Computers & Graphics ▮ (▮▮▮▮) ▮▮▮–▮▮▮

Contents lists available at ScienceDirect

Computers & Graphics

journal homepage: www.elsevier.com/locate/cag

Technical Section

Cage-based deformation transfer

Lu Chen [a], Jin Huang [a,*], Hanqiu Sun [b], Hujun Bao [a]

[a] State Key Lab. of CAD&CG, Zhejiang University, Hangzhou, China
[b] Chinese University of Hong Kong, Shatin, New Territories, Hong Kong

ARTICLE INFO

Article history:
Received 4 May 2009
Received in revised form
7 January 2010
Accepted 21 January 2010

Keywords:
Mesh deformation
Deformation transfer
Green Coordinates

ABSTRACT

We present a cage-based method for transferring animation from a mesh sequence or motion capture data to geometric models in variant representations. To reach the aimed generality, the target model is first embedded into a cage by Green Coordinates interpolation, which preserves the geometric details inherently. The deformation gradient sequences of some user-selected points on the source are then extracted and used as the gradient constraints in the cage to guide the target deformation. The variation of deformation gradients in the cage is minimized to avoid degeneration and ensure smoothness of the result. Position and length constraints can also be applied for more controls. The optimal positions of target cage vertices can be efficiently evaluated by solving a non-linear minimization problem in terms of a few variables. The main advantage of our algorithm is that both source and target can be in a wide range of shapes and deformation representations. Moreover, we can transfer the deformation from multiple sequences onto a single target model.

1. Introduction

To get realistic or natural deformation sequences, people often resort to motion capture, physical simulation, or manual design of skilled artists. It is usually a difficult and time-consuming job in many cases. Therefore, the reuse of data becomes quite appealing.

The ability of handling different geometry representations is also a desired property of the algorithm reuse, which will greatly improve generality and productivity. The source animation can be represented in motion capture data or a mesh sequence. The target geometric representation can be not only manifold meshes, but also non-manifold, multiple-part objects, polygon soup, or even point data. For the application of deformation transfer, how to handle complex models in various representations is still a challenging problem.

To address this problem, we employ the deformation gradients and cage as the transfer intermediate. Deformation gradient on certain point, which is defined as a 3×3 affine transformation, can be extracted from many types of deformation sources. We also use Green Coordinates [14] to embed target model into a cage, which is usually a polygon mesh enclosing the original model. Such space deformation technique can handle models in various representations. Based on these intermediates, we put forward a new cage-based subspace method for deforming the cage with an embedded model using deformation gradient constraints. First, users build several point-to-point correspondence between source and target models. Then, the deformation transfer is done by deforming the cage with target model using deformation gradient constraints, the data of which are extracted from corresponding points of the source sequence.

1.1. Contribution

The main contribution of this paper is a method for transferring animation sequences between various types of source and target models. The advantages of our method include:

Generality: Our method can transfer deformation between a wide range of deformation and model representations. This generality comes from the fact that a wide range of shape representations can be embedded in the cage space and deformation gradients can easily be extracted from various animation sources. Also, our method can even transfer the deformation from multiple sequences to a single target model by the point-to-point deformation gradient correspondence.

Efficiency: Different from linear subspace methods, the Green Coordinates we employed ensure quasi-conformal deformation, which preserves the shape and details inherently [14]. Therefore, pure linear constraints can transfer the deformation from source model with the details preserved. Because Green Coordinates are non-linear in terms of cage vertex positions, the optimization problem is also non-linear. However, it can be solved efficiently without traversing inner sample points to evaluate the gradient of the deformation energy, which is a common problem in previous subspace deformation methods [10,22].

* Corresponding author.
E-mail address: hj@cad.zju.edu.cn (J. Huang).

0097-8493/$ - see front matter © 2010 Elsevier Ltd. All rights reserved.
doi:10.1016/j.cag.2010.01.003

1.2. Related works

A large number of approaches for interactive shape deformation have been proposed during the last decade and some of them have also been used for deformation transfer. We will take a brief overview of the related works in this section.

Surface-based deformation: Multi-resolution approaches can intuitively preserve the local geometric details by the frequency decomposition techniques [13,8]. Since the explicit multi-scale decomposition and reconstruction cannot distribute distortion evenly, methods that optimize gradient-based or Laplacian-based energies can be used to preserve surface details [23,15,24]. However, specifying orientations for each mesh face is inevitable for such linear least square optimization based methods. Particularly, it is difficult to achieve shape preserving under large deformation. This problem was recently solved by Botsch et al. [4], Au et al. [1] and other non-linear optimization methods.

Embedded deformation: Surface-based deformation methods share a common limitation that they require the deforming model to be a manifold. This limitation restricts their applications from complex geometric representations. Space-based methods, having objects embedded in the deforming space, are usually weakly correlated to the geometric representations. Various methods are brought forward using different space expressions and interpolation bases. Radial basis functions (RBFs) are useful tools for smoothly propagating the deformation from control points to embedded models [3,22]. The work of Botsch et al. [5] and many finite element methods discretize the object by linear interpolated cells [17]. Some other methods, developed from the previous free-form deformation methods [19], which have to embed the object in a lattice with the special topology, use a closed polygon mesh (cage) to enclose the deformable model [12,11,10]. Most space-based methods use interpolation bases which are linear to the vertex positions of the control mesh, such as mean value coordinates, harmonic coordinates and barycenter coordinates, etc. And it is hard to get smooth and conformal deformation result with only a few control points, cage vertices or elements. Recently, Lipman et al. [14] presented a method which introduces the face normals into the interpolation bases and makes the deformation function non-linear to the vertex positions of the control mesh. This method ensures high quality conformal deformation results with quite simple cages. The above embedded deformation methods could also be regarded as using reduced deformation models or subspaces to simplify the complex deformation problem by making it independent of the high geometric complexity. This idea of dimensionality reduction was also used in the work of Der et al. [7]. It identifies control parameters of a reduced deformation model with a set of transformation matrices controlling shape deformations. Although this subspace could be constructed automatically and implicitly, it needs a set of deformation examples to span the deformation space, which is hard to be satisfied in our application.

Deformation transfer: The animation can be represented by mesh sequences. For such data, Sumner et al. [21] transferred the deformation from source model to target model by deformation gradients over triangle mesh. This method requires the source and target models both to be manifold triangle meshes, and cannot be applied to other geometric representations (e.g. polygon soup) directly. Motion capture data are another important source of animation. Shi et al. [20] and Zhou et al. [25] demonstrated that it can be used to drive the deformation of manifold triangle meshes. All of the above methods can only handle manifold mesh. To address various representations of target models, we adopt embedded deformation method. In order to handle both mesh sequence and motion capture data in a uniform way, we use gradient constraints to transfer the deformation. The embedding deformation methods employed in the works of Huang et al. [10] and Sumner et al. [22] can be used under this strategy. However, both methods involve a non-linear detail preserving energy in terms of discretization samples (vertex positions of the mesh or deformation gradients and translations on the graph nodes). It is inevitable for them to traverse numbers of these discretization samples to evaluate the time-variant energy gradient during each iteration. This could lead to low performance for complex models. Because they use the linear interpolation bases inherently, which may cause artifacts like shearing and anisotropic scaling, they cannot preserve the local details well. Some comparisons between Green Coordinates and linear subspace could be seen in Figs. 2 and 3. Our method avoids these problems by introducing the non-linear bases which are subjected to the vertex positions of the control mesh and solve the deformation independently of the embedded objects during the runtime calculation. Choi et al. [6] proposed a real-time simulation technique which eliminates the linearization artifacts in large deformation while retaining the efficiency of the previous modal analysis methods [9,18]. They also demonstrated that this technique can be used for the constraint-based motion retargeting, a similar application with deformation transfer. The deformation space of their method is spanned by the modal displacements calculated from the eigenvalue-decomposition. Different from this, our method defines the deformation space with a cage and it can be used for designing the deformation subspace intuitively. For example, in the articulated model deformation, the users can arrange the joint positions by designing the tesselation of the cage, which could be independent of the embedded model's shape.

1.3. Overview

Before introducing technical details, we briefly describe the procedure of our method.

- The input of our method includes a source deformation pose or sequence (Fig. 1(a)) and a target model with a cage (Fig. 1(b)). The cage is usually manually built by users and indicates the freedom of the deformation. Building such a cage is a common preliminary of cage-based deformation methods [12,11,10].
- The user selects corresponding points from the source and the target (as shown in Fig. 1(a, b), where the corresponding points are indicated by letters).
- The deformation gradients of the points on the source are extracted (shown in Fig. 1(a), where the material frames [2] on the selected points are extracted as the deformation gradients) and used as corresponding constraints to deform the target model by the subspace deformation method (shown in Fig. 1(c)).

In the next section we will introduce our cage-based subspace deformation method. Some technical details used in deformation transfer procedure are presented in Section 3. Implementation details are described in Section 4. Finally, we show our results in Section 5 and conclude this paper in Section 6.

