv^f and pv^f are deformed vertices at frame f resulting from a method with bone weights and point weights, respectively. This form of metric has been reported to be less sensitive to global motions [21].

As Fig. 6 shows, LBS with point handles fails in producing rigid bending. Our method, called *LBS + ARAP + Point handles*, produces better visual and quantitative results.

Relationships between computational time and the number of iterations used to solving the minimization problem, called *ARAP iterations* for short, between E_{RMS} and the number of ARAP iterations, between E_{RMS} and threshold l, are tested on the hand model. Figure 7 reports the results. Though the computational time increases along with the number of iterations, our energy minimization runs three iterations to yield the lowest error. In cases of larger iterations, the shape is bent more rigidly by our scheme than by skeleton-based LBS. So, the error then is slightly increased. This happened as the bone lengths of the hand model are small. Since BBW yields local and shape-aware weights [4,5],

the threshold l is decided intuitively, reducing user errors. As a result, no sharp line is observed in the right subplot.

Discussion. The use of skeleton is known to facilitate the input of bone motions by either motion capture techniques or inverse kinematics. Currently, point handles' counterparts are joints of a corresponding skeletal model in order to maintain an anatomical embedding. Due to this setting, bone transformations were directly reused as point handle transformations in our experiments. As a consequence, the aforementioned techniques facilitating motion input have been investigated in skinned models based on the point handle metaphor, and encouraging results have been obtained.

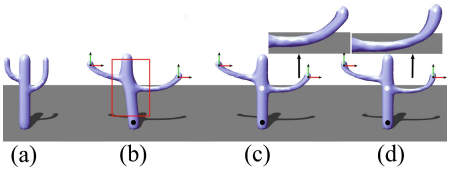

Fig. 8. The cactus at rest pose (a) is skinned with four point handles. A pose is achieved by moving two point handles in branch (coordinate frames) and fixing the one at the base (b). In this case, only these three point handles are used to define positional constraints of ARAP. If one desires locally controlled deformation of a branch, our scheme allows it by additionally enabling the middle point handle (white) to provide its proximal vertices as new constraints on-the-fly (c). LBS also produces such a deformation style (d), but the branches are over-stretched, compared to our scheme, see close ups.

Compelling deformations can be obtained by ARAP (as-rigid-as-possible deformation energy) with few user edits. This has been demonstrated in ARAP-based surface modeling [16] for instance. This advantage can be augmented to reduce the degrees of freedom for certain character postures in our skinning scheme based on point handles. Given a skinned model with many point handles, user specifies only a subset of the degrees of freedom, and the rest deformations are automatically determined using the nonlinear ARAP. This is similar to the method [22] that uses nonlinear, rigidity energies to infer the rest skinning transformations. It is also allowed to adjust the positional constraints of ARAP on-the-fly. As Fig. 8 shows, an extra point handle is enabled to provide its proximal vertices as positional constraints. As a result, resulting deformations are locally controlled if it is desired. In this case, the newly enabled point handle actually serves to occlude the rotation propagations from the moving point handle to the regions far away. As Figs. 8 and 9 show, closed-form blending techniques, such as LBS and DQS, do not support shape-reserving property, but our scheme better maintains the shape locality.

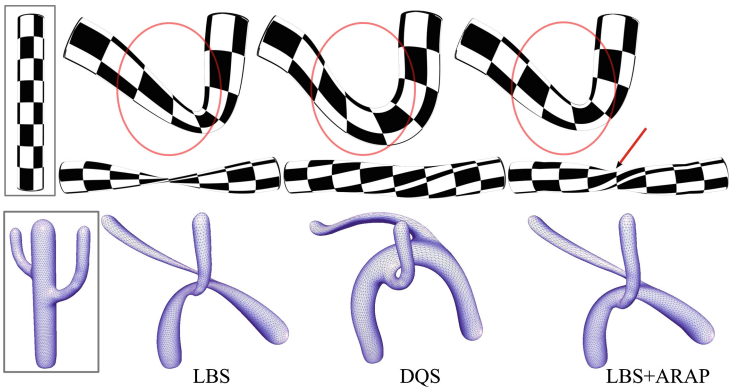

Fig. 9. LBS and DQS do not maintain surface details during deformations, while our method (LBS + ARAP) does. DQS aims at volume preservation, but without considering the realism of posture. For example, the interesting twisting of the cylinder is lost when using DQS. Models in rectangle are in their rest poses.

5 Conclusions

In this paper, we have presented a novel skinning scheme based on point handles, which are much easier to design and embed into the character body than a skeleton. Our method addressed the limitation of bending, when using a conventional blend skinning with point weights, by optimizing the vertex positions using a nonlinear, rigidity energy. We have demonstrated the effectiveness of our method by experimental results of two animation models mainly containing rigid bending. Automatic skinning has been explored in order to reduce the manual work. Also, our method allows for versatile posing and it is robust to large deformations.

Currently, point handles are considered as counterparts of joints. In cases of skinned models with long bones, the mesh part wrapping around a long bone might not be rigid enough when bending by our scheme. This is because we only defines constraints on edge lengths in order to obtain substantial speedups. To overcome this, we plan to develop a method that automatically selects better placements of point handles based on *Schelling points* [23]. To further obtain speedups, we will incorporate existing weight reduction techniques [24,25] to reduce the computational cost of a dense-weight skinning model.