2. Subspace deformation

The framework of our approach is similar to other deformation methods [22,5,10]. Our framework intends to solve the deformed models by minimizing the deformation energy. Usually, the total energy can be mainly divided into two parts: one is for satisfying the constraints specified by the users and the other part indicates an objective of the deformation, like detail preservation and

Please cite this article as: Chen L, et al. Cage-based deformation transfer. Computers and Graphics (2010), doi:10.1016/j.cag.2010.01.003

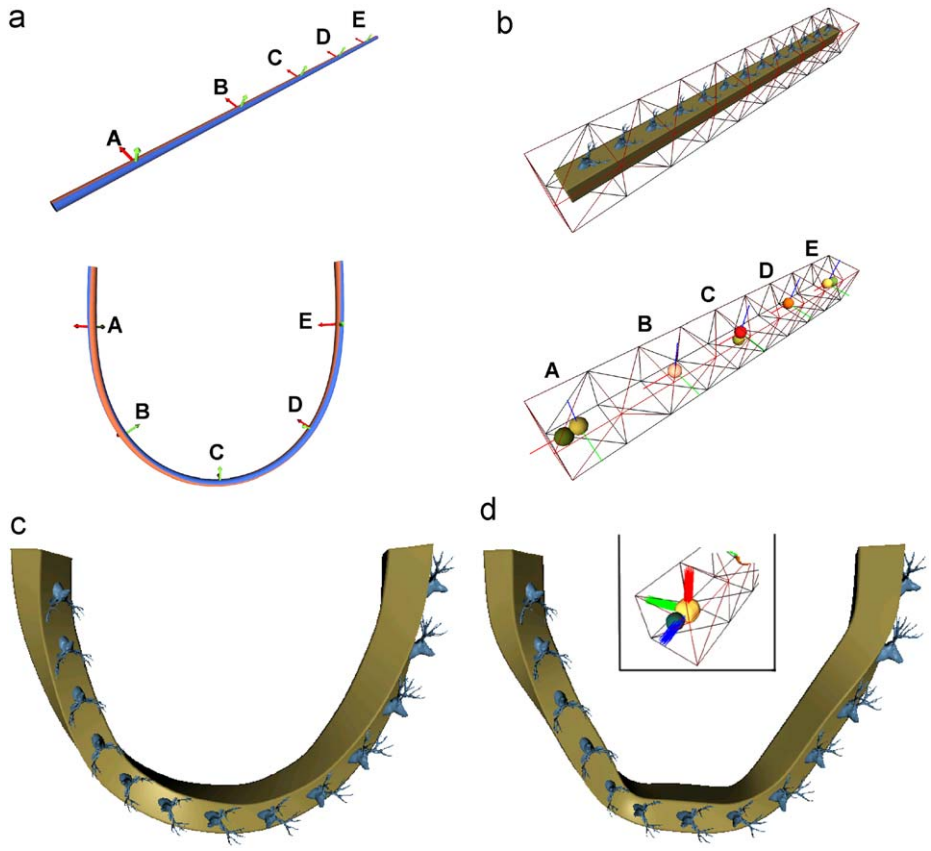

Fig. 1. (a) One deformed cable pose from a simulation program and five user-selected points with their material frames. (b) The up figure shows the static deer-beam model with the cage. The bottom one shows the gradient controls (the highlighted points with the local frame axes) and the position constraint controls (the other points). These constraints are all set by users. (c) The deformation transfer result for the beam model. We can get visually smooth deformation result under sparse constraints using our smoothness energy (shown in Section 2.1). (d) Without the smoothness energy, the deformation gradient changes unsmoothly in the cage. The small figure demonstrates the duplicated gradient constraints we used for preventing degeneration in this case.

smoothness. We can minimize the energy with the following formulation:

$$\min_{M'} E_{object}(M') + E_{constraint}(M'), \tag{1}$$

where M' is a vector of the vertices of the deformed model.

In our subspace method, the model is embedded in a cage and the target model can be expressed by the cage with interpolation function. Then, the energy minimization comes to

$$\min_{P'} E_{object}(f(P')) + E_{constraint}(f(P')), \tag{2}$$

where P' is the deformed cage mesh and $f(P') = M'$ is the interpolation function. In our method, the Green Coordinates interpolation which can preserve the details inherently was employed. So E_{object} does not need to include detail preservation energy explicitly for the subspace deformation. In the following parts, we will present our formulation of the energies for the subspace and the constraints.

2.1. Green Coordinates subspace

Instead of the previous linear interpolation methods, Green Coordinates interpolation, which ensures high quality quasi-conformal deformation, is employed in our method to encage the target model. Such a choice leads to a non-linear deformation energy only in terms of a few cage vertex positions. In this section, we first briefly review Green Coordinates representation and then introduce an energy term to enforce the smoothness of the deformation result.

We use the same notations as in [14]. Let the cage be an oriented simplicial surface that $P = (\mathbb{V}, \mathbb{T})$, where $\mathbb{V} = \{\boldsymbol{v}_i\}_{i \in I_{\mathbb{V}}} \subset \mathbb{R}^3$ are the vertices and $\mathbb{T} = \{t_j\}_{j \in I_{\mathbb{T}}}$ are the simplicial face elements. Here, $I_{\mathbb{V}}$ and $I_{\mathbb{T}}$ are the index sets of the vertices and faces, respectively. $\boldsymbol{n}(t_j)$ denotes the unit length outward normal to the oriented simplicial face $t_j (\|\boldsymbol{n}(t_j)\| = 1)$. The deformation function on each interior point $\boldsymbol{\eta} (\boldsymbol{\eta} \in \mathbb{R}^3)$ about the deformed cage (with the deformed cage vertices $\boldsymbol{v}'_i$ and faces t'_j) is described as following:

$$\boldsymbol{\eta}' = F(\boldsymbol{\eta}) = \sum_{i \in I_{\mathbb{V}}} \phi_i(\boldsymbol{\eta}) \boldsymbol{v}'_i + \sum_{j \in I_{\mathbb{T}}} \psi_j(\boldsymbol{\eta}) s_j \boldsymbol{n}(t'_j), \tag{3}$$

where F maps $\boldsymbol{\eta}$ to its new position $\boldsymbol{\eta}'$ after cage deformation. ϕ and ψ are the Green Coordinates used in [14] for the interpolation. s_j is the scale factor of the j th face:

$$s_j = \frac{\sqrt{\|\boldsymbol{a}\|^2 \|\boldsymbol{b}_0\|^2 - 2(\boldsymbol{a} \cdot \boldsymbol{b})(\boldsymbol{a}_0 \cdot \boldsymbol{b}_0) + \|\boldsymbol{b}\|^2 \|\boldsymbol{a}_0\|^2}}{\sqrt{8} area(t_j)},$$

where $\boldsymbol{a}_0$, $\boldsymbol{b}_0$, $\boldsymbol{a}$ and $\boldsymbol{b}$ are any two corresponding edge vectors of the original triangle t_j and deformed triangle t'_j.

We rearrange the summation into matrix form and pack variables as following:

$$F(\boldsymbol{\eta}) = \Phi(\boldsymbol{\eta}) V + \Psi(\boldsymbol{\eta}) N, \tag{4}$$

where V and N are the column vectors of packed vertices coordinates $\boldsymbol{v}'_i$ and scaled normal vectors $s_j \boldsymbol{n}(t'_j)$ of the deformed cage, respectively. Φ and Ψ are matrices in dimensions $3 \times 3|I_{\mathbb{V}}|$ and $3 \times 3|I_{\mathbb{T}}|$, respectively.

Please cite this article as: Chen L, et al. Cage-based deformation transfer. Computers and Graphics (2010), doi:10.1016/j.cag.2010.01.003

Because the subspace defined by Green Coordinates can preserve the details inherently, it seems that we could solve the following system directly for the shape preserving deformation:

$$\min_{P'} E_{constraint}(f(P')). \tag{5}$$

However, compared with the degrees of freedom (DOFs) of deformation, which is $3|I_{\mathbb{V}}|$ (the normal variable N lies on the vertices V during the deformation which fixes the triangles' topology and orientations), there might be much fewer DOFs associated with the constraints specified by users. It leads to a degenerated optimization problem for solving (5). More seriously, the deformation gradient in the quasi-conformal deformation field may not change smoothly (Fig. 1(d))[1] without any object energy E_{object} being used.

To overcome these problems, we penalize the variation of deformation gradient to make the deformation as smooth as possible. We measure the variation of deformation gradient at point $\boldsymbol{\eta}$ in the cage by the Hessian matrix of the deformation mapping function $\partial^2 F(\boldsymbol{\eta})/\partial\boldsymbol{\eta}\partial\boldsymbol{\eta}$ as follows:

$$\begin{aligned}\left\|\frac{\partial^2 F(\boldsymbol{\eta})}{\partial\boldsymbol{\eta}\partial\boldsymbol{\eta}}\right\|_F^2 &= \sum_{k\in\{x,y,z\}}\left\|\frac{\partial^2 F_k(\boldsymbol{\eta})}{\partial\boldsymbol{\eta}\partial\boldsymbol{\eta}}\right\|_F^2\\ &= \sum_{k\in\{x,y,z\}}\sum_{p,q\in\{x,y,z\}}\left(\frac{\partial^2 F_k(\boldsymbol{\eta})}{\partial p\partial q}\right)^2\\ &= \sum_{k,p,q\in\{x,y,z\}}\left(\frac{\partial^2 \Phi_k(\boldsymbol{\eta})}{\partial p\partial q}\cdot V+\frac{\partial^2 \Psi_k(\boldsymbol{\eta})}{\partial p\partial q}\cdot N\right)^2\\ &\triangleq \sum_{k,p,q\in\{x,y,z\}}(\hat{\Phi}_{kpq}(\boldsymbol{\eta})\cdot V+\hat{\Psi}_{kpq}(\boldsymbol{\eta})\cdot N)^2,\end{aligned} \tag{6}$$

where $\|\cdot\|_F$ denotes the Frobenius norm. $F_k(\boldsymbol{\eta}), k\in\{x,y,z\}$ denotes the three scalar components of the value of deformation function at point $\boldsymbol{\eta}$. And Φ_k, Ψ_k stand for the corresponding row vectors in the matrices Φ and Ψ. The symbol $\triangleq$ means "is equal to by definition" and here it defines that $\hat{\Phi}_{kpq}(\boldsymbol{\eta})=\partial^2\Phi_k(\boldsymbol{\eta})/\partial p\partial q$ and $\hat{\Psi}_{kpq}(\boldsymbol{\eta})=\partial^2\Psi_k(\boldsymbol{\eta})/\partial p\partial q$. Then, the energy for smoothness in the whole cage can be defined as the integral of the above term measured in the cage region $\mathbb{C}$:

$$E_{smooth}=\int\int\int_{\eta\in\mathbb{C}}\left\|\frac{\partial^2 F(\boldsymbol{\eta})}{\partial\boldsymbol{\eta}\partial\boldsymbol{\eta}}\right\|_F^2 d\omega. \tag{7}$$

Since we do not have a close-form expression of this integral, we convert it into a sum of finite samples. We subdivide the bounding box of the cage into a uniform lattice. Based on our experience, a regular lattice with 60 nodes along the longest edge of the bounding box can achieve visually satisfied result in usual cases. The second derivative terms $\hat{\Phi}_{kpq}(\boldsymbol{\eta}_i)$ and $\hat{\Psi}_{kpq}(\boldsymbol{\eta}_i)$ are calculated using finite difference method with the assistance of the lattice. Hence derivative of one node is computed from the function values of the neighboring nodes. Then, the smoothness energy can be approximated by the following summation:

$$E_{smooth}\approx\sum_{i=1}^{m_s}\sum_{k,p,q\in\{x,y,z\}}(\hat{\Phi}_{kpq}(\boldsymbol{\eta}_i)\cdot V+\hat{\Psi}_{kpq}(\boldsymbol{\eta}_i)\cdot N)^2=\|\hat{\boldsymbol{\Phi}}V+\hat{\boldsymbol{\Psi}}N\|^2, \tag{8}$$

where the m_s sample points are the nodes on the uniform lattice which can calculate the second derivatives. $\hat{\boldsymbol{\Phi}}$ and $\hat{\boldsymbol{\Psi}}$ are packed matrices for all the summarizing terms. Here, both the row dimension of $\hat{\boldsymbol{\Phi}}$ and $\hat{\boldsymbol{\Psi}}$ are $27m_s$, and 27 is the summarizing element number of the second derivative terms for one sample point.

[1] While generating this figure, we duplicate several gradient constraints around the original ones to prevent degeneration.

After using E_{smooth} as E_{object} in formula (2), undesired deformation variation can be efficiently eliminated from the conformal deformation space defined by the Green Coordinates, as shown in Fig. 1(c).

2.2. Constraints

In this section, we introduce the position, gradient and length constraints which are used in our deformation algorithm. All of them are formulated as penalty energies in the optimization framework.

2.2.1. Position constraint

The position constraint is an intuitive way to interact with the deformation procedure. Users can manipulate the vertices on the embedded object directly. Denoting the original positions of the k_p control points as $\boldsymbol{p}_i (i\in\{1,\ldots,k_p\})$ and the deformed positions as $P_i (i\in\{1,\ldots,k_p\})$, the energy term of the position constraints can be formulated as

$$\begin{aligned}E_{pos}&=\sum_{i=1}^{k_p}\|F(\boldsymbol{p}_i)-P_i\|^2\\ &=\sum_{i=1}^{k_p}\|\Phi(\boldsymbol{p}_i)V+\Psi(\boldsymbol{p}_i)N-P_i\|^2\\ &\triangleq\sum_{i=1}^{k_p}\|\Phi_i V+\Psi_i N-P_i\|^2.\end{aligned} \tag{9}$$

As shown in Figs. 2 and 3, the bend and twist effect can be achieved by minimizing the weighted sum of position constraints and the smoothness energy. The weights setting schema will be discussed in Section 4. Moreover, because the smoothness energy and the other two constraints cannot restrict the global translation of the target object, at least one position constraint must be given.

2.2.2. Gradient constraint

The gradient constraint is used for controlling the deformation gradients on the corresponding points in the cage space. As described in the Introduction section, it is the essential constraint used for the application of deformation transfer.

We denote the k_g gradient constraint points as $\boldsymbol{g}_i$ $(i\in\{1,\ldots,k_g\})$, and the corresponding deformation gradient on them as G_i $(i\in\{1,\ldots,k_g\})$. G_i is a 3×3 matrix, which may contain rotation, scale or shear transformation. We try to minimize the difference between the deformation gradient on the point $\boldsymbol{g}_i$ and the guidance G_i with the following energy:

$$\begin{aligned}E_{grad}&=\sum_{i=1}^{k_g}\|\nabla F(\boldsymbol{g}_i)-G_i\|_F^2=\sum_{i=1}^{k_g}\sum_{k\in\{x,y,z\}}\left\|\frac{\partial\Phi(\boldsymbol{g}_i)}{\partial k}V+\frac{\partial\Psi(\boldsymbol{g}_i)}{\partial k}N-G_{ik}\right\|^2\\ &=\sum_{i=1}^{k_g}\|\tilde{\Phi}(\boldsymbol{g}_i)V+\tilde{\Psi}(\boldsymbol{g}_i)N-\tilde{G}_i\|^2\triangleq\sum_{i=1}^{k_g}\|\tilde{\Phi}_i V+\tilde{\Psi}_i N-\tilde{G}_i\|^2,\end{aligned} \tag{10}$$

where G_{ik} is the column vector in the gradient matrix for the corresponding component and $\tilde{G}_i$ packs the columns of G_i into one column vector with dimension 9×1. $\tilde{\Phi}_i$ and $\tilde{\Psi}_i$ are the packed matrices for the summarizing terms of three components and terms $\partial\Phi(\boldsymbol{g}_i)/\partial k, \partial\Psi(\boldsymbol{g}_i)/\partial k$ are calculated by finite difference with step size which is consistent with the grid size used for calculating $\hat{\boldsymbol{\Phi}}$ and $\hat{\boldsymbol{\Psi}}$.

Gradients G_i can be set by users' interactions or transferred from other sources. In Fig. 1, deformation gradients are transferred from simulation data. In Fig. 4, gradient constraints are set by users for constraining the orientations of the head and feet.

Please cite this article as: Chen L, et al. Cage-based deformation transfer. Computers and Graphics (2010), doi:10.1016/j.cag.2010.01.003

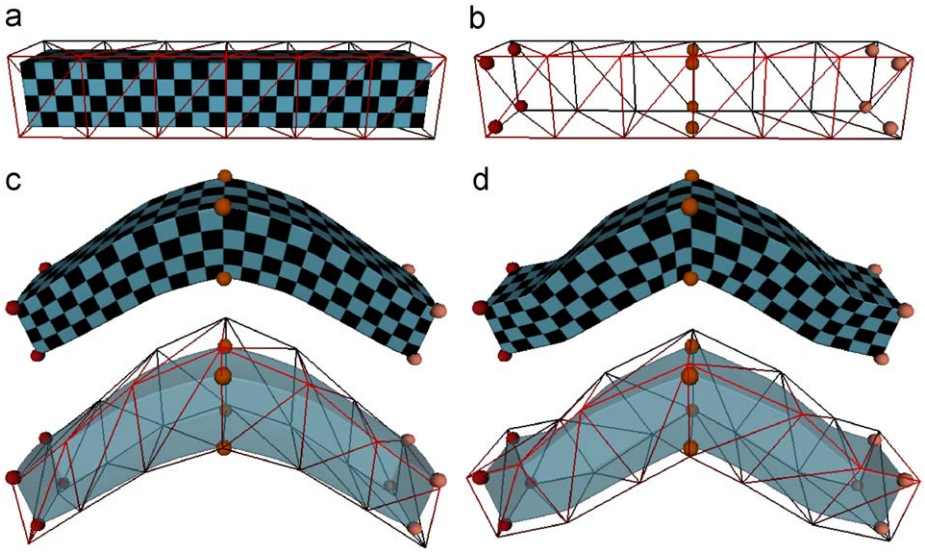

Fig. 2. The bend results using different subspace methods. (a) The rest shape with the cage. (b) The constraints used for the bend case. (c) The bend result of our method. (d) The bend result of mean value coordinates based subspace method described in [10].