References

1. Shapira, L., Shamir, A., Cohen-Or, D.: Consistent mesh partitioning and skeletonisation using the shape diameter function. Vis. Comput. **24**, 249–259 (2008)
2. Baran, I., Popović, J.: Automatic rigging and animation of 3d characters. ACM Trans. Graph. **26**, 72 (2007)

3. Jacobson, A., Panozzo, D., Glauser, O., Pradalier, C., Hilliges, O., Sorkine-Hornung, O.: Tangible and modular input device for character articulation. ACM Trans. Graph. **33**, 82:1–82:12 (2014)
4. Jacobson, A., Baran, I., Popović, J., Sorkine, O.: Bounded biharmonic weights for real-time deformation. ACM Trans. Graph. **30**, 78:1–78:8 (2011)
5. Jacobson, A., Sorkine, O.: Stretchable and twistable bones for skeletal shape deformation. ACM Trans. Graph. **30**, 165:1–165:8 (2011)
6. Wang, X.C., Phillips, C.: Multi-weight enveloping: least-squares approximation techniques for skin animation. In: Proceedings of the 2002 ACM SIGGRAPH/Eurographics Symposium on Computer Animation, SCA 2002, pp. 129–138 (2002)
7. Merry, B., Marais, P., Gain, J.: Animation space: a truly linear framework for character animation. ACM Trans. Graph. **25**, 1400–1423 (2006)
8. Wang, R.Y., Pulli, K., Popović, J.: Real-time enveloping with rotational regression. ACM Trans. Graph. **26**, 73 (2007)
9. Kavan, L., Collins, S., O'Sullivan, C.: Automatic linearization of nonlinear skinning. In: Proceedings of the 2009 Symposium on Interactive 3D Graphics and Games, I3D 2009, pp. 49–56 (2009)
10. Kavan, L., Collins, S., Žára, J., O'Sullivan, C.: Skinning with dual quaternions. In: Proceedings of the 2007 Symposium on Interactive 3D Graphics and Games, I3D 2007, pp. 39–46 (2007)
11. Yang, X., Somasekharan, A., Zhang, J.J.: Curve skeleton skinning for human and creature characters: research articles. Comput. Animat. Virtual Worlds **17**, 281–292 (2006)
12. Forstmann, S., Ohya, J., Krohn-Grimberghe, A., McDougall, R.: Deformation styles for spline-based skeletal animation. In: Proceedings of the 2007 ACM SIGGRAPH/Eurographics Symposium on Computer Animation, SCA 2007, pp. 141–150 (2007)
13. Kavan, L., Collins, S., Žára, J., O'Sullivan, C.: Geometric skinning with approximate dual quaternion blending. ACM Trans. Graph. **27**, 105:1–105:23 (2008)
14. Sederberg, T.W., Parry, S.R.: Free-form deformation of solid geometric models. In: Proceedings of the 13th Annual Conference on Computer Graphics and Interactive Techniques, SIGGRAPH 1986, pp. 151–160 (1986)
15. Ju, T., Schaefer, S., Warren, J.: Mean value coordinates for closed triangular meshes. ACM Trans. Graph. **24**, 561–566 (2005)
16. Sorkine, O., Alexa, M.: As-rigid-as-possible surface modeling. In: Proceedings of the Fifth Eurographics Symposium on Geometry Processing, SGP 2007, pp. 109–116 (2007)
17. Botsch, M., Sorkine, O.: On linear variational surface deformation methods. IEEE Trans. Vis. Comput. Graph. **14**, 213–230 (2008)
18. Sorkine, O., Cohen-Or, D., Lipman, Y., Alexa, M., Rössl, C., Seidel, H.P.: Laplacian surface editing. In: Proceedings of the 2004 Eurographics/ACM SIGGRAPH Symposium on Geometry Processing, SGP 2004, pp. 175–184 (2004)
19. Guennebaud, G.: Eigen: a c++ linear algebra library, version 3.0 (2011). http://eigen.tuxfamily.org/ (2011). Accessed 22 November 2014
20. Meyer, M., Desbrun, M., Schröder, P., Barr, A.H.: Discrete differential-geometry operators for triangulated 2-manifolds. In: Hege, H.-C., Polthier, K. (eds.) Visualization and mathematics III, pp. 35–57. Springer, Heidelberg (2003)
21. Kavan, L., Sloan, P.P., O'Sullivan, C.: Fast and efficient skinning of animated meshes. Comput. Graph. Forum **29**, 327–336 (2010)

22. Jacobson, A., Baran, I., Kavan, L., Popović, J., Sorkine, O.: Fast automatic skinning transformations. ACM Trans. Graph. **31**, 77:1–77:10 (2012)
23. Chen, X., Saparov, A., Pang, B., Funkhouser, T.: Schelling points on 3d surface meshes. ACM Trans. Graph. **31**, 29:1–29:12 (2012)
24. Landreneau, E., Schaefer, S.: Poisson-based weight reduction of animated meshes. Comput. Graph. Forum **29**, 1945–1954 (2010)
25. Le, B.H., Deng, Z.: Two-layer sparse compression of dense-weight blend skinning. ACM Trans. Graph. **32**, 124:1–124:10 (2013)