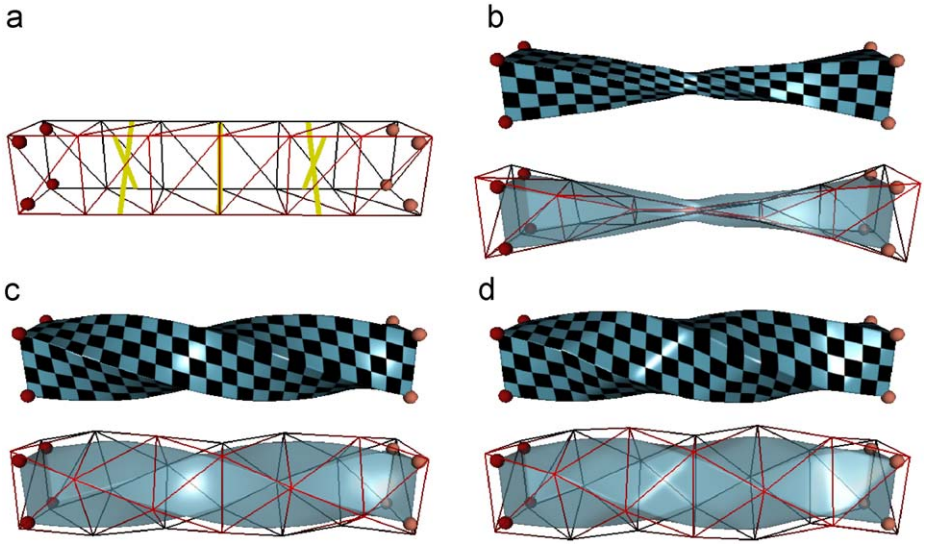

Fig. 3. The twist results using different subspace methods. (a) The constraints used for the twist case, where the yellow bars show the segments of the point pairs used as length constraints. These length constraints are only used in (c). (b) The twist result of our method with the smoothness energy and the position constraints only. (c) The twist result of our method with additional length constraints. (d) The twist result of method in [10]. This method does not involve length constraints. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

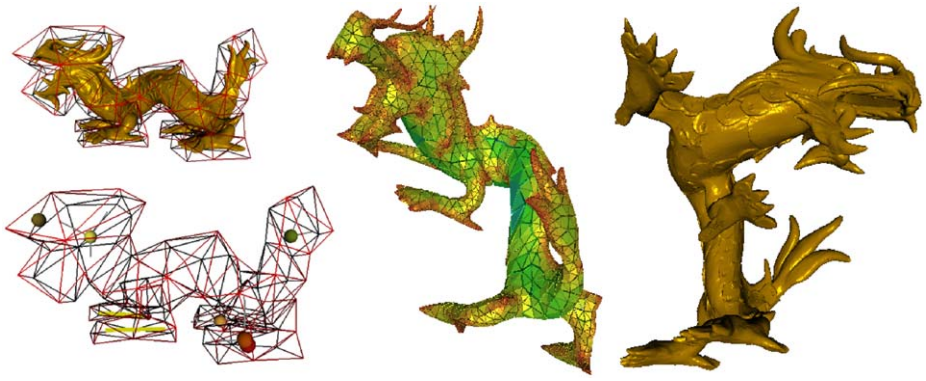

Fig. 4. The deformation result of the Asian dragon model. Both the surface representation and the tetrahedrons representation can be deformed with the same cage and constraints. The colors of the tetrahedrons do not relate to the deformation energy, but their volumes. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

Please cite this article as: Chen L, et al. Cage-based deformation transfer. Computers and Graphics (2010), doi:10.1016/j.cag.2010.01.003

2.2.3. Length constraint

Keeping some parts as-rigid-as-possible is a common requirement in many deformation cases. For example, in the character animation, the parts like head, legs and arms are assumed to be nearly rigid. Because the Green Coordinates restrict the deformation to be quasi-conformal, which leave almost only isotropic scale and rotation transformation in the deformation gradient, we can achieve quasi-rigid effects by restricting the isotropic scale in certain region. For this, we preserve the distance between some point pairs by the length constraints introduced below.

Denoting two user-selected points in the cage space as $\boldsymbol{p}^s$ and $\boldsymbol{p}^e$, the length constraint intends to keep the distance of the two points during the deformation. Given a set of such point pairs as $\boldsymbol{p}_i^s - \boldsymbol{p}_i^e (i \in 1, \ldots, k_l)$, the energy term of the length constraints can be formulated as

$$\begin{aligned} E_{len} &= \sum_{i=1}^{k_l} (\|F(\boldsymbol{p}_i^s) - F(\boldsymbol{p}_i^e)\|^2 - l_i^2)^2 \\ &= \sum_{i=1}^{k_l} (\|(\Phi(\boldsymbol{p}_i^s) - \Phi(\boldsymbol{p}_i^e))V \\ &\quad + (\Psi(\boldsymbol{p}_i^s) - \Psi(\boldsymbol{p}_i^e))N\|^2 - l_i^2)^2 \\ &\triangleq \sum_{i=1}^{k_l} (\|\overline{\Phi}_i V + \overline{\Psi}_i N\|^2 - l_i^2)^2, \end{aligned} \tag{11}$$

where l_i is the initial distance of the two points.

As the comparison shown in Fig. 3(b) and (c), our length constraints (shown in Fig. 3(a)) prevent shrinkage at corresponding parts, and the result achieves visually rigid effect.

2.3. Interactive deformation

Putting all the energies introduced above together, we get the following unconstrained optimization problem:

$$\min_V \omega_f E_{smooth} + \omega_g E_{grad} + \omega_p E_{pos} + \omega_l E_{len}, \tag{12}$$

where ω_f, ω_g, ω_p, ω_l are the weighting parameters of the energies. The weighting scheme would be discussed in Section 4.

By solving the non-linear optimization problem in terms of cage vertex positions (details can be found in Section 4), we can interactively deform the embedded object with details preserved.

In Fig. 5, we use only the position constraints to achieve large deformation effects with the details preserved well. In Fig. 4, gradient constraints are used for fixing the orientations of the dragon head and the feet. The length constraints prevent the unexpected scaling on the forefeet and head. Another dragon model composed of tetrahedrons can be used to perform the deformation procedure with the same constraints and cage, which shows that our method can handle volume-data geometric representations and does not need the information of topological connection.

3. Deformation transfer

We use deformation gradients to transfer deformation from source to target. The deformation gradients information can be extracted from various deformation sources. For a mesh deformation sequence, the deformation gradients are calculated on groups of selected vertices by shape matching technique [16], as shown in Fig. 6(a). For the motion capture data shown in Fig. 7, the deformation gradients can be the transformations of the selected bones. It is also possible to extract the deformation gradient from many other deformation representations. After the deformation gradient sequences were got from the source, users can apply them to drive the target model. Each deformation gradient sequence will be assigned to one gradient constraint point in the cage. Then the cage is deformed with the guidance of these deformation gradient constraints.

Taking Fig. 6 for example, we transfer the horse running gaits from the given mesh sequence to another horse model. Eight vertex groups on the legs of the source model are selected to extract the deformation gradients. Then we put the corresponding gradient controls in the leg parts of the cage, and this step could be considered as setting up correspondence.

Actually, users can choose deformation from different deformation sources and combine them into one target model. Fig. 7 shows such an example that we extract the deformations from two different motion capture sources and transfer to one target model, thus a new animation is generated.

3.1. Adapt gradient sequence to the target model

Sometimes, because the local frames of the source and target model are not consistent, the deformation gradients extracted from the source model cannot be used directly and we may need to adjust them by some extra transformations when applying them on the gradient control. For this, we introduce a pre-procedure operation and a post-procedure operation which apply the gradient data on the corresponding controls, respectively.

Without loss of generality, we assume that all the input deformation gradient sequences contain the data with the same frame number. Denote the input deformation gradient sequences as $S_i^j (i \in 1, \ldots, k_g, j \in 1, \ldots, k_f)$ (k_g is the number of gradient control and k_f is the frame number), where S_i^j is a 3×3 matrix. As shown in Fig. 8(a), the target right arm's orientation is not consistent with the static pose of the motion capture. In our method, we complement this orientation difference by a deformation procedure. The users can deform the model's right arm by rotating the gradient controls on the arm interactively (Fig. 8(b)). When the deformed pose is

Fig. 5. The deformation result of the armadillo model.

Please cite this article as: Chen L, et al. Cage-based deformation transfer. Computers and Graphics (2010), doi:10.1016/j.cag.2010.01.003

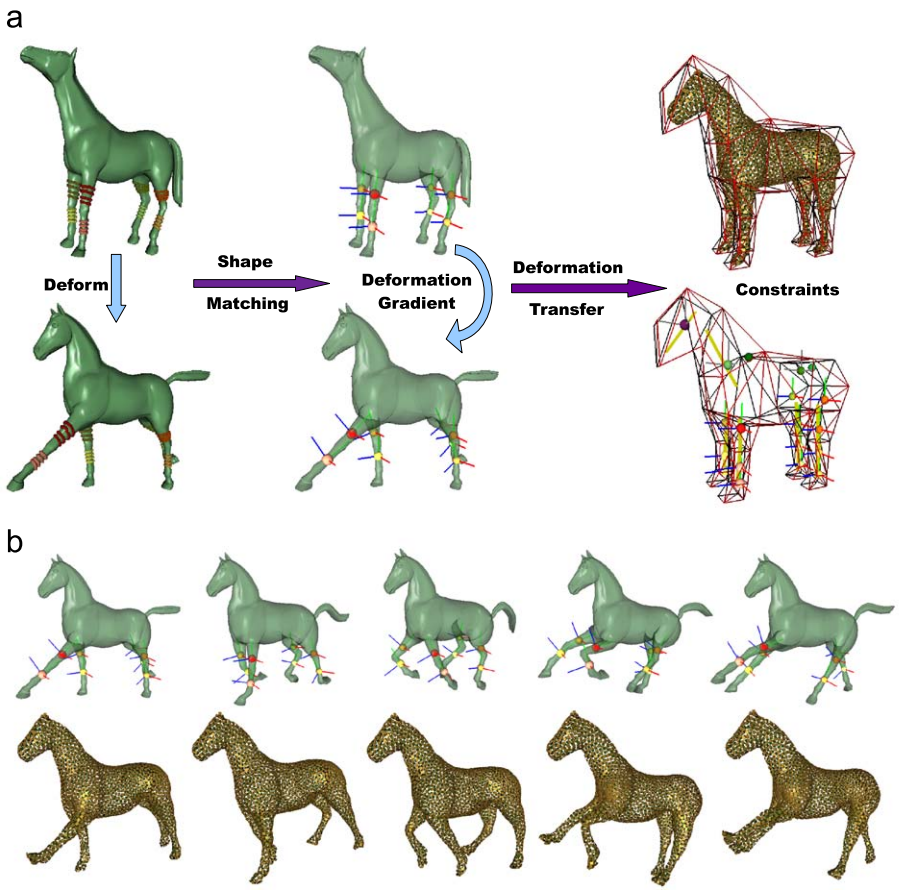

Fig. 6. (a) The left four figures show the procedure of shape matching [16]. The right two figures show the target model with the cage and the constraint controls we used for this case. The highlighted gradient controls are used for the gradient transfer and the color indicates the corresponding mapping. (b) The top row is the original mesh sequence with extracted gradients. The bottom row is the target model poses after gradient transfer. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

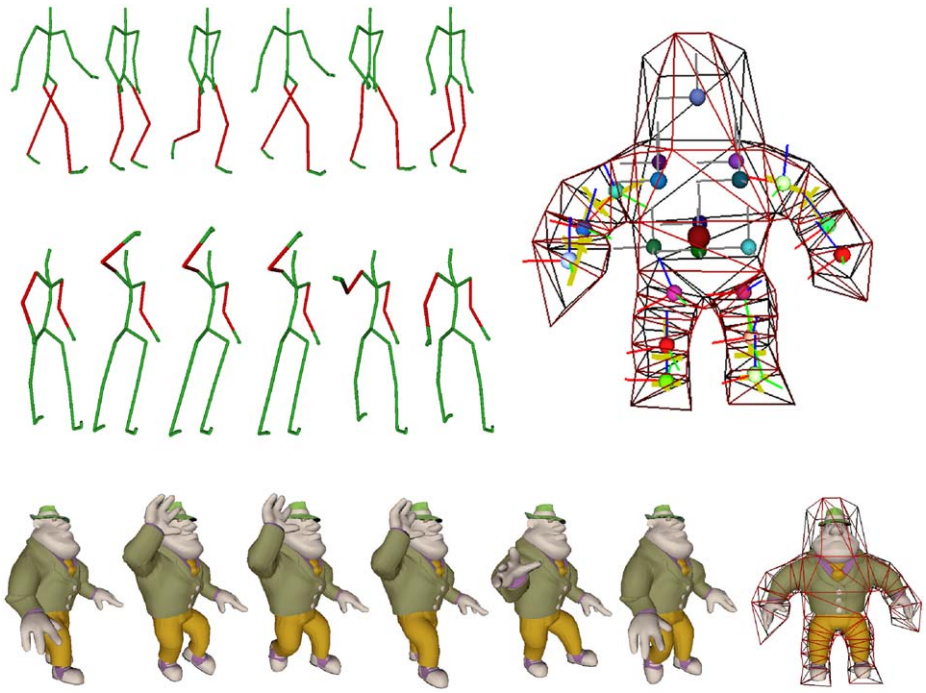

Fig. 7. The walking bubble man with waving hand. The two rows on the top left show the source motion capture data and the red bones are selected. The figure on the top right corner shows the cage and constraint controls (gradient and length controls) used for this demo. The figure on the bottom right corner shows the rest target model in the cage. The other six figures in the bottom row show the transfer result. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

Please cite this article as: Chen L, et al. Cage-based deformation transfer. Computers and Graphics (2010), doi:10.1016/j.cag.2010.01.003

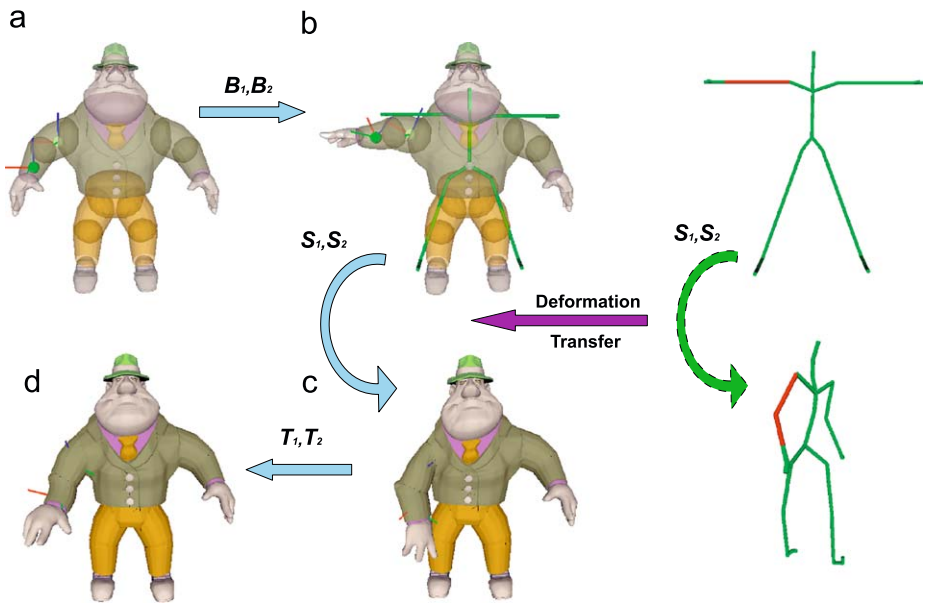

Fig. 8. The process of user interactions for adapting deformation gradient to the target model (the right arm only). The right two figures show the static and deformed pose in motion capture data and two deformation gradients of bones on the arm are extracted. (a) Rest target model with the gradient controls used for deformation transfer. To make more concisely, we hide the other constraints which are set for the deformation. (b) The pre-procedure operation: deforming the rest shape to match the reference skeleton. (c) Apply the deformation gradients. (d) The post-procedure operation: adjusting the orientation of the arm with transferred data.

consistent with the source model, the adjusted transformation B_i is recorded on the corresponding gradient control, where B_i is a 3×3 rotate matrix. Now the target model is actually deformed according to the gradient constraints $G_i = B_i$. After this pre-procedure operation, the deformation gradient S_i^j is applied on each gradient control for transferring the deformation (Fig. 8(c)) and the gradient constraints are $G_i = S_i^j B_i$. However, the transferred gradients may still not be suitable to the model. As shown in Fig. 8(c), because the body of motion capture source is toward a different direction with the target model, the transferred arm's orientation does not fit the target body's, thus causes intersections. This orientation difference can also be complemented by a deformation procedure similar to the previous one. The user rotates the gradient controls to deform the model into a proper pose (Fig. 8(d)) and then the adjusted transformation T_i, which is also a 3×3 rotation matrix, is recorded on each gradient control. Finally, the target model is deformed according to the gradient constraints $G_i = T_i S_i^j B_i$.

During this procedure, the users only need to set one B_i and one T_i for each gradient control if necessary. It is worth to emphasize that because the orientation differences B_i and T_i of one gradient control are consistent for all the frames, only one B_i and one T_i need to be adjusted for the whole sequence on each gradient control for the deformation transfer. Besides, our method is suitable to the case that the source data are given or difficult to be modified. If there are any more convenient methods to modify the source data (e.g. if it is easier to deform the static skeleton or source mesh into the static pose of the target model, which is an equivalent procedure to ours), the corresponding steps in our procedure could be omitted.

4. Implementation details

We will introduce details about the weighting schema, linearization techniques, pre-computation and convergence of our method in this section.

Weighting schema: The weighting parameters of the energies ω_f, ω_g, ω_p, ω_l in expression (12) can be used for adjusting the balance between emphasizing of different constraints. According to our experience, we found that a large range of values work well and a minimal amount of example-specific tuning is required. In this paper, we use the setting of $\omega_f = 1.0$, $\omega_g = 5.0$, $\omega_p = 0.5$, $\omega_l = 0.5$ for most of the examples and it works well for our deformation transfer application. One exception is that we use $\omega_l = 5$ in Fig. 3(c) to emphasize the effect of length constraints. Note that the Hessian matrices of the energies may have quite different Frobenius norms and we re-scaled them[2] before applying the above weighting.

Linearization: We use an iterative method to solve the non-linear least square problem. At each iteration k, the algorithm solves a subproblem of linearized least square for the new deformed cage vertices vector V_k, while the scaled normal N_k is treated as a function of V_k and updated from V_k after each iteration. The linearization is done by the Taylor expansion at V_{k-1}:

$$V_k = \operatorname*{argmin}_{V_k} \omega_f E'_{smooth} + \omega_g E'_{grad} + \omega_p E'_{pos} + \omega_l E'_{len},$$

$$E'_{smooth} = \|(\hat{\mathbf{\Phi}} V_{k-1} + \hat{\mathbf{\Psi}} N_{k-1}) + (\hat{\mathbf{\Phi}} + \hat{\mathbf{\Psi}} J_{k-1})(V_k - V_{k-1})\|^2,$$

$$E'_{grad} = \sum_i^{k_g} \|(\tilde{\Phi}_i V_{k-1} + \tilde{\Psi}_i N_{k-1} - \tilde{G}_i) + (\tilde{\Phi}_i + \tilde{\Psi}_i J_{k-1})(V_k - V_{k-1})\|^2,$$

$$E'_{pos} = \sum_i^{k_p} \|(\Phi_i V_{k-1} + \Psi_i N_{k-1} - P_i) + (\Phi_i + \Psi_i J_{k-1})(V_k - V_{k-1})\|^2,$$

$$E'_{len} = \sum_i^{k_l} ((\|\overline{\Phi}_i V_{k-1} + \overline{\Psi}_i N_{k-1}\|^2 - l_i^2) + 2(\overline{\Phi}_i V_{k-1} + \overline{\Psi}_i N_{k-1})^T (\overline{\Phi}_i + \overline{\Psi}_i J_{k-1})(V_k - V_{k-1}))^2, \quad (13)$$

where $J = \partial N / \partial V$.

[2] During the deformation, the Frobenius norms of the Hessian matrices may change because the Jacobian matrix of the scaled normal updates and here we always re-scale them according to the norms which are corresponding to the Jacobian matrix of static pose.

Please cite this article as: Chen L, et al. Cage-based deformation transfer. Computers and Graphics (2010), doi:10.1016/j.cag.2010.01.003

In addition, we find that based on the definition of N in the Green Coordinates, here is the relation $N=JV$ (detailed proof is shown in Appendix A). Therefore, we can further simplify the linearization expression by substituting equation $N_{k-1}=J_{k-1}V_{k-1}$:

$$E'_{smooth} = \|\hat{\mathbf{\Phi}}V_k + \hat{\mathbf{\Psi}}J_{k-1}V_k\|^2,$$

$$E'_{grad} = \sum_i^{k_g} \|\tilde{\Phi}_i V_k + \tilde{\Psi}_i J_{k-1}V_k - \tilde{G}_i\|^2,$$

$$E'_{pos} = \sum_i^{k_p} \|\Phi_i V_k + \Psi_i J_{k-1}V_k - P_i\|^2,$$

$$E'_{len} = \sum_i^{k_l} (2(\overline{\Phi}_i V_{k-1} + \overline{\Psi}_i N_{k-1})^T(\overline{\Phi}_i + \overline{\Psi}_i J_{k-1})V_k - (\|\overline{\Phi}_i V_{k-1} + \overline{\Psi}_i N_{k-1}\|^2 + l_i^2))^2. \quad (14)$$

Because we use the first-order approximations of the expressions in the least square, our solution is indeed equivalent to Gauss–Newton method, while only the second and higher order derivatives were ignored.

Pre-computing: As mentioned in Section 2.1, the matrices $\hat{\mathbf{\Phi}}$ and $\hat{\mathbf{\Psi}}$ are of high dimension $27m_s \times 3|I_{\mathbb{V}}|, 27m_s \times 3|I_{\mathbb{T}}|$. It is expensive to calculate and store these matrices. On the other hand, the gradient of the approximated smoothness energy is

$$\frac{\partial E'_{smooth}}{\partial V} = 2(\hat{\mathbf{\Phi}} + \hat{\mathbf{\Psi}}J_{k-1})^T(\hat{\mathbf{\Phi}} + \hat{\mathbf{\Psi}}J_{k-1})V_k = 2(\hat{\mathbf{\Phi}}^T\hat{\mathbf{\Phi}} + \hat{\mathbf{\Phi}}^T\hat{\mathbf{\Psi}}J_{k-1} + J_{k-1}^T\hat{\mathbf{\Psi}}^T\hat{\mathbf{\Phi}} + J_{k-1}^T\hat{\mathbf{\Psi}}^T\hat{\mathbf{\Psi}}J_{k-1})V_k, \quad (15)$$

where $\hat{\mathbf{\Phi}}^T\hat{\mathbf{\Phi}}$, $\hat{\mathbf{\Phi}}^T\hat{\mathbf{\Psi}}$, $\hat{\mathbf{\Psi}}^T\hat{\mathbf{\Phi}}$ and $\hat{\mathbf{\Psi}}^T\hat{\mathbf{\Psi}}$ are $3|I_{\mathbb{V}}| \times 3|I_{\mathbb{V}}|$, $3|I_{\mathbb{V}}| \times 3|I_{\mathbb{T}}|$, $3|I_{\mathbb{T}}| \times 3|I_{\mathbb{V}}|$ and $3|I_{\mathbb{T}}| \times 3|I_{\mathbb{T}}|$, respectively. So we pre-compute and reuse these matrix products instead of $\hat{\mathbf{\Phi}}, \hat{\mathbf{\Psi}}$. Other variables ($\tilde{\Phi}_i$, $\tilde{\Psi}_i$, Φ_i, Ψ_i, $\overline{\Phi}_i$ and $\overline{\Psi}_i$) in the constraint energies can be precomputed after the constraints are set. At each step of iteration, we only need to assemble them with the updated N_{k-1} and J_{k-1}, then solve a dense $3|I_{\mathbb{V}}|$ dimensional linear system to get V_k.

Convergence: To ensure the stability of the iteration procedure, we shorten the step size according to a coefficient $\alpha = 0.2$, i.e. the actual output of the k th iteration is $(1-\alpha)V_{k-1} + \alpha V_k$. As shown in Fig. 9, our solver converges fast and stably during the iteration procedure. In typical cases, the procedure converges into a visually stable state within about 20 iterations. Our Gauss–Newton solver has super-linear convergence rate, which is over the linear convergence rate of steepest descent method. Besides, the cost of each iteration of our method is cheaper than the steepest descent method, since we do not need to calculate the total energy several times to determine the step size.

5. Results

We perform our method for transferring the deformation from different sources to the geometric models in variant representations. As shown in Fig. 1, the deformation of the cable in the physical simulation is transferred to the deer-beam model.

Fig. 6 shows that we transfer the gaits of a running horse from a mesh sequence to another horse model composed of disconnected mosaic blocks. All the mosaic blocks are deformed by rigid transformations during the deformation, where the rotations of each block are extracted from the deformation gradients on the block centroids by polar decomposition and the translations are calculated by the Green Coordinates interpolation. The gradient controls on the feet use the same data with corresponding shanks. The actions of the stifles and the gaskins are also consistent by using the same gradients. The length constraints are applied on the legs, head and neck, which are denoted by the yellow bars in the figure.

Fig. 7 shows the case of transferring deformation from multiple motion capture data to the bubble man model. In this example, we transfer the motion of the legs in the upper motion capture data and the motion of the arms in the lower data to a single model simultaneously. The highlighted gradient controls are used for transferring the deformation gradients while the others are used for fixing the orientations of the chest and head. The gradient sequences for the gradient controls on the feet are the same as the ones for corresponding calves. The gradient controls on the hands are used in the same way. Length constraints are applied on the arms and legs to prevent undesirable shrinkage under large deformation.

Fig. 10 shows the case of transferring a jumping motion from motion capture data to a robot model. In this example, we select 12 bones for the gradient transfer. The corresponding gradient controls are highlighted in the figure. The two gradient controls on the chest and the stomach share the same data from one bone of the source. One position constraint is set on the left foot and used for setting the translation during the jumping action. The translation sequence is calculated from position sequence of the left toe in the motion capture data. Two length constraints are applied on the hands making these parts as-rigid-as-possible during the deformation.

Table 1 shows the statistics of the demo scenes in the paper, including the complexity of the scenes and the performance of the

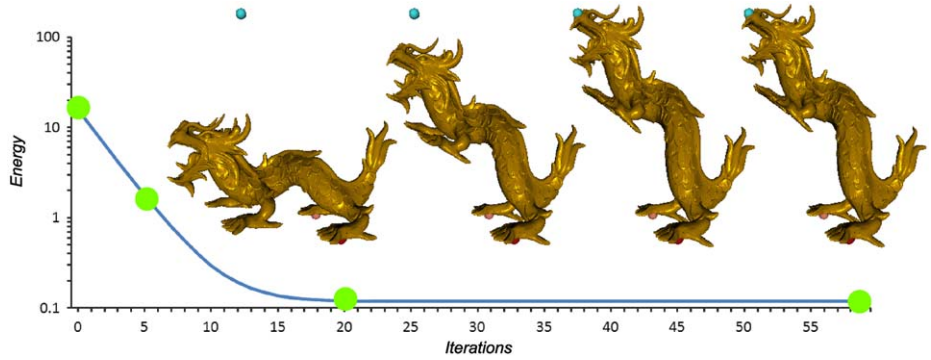

Fig. 9. The convergence curve of the solver. The asian dragon is deformed with the same constraints shown in Fig. 4. One of the position controls (the highlighted ball) is used to uplift the dragon. The curve shows the variety of the total energy (log-plot) of the system over the iterative solving procedure. The horizontal axis indicates the number of iteration, and the green points on the curve indicate the energies corresponding to the poses shown above. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

Please cite this article as: Chen L, et al. Cage-based deformation transfer. Computers and Graphics (2010), doi:10.1016/j.cag.2010.01.003

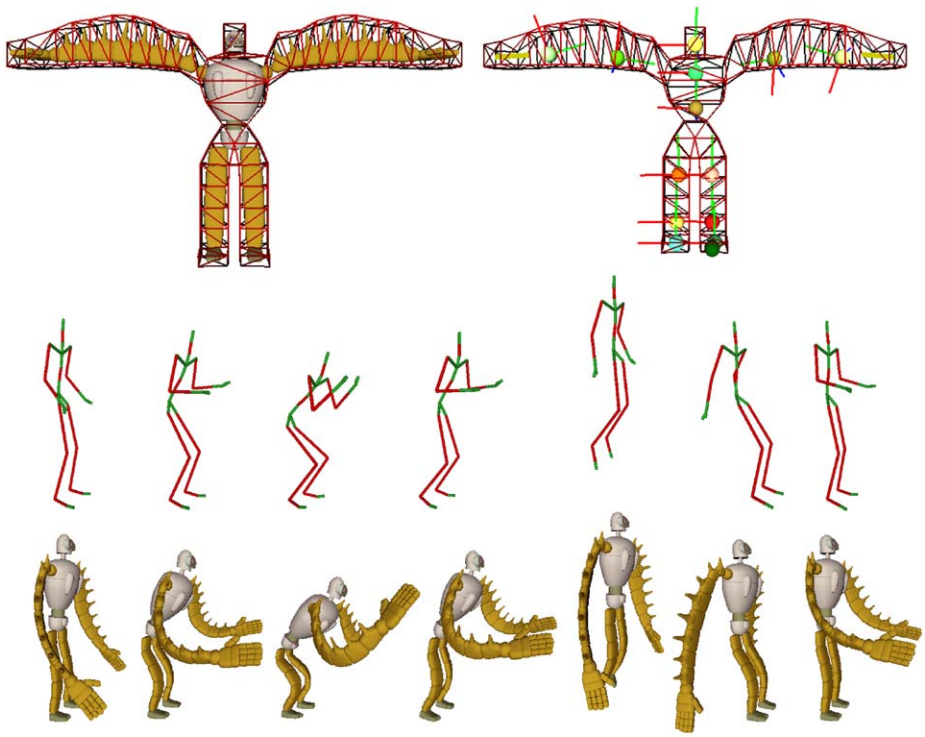

Fig. 10. The demo of the robot model. The top row shows the static model with the cage and the constraints we used for the demo. The middle row shows the original motion capture data. The red bones are the gradient sources. The bottom row shows the transferred animation. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

Table 1
The statistics of demos shown in the paper.

	Cage verts.	Embed verts.	Sample-pts.	Pre-compute time (s)		Run time (ms)	
				$\hat{\Phi}^T\hat{\Phi}$, etc.	Green coords.	Solver	Green interp.
Bubble man	106	52 736	24 319	57.5	90.3	102.25	14.35
Deer beam	40	72 116	2891	2.0	44.8	7.01	9.74
Armadillo	120	172 962	22 499	59.0	336.8	52.9	50.75
Asian dragon	126	249 934	9356	26.5	509.3	72.19	77.16
Tetra dragon	126	57 816	9356	26.5	118.2	72.19	20.00
Mosaic horse[a]	88	3076	19 760	37.8	29.7	49.13	83.01
Robot	162	20 556	6367	23.3	53.8	114.45	9.87

The second, third and fourth columns show the vertices number of the cages, the vertices number of the embedded models and the number of sample points we used for the numeric integration (Eq. (8)), respectively. The two columns of pre-compute time show the precomputed time of calculating the second derivative terms like $\hat{\Phi}^T\hat{\Phi}$ and the Green Coordinates. The two columns under run time show the run-time performance of our subspace solver and Green Coordinates interpolation, respectively.

[a] In the row of the horse demo, the embed vertices' number is the number of the mosaic blocks and the second column of run-time contains the time for driving the mosaics using the deformation gradients.

algorithm. The computation time, including both pre-computation and run-time, can be divided into two parts: the time of our subspace method spent on deforming the cage and Green Coordinates interpolation for driving the embedded object. The pre-compute time of our subspace method, mainly used on calculating the derivative terms $\hat{\Phi}^T\hat{\Phi}$, $\hat{\Phi}^T\hat{\Psi}$, $\hat{\Psi}^T\hat{\Phi}$ and $\hat{\Psi}^T\hat{\Psi}$, is related to the sample points' number we use for the numeric integration of Eq. (8). Running time column lists the time of one iteration of the solver and embedded mesh interpolation but does not include the normal calculation for rendering. The runtime cost of our subspace solver is related to the complexity of the cage and the constraints used for the deformation. All the timings in the table are measured on an Intel Pentium Dual 2.0 GHz, 3.0 GB RAM machine with a NVIDIA GForce 9800GT graphics card.

CUBLAS library was called by the Green Coordinate interpolation and the subspace solver in the runtime computing to accelerate matrix multiplication. Here, we convert the sparse Jacobi matrix J into a dense one for the multiplication calculation using CUBLAS on GPU, because it is faster than the CPU sparse matrix multiplication on our machine when the cage vertices's number is between 30 and 200, which covers most common cases in our application. When the vertices number is greater than 200, the computation complexity of the dense matrix multiplication is too high to be complemented by the parallel ability of the GPU and when the vertices number is smaller than 30, the GPU multiplication is slower due to the data transmission between memories.

6. Conclusion

We present a method for transferring the deformation from animation sequences to a target model. A new cage-based

Please cite this article as: Chen L, et al. Cage-based deformation transfer. Computers and Graphics (2010), doi:10.1016/j.cag.2010.01.003

subspace deformation method is brought forward to handle the target model in variant geometry representations. We employ deformation gradient as the transfer proxy, which is easy to extract from various animation sources. The subspace spanned by Green Coordinates preserves the shape inherently, thus we can achieve high quality deformation results with interactive frame rate by solving a non-linear optimization problem which is in terms of a few variables.

The major limitation of our method is that the deformation result depends on the shape and tessellation of the cage. Users should have some knowledge about building a proper cage for given animation sequences. It is not always an easy job for users to build a good cage, especially when the cage is extremely coarse. Detailed deformation in the animation sequence cannot be transferred to the target model because a coarse cage cannot provide enough DOFs, which is also a limitation of our algorithm. Both limitations are common problems of subspace methods. In the future, we would like to develop a post-processing method to complement the lost DOFs to overcome the above limitations.

Acknowledgements

This work is supported in part by National Natural Science Foundation of China (No. 60703039, 60933007), 973 program of China (No. 2009CB320801), Chinese Universities Scientific Fund (No. 2009QNA5023) and Hanqiu Sun is supported by RGC research grant (no. 416007), UGC direct grant for research (ref. 2050423). We also thank the anonymous reviewers for their helpful comments and suggestions.

Appendix A. Derivation of scaled normal equation

Here we will prove the equation $N=JV$, which is referred in Section 4. Without loss of generality, we consider the case of one single triangle. The V and N stand for the vertices and normal of one single triangle, respectively, in the following proof. Denote the vertices of the triangle as $V=[\boldsymbol{v}_1^T,\boldsymbol{v}_2^T,\boldsymbol{v}_3^T]^T$ (9×1 vector) and two edge vectors $\boldsymbol{a}=\boldsymbol{v}_2-\boldsymbol{v}_1, \boldsymbol{b}=\boldsymbol{v}_3-\boldsymbol{v}_2$. The scaled normal is $N=s\boldsymbol{n}=s\tilde{\boldsymbol{n}}/\|\tilde{\boldsymbol{n}}\|$, where $\tilde{\boldsymbol{n}}=\boldsymbol{a}\times\boldsymbol{b}$. Denote edge variable $D=[\boldsymbol{a}^T,\boldsymbol{b}^T]^T$ (6×1 vector) and because D is linear to the vertices V,

$$N=JV \;\Leftrightarrow\; N=\frac{\partial N}{\partial D}\frac{\partial D}{\partial V}V=\frac{\partial N}{\partial D}\begin{pmatrix}-I & I & 0\\ 0 & -I & I\end{pmatrix}V=\frac{\partial N}{\partial D}D. \tag{16}$$

Now, consider the right part of the last equation:

$$\frac{\partial N}{\partial D}D=\frac{\partial(s\boldsymbol{n})}{\partial D}D=\left(\frac{\partial s}{\partial D}\cdot D\right)\boldsymbol{n}+s\frac{\partial \boldsymbol{n}}{\partial D}D. \tag{17}$$

From definition of the scale

$$s=\frac{\sqrt{\|\boldsymbol{a}\|^2\|\boldsymbol{b}_0\|^2-2(\boldsymbol{a}\cdot\boldsymbol{b})(\boldsymbol{a}_0\cdot\boldsymbol{b}_0)+\|\boldsymbol{b}\|^2\|\boldsymbol{a}_0\|^2}}{\sqrt{8}area(t)}.$$

The partial derivative of the scale is

$$\begin{aligned}\frac{\partial s}{\partial D}\cdot D&=\begin{pmatrix}\frac{\partial s}{\partial \boldsymbol{a}}\\ \frac{\partial s}{\partial \boldsymbol{b}}\end{pmatrix}\cdot\begin{pmatrix}\boldsymbol{a}\\ \boldsymbol{b}\end{pmatrix}=\frac{\partial s}{\partial \boldsymbol{a}}\cdot\boldsymbol{a}+\frac{\partial s}{\partial \boldsymbol{b}}\cdot\boldsymbol{b}\\ &=\frac{\|\boldsymbol{b}_0\|^2\boldsymbol{a}^T-(\boldsymbol{a}_0\cdot\boldsymbol{b}_0)\boldsymbol{b}^T}{\sqrt{8}area(t)\sqrt{\|\boldsymbol{a}\|^2\|\boldsymbol{b}_0\|^2-2(\boldsymbol{a}\cdot\boldsymbol{b})(\boldsymbol{a}_0\cdot\boldsymbol{b}_0)+\|\boldsymbol{b}\|^2\|\boldsymbol{a}_0\|^2}}\cdot\boldsymbol{a}\\ &+\frac{-(\boldsymbol{a}_0\cdot\boldsymbol{b}_0)\boldsymbol{a}^T+\|\boldsymbol{a}_0\|^2\boldsymbol{b}^T}{\sqrt{8}area(t)\sqrt{\|\boldsymbol{a}\|^2\|\boldsymbol{b}_0\|^2-2(\boldsymbol{a}\cdot\boldsymbol{b})(\boldsymbol{a}_0\cdot\boldsymbol{b}_0)+\|\boldsymbol{b}\|^2\|\boldsymbol{a}_0\|^2}}\cdot\boldsymbol{b}\\ &=\frac{\|\boldsymbol{b}_0\|^2\boldsymbol{a}^T\boldsymbol{a}-2(\boldsymbol{a}_0\cdot\boldsymbol{b}_0)\boldsymbol{b}^T\boldsymbol{a}+\|\boldsymbol{a}_0\|^2\boldsymbol{b}^T\boldsymbol{b}}{\sqrt{8}area(t)\sqrt{\|\boldsymbol{a}\|^2\|\boldsymbol{b}_0\|^2-2(\boldsymbol{a}\cdot\boldsymbol{b})(\boldsymbol{a}_0\cdot\boldsymbol{b}_0)+\|\boldsymbol{b}\|^2\|\boldsymbol{a}_0\|^2}}=s.\end{aligned} \tag{18}$$

The partial derivation of the normal is

$$\begin{aligned}\frac{\partial \boldsymbol{n}}{\partial D}D&=\frac{\partial \boldsymbol{n}}{\partial \tilde{\boldsymbol{n}}}\frac{\partial \tilde{\boldsymbol{n}}}{\partial D}D=\frac{I-\boldsymbol{n}\boldsymbol{n}^T}{\|\tilde{\boldsymbol{n}}\|}(-[\boldsymbol{b}]_\times \quad [\boldsymbol{a}]_\times)\begin{pmatrix}\boldsymbol{a}\\ \boldsymbol{b}\end{pmatrix}\\ &=2\frac{I-\boldsymbol{n}\boldsymbol{n}^T}{\|\tilde{\boldsymbol{n}}\|}(\boldsymbol{a}\times\boldsymbol{b})=2(I-\boldsymbol{n}\boldsymbol{n}^T)\boldsymbol{n}=2(\boldsymbol{n}-\boldsymbol{n}(\boldsymbol{n}^T\boldsymbol{n}))=0.\end{aligned} \tag{19}$$

Substitute Eqs. (18) and (19) into Eq. (17),

$$\frac{\partial N}{\partial D}D=s\boldsymbol{n}+s\cdot 0=s\boldsymbol{n}=N. \tag{20}$$

Thus, $N=JV$.

Appendix B. Supplementary data

Supplementary data associated with this article can be found in the online version at doi:10.1016/j.cag.2010.01.003.

References

[1] Au OK-C, Fu H, Tai C-L, Cohen-Or D. Handle-aware isolines for scalable shape editing. ACM Trans Graph 2007;26(3):83.
[2] Bergou M, Wardetzky M, Robinson S, Audoly B, Grinspun E. Discrete elastic rods. In: SIGGRAPH '08: ACM SIGGRAPH 2008 papers. New York, NY, USA: ACM; 2008. p. 1–12.
[3] Botsch M, Kobbelt L. Real-time shape editing using radial basis functions. In: Computer Graphics Forum, 2005. p. 611–21.
[4] Botsch M, Pauly M, Gross M, Kobbelt L. Primo: coupled prisms for intuitive surface modeling. In: SGP '06: proceedings of the fourth Eurographics symposium on geometry processing. Eurographics Association, Aire-la-Ville, Switzerland, 2006. p. 11–20.
[5] Botsch M, Pauly M, Wicke M, Gross M. Adaptive space deformations based on rigid cells. Comput Graph Forum 2007;26(3):339–47.
[6] Choi MG, Ko H-S. Modal warping: real-time simulation of large rotational deformation and manipulation. IEEE Trans Visualization Comput Graph 2005;11(1):91–101.
[7] Der KG, Sumner RW, Popović J. Inverse kinematics for reduced deformable models. ACM Trans Graph 2006;25(3):1174–9.
[8] Guskov I, Sweldens W, Schröder P. Multiresolution signal processing for meshes. In: SIGGRAPH '99: proceedings of the 26th annual conference on computer graphics and interactive techniques. New York, NY, USA: ACM Press, Addison-Wesley Publishing Co.; 1999. p. 325–34.
[9] Hauser KK, Shen C, O'Brien JF. Interactive deformation using modal analysis with constraints. In: Graphics Interface. Halifax, Nova Scotia: CIPS, Canadian Human Computer Commnication Society, 2003. p. 247–56.
[10] Huang J, Shi X, Liu X, Zhou K, Wei L-Y, Teng S-H, et al. Subspace gradient domain mesh deformation. ACM Trans Graph 2006;25(3):1126–34.
[11] Joshi P, Meyer M, DeRose T, Green B, Sanocki T. Harmonic coordinates for character articulation. In: SIGGRAPH '07: ACM SIGGRAPH 2007 papers. New York, NY, USA: ACM; 2007. p. 71.
[12] Ju T, Schaefer S, Warren J. Mean value coordinates for closed triangular meshes. In: SIGGRAPH '05: ACM SIGGRAPH 2005 papers. New York, NY, USA: ACM; 2005. p. 561–6.
[13] Kobbelt L, Campagna S, Vorsatz J, Seidel H-P. Interactive multi-resolution modeling on arbitrary meshes. In: SIGGRAPH '98: proceedings of the 25th annual conference on computer graphics and interactive techniques. New York, NY, USA: ACM; 1998. p. 105–14.
[14] Lipman Y, Levin D, Cohen-Or D. Green coordinates. In: SIGGRAPH '08: ACM SIGGRAPH 2008 papers. New York, NY, USA: ACM; 2008. p. 1–10.
[15] Lipman Y, Sorkine O, Cohen-Or D, Levin D, Rössl C, Seidel H-P. Differential coordinates for interactive mesh editing. In: Proceedings of shape modeling international. IEEE Computer Society Press; 2004. p. 181–90.
[16] Müller M, Heidelberger B, Teschner M, Gross M. Meshless deformations based on shape matching. ACM Trans Graph 2005;24(3):471–8.
[17] Nealen A, Mueller M, Keiser R, Boxerman E, Carlson M. Physically based deformable models in computer graphics. Comput Graph Forum 2006;25(4):809–36.
[18] Pentland A, Williams J. Good vibrations: modal dynamics for graphics and animation. SIGGRAPH Comput Graph 1989;23(3):207–14.
[19] Sederberg TW, Parry SR. Free-form deformation of solid geometric models. In: SIGGRAPH '86: proceedings of the 13th annual conference on computer graphics and interactive techniques. New York, NY, USA: ACM; 1986. p. 151–60.

[20] Shi L, Yu Y, Bell N, Feng W-W. A fast multigrid algorithm for mesh deformation. ACM Trans Graph 2006;25(3):1108–17.
[21] Sumner RW, Popović J. Deformation transfer for triangle meshes. In: SIGGRAPH '04: ACM SIGGRAPH 2004 papers. New York, NY, USA: ACM; 2004. p. 399–405.
[22] Sumner RW, Schmid J, Pauly M. Embedded deformation for shape manipulation. In: SIGGRAPH '07: ACM SIGGRAPH 2007 papers. New York, NY, USA: ACM; 2007. p. 80.
[23] Yu Y, Zhou K, Xu D, Shi X, Bao H, Guo B, et al. Mesh editing with Poisson-based gradient field manipulation. In: SIGGRAPH '04: ACM SIGGRAPH 2004 papers. New York, NY, USA: ACM; 2004. p. 644–51.
[24] Zhou K, Huang J, Snyder J, Liu X, Bao H, Guo B, et al. Large mesh deformation using the volumetric graph Laplacian. In: SIGGRAPH '05: ACM SIGGRAPH 2005 papers. New York, NY, USA: ACM; 2005. p. 496–503.
[25] Zhou K, Huang X, Xu W, Guo B, Shum H-Y. Direct manipulation of subdivision surfaces on gpus. ACM Trans Graph 2007;26(3):91.

Please cite this article as: Chen L, et al. Cage-based deformation transfer. Computers and Graphics (2010), doi:10.1016/j.cag.2010.01.003