Inverse Kinematics for Reduced Deformable Models

Kevin G. Der Robert W. Sumner[†] Jovan Popović

Computer Science and Artificial Intelligence Laboratory
Massachusetts Institute of Technology

[†]ETH Zürich

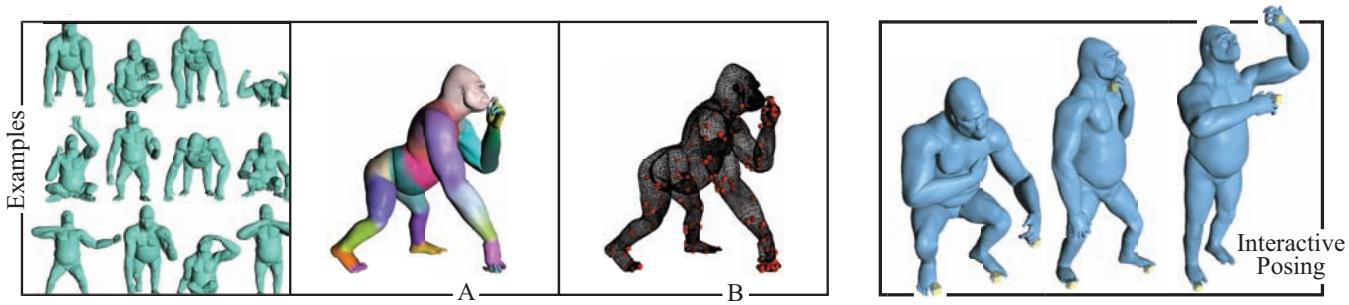

Figure 1: Our method uses example shapes to build a reduced deformable model, visualized in (A) by coloring portions of the mesh that move in a coordinated fashion. (B) A small number of proxy vertices are found that summarize the movement of the example meshes independent of their geometric complexity, providing a resolution-independent metric for mesh posing. The user can pose even highly detailed meshes interactively with just a few vertex constraints.

Abstract

Articulated shapes are aptly described by reduced deformable models that express required shape deformations using a compact set of control parameters. Although sufficient to describe most shape deformations, these control parameters can be ill-suited for animation tasks, particularly when reduced deformable models are inferred automatically from example shapes. Our algorithm provides intuitive and direct control of reduced deformable models similar to a conventional inverse-kinematics algorithm for jointed rigid structures. We present a fully automated pipeline that transforms a set of unarticulated example shapes into a controllable, articulated model. With only a few manipulations, an animator can automatically and interactively pose detailed shapes at rates independent of their geometric complexity.

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

Keywords: Animation with Constraints, Deformations

Contact: {kevinder|jovan}@csail.mit.edu
[†]sumnerb@inf.ethz.ch

1 Introduction

Efficient and intuitive manipulation of detailed triangle meshes is challenging because they have thousands of degrees of freedom. Modeling algorithms must cope with this geometric complexity to provide effective tools for sculpting broad changes as well as fine-scale details. However, in animation, the complexity of a character's *movement* is far less than its geometric complexity since vertices move in a coordinated fashion. An articulated character bends its limbs at the joints, so most limb vertices move together rigidly. Even non-articulated deformations such as those of a slithering snake, facial expressions, or skin deformations are highly correlated because an individual vertex of a detailed mesh never moves independently with respect to its neighbors.

Animators often build a reduced deformable model that represents meaningful deformations by instrumenting a static mesh with control parameters that modify posture, bulge muscles, change facial expressions, and generate other necessary deformations. These controls provide a compact representation of the mesh deformation and allow the animator to generate movement efficiently. However, many animation tasks are more easily accomplished through direct manipulation. In particular, reaching for or interacting with surrounding objects is most effectively expressed through direct control of contact vertices.

Reduced deformable models can also be inferred automatically from a set of example deformations. Although this approach eases the laborious task of designing controls by hand, applications are limited because the inferred control parameters are often ill-suited for animation tasks. Our work hides these unintuitive control parameters with a procedure for direct manipulation of reduced deformable models, allowing the animator to generate meaningful mesh deformations without learning the mapping between the controls and their effects.

Our method identifies control parameters of a reduced deformable model with a set of transformation matrices that control shape deformations. The animator can then select and move any subset of mesh vertices to pose the entire shape (Figure 1). In general, direct manipulation is an ill-posed problem whether using skeletons or inferred models, since many pose configurations can satisfy a given set of user constraints. We use a nonlinear optimization to find the pose whose transformations best meet the animator's constraints with a resolution-independent objective function, while favoring poses close to the space of examples. Lastly, a linear reconstruction computes the new deformed vertex positions. This final step is linear in the number of vertices but is computationally negligible when implemented efficiently in hardware. In total, the result is an inverse-kinematics method that permits interactive animation of highly detailed shapes using a space learned from just a few scanned or hand-sculpted poses.

2 Related Work

Because of an ever increasing demand for realism in computer graphics and the wide availability of 3D scanners, meshes with extremely high geometric complexity are commonplace. Research in mesh editing focuses on tools to manage the complexity while performing sculpting operations. Detail preservation is a central goal: low-frequency changes to the mesh should preserve the high-frequency details. Multiresolution methods [Zorin et al. 1997; Lounsbery et al. 1997; Kobbelt et al. 1998] address mesh detail directly by generating a hierarchy of simplified meshes together with corresponding detail coefficients. Modeling tools based on differential representations encode mesh detail in terms of local shape properties such as curvature, scale, and orientation. Sculpting operations can be performed by solving a discretized version of the Laplace [Lipman et al. 2004; Sorkine 2005] or Poisson equation [Yu et al. 2004]. The most successful editing methods are linear in the mesh vertices and admit efficient solution even for detailed meshes through pre-factorization of the linear system. A recent technique [Lipman et al. 2005] achieves rotation invariance with two linear solves. While mesh editing tools excel at sculpting operations, they are not designed for animation since they cannot easily incorporate the user's semantic knowledge about the deformations needed for a particular task.

Mesh-based inverse kinematics learns a space of natural deformations from example meshes [Sumner et al. 2005]. All tasks, which are specified by direct manipulation of a few mesh vertices, are accomplished with deformations that mimic those of the examples. However, the drawback of this method is that the number of unknowns in the optimization is proportional to the *geometric* complexity of the mesh, rather than to the *deformation* complexity. Each triangle is assigned a transformation matrix that encodes an independent rotation and stretch. The correlated movement of vertices in the example meshes is ignored so that non-meaningful deformations incur the same computational cost as highly structured ones. A special-purpose solver can solve the associated nonlinear optimization problem quickly enough for interaction with moderately sized meshes of around 10,000 vertices. Conversely, our method takes advantage of the deformation complexity to achieve interactive rates for meshes with extremely detailed geometry.

Shape interpolation or manual rigging with skeletal structures is the traditional way to represent deformations compactly, particularly in film production where the quality of the final result often justifies the cost of manual labor [Lewis et al. 2000; Sloan et al. 2001]. In other applications, methods such as modal analysis [Pentland and Williams 1989; James and Pai 2002] compute a set of basis deformations to enable real-time simulations of deformable models, while other methods rely on similar principles to achieve compression or hardware-accelerated rendering [Alexa and Müller 2000; Karni and Gotsman 2004; James and Twigg 2005]. Existing techniques that solve for the parameters of a linear superposition of basis vectors using an inverse-kinematics interface [Zhang et al. 2004] perform well when exhibited deformations are small but do not extend to deformations of articulated figures [Sumner et al. 2005].

Our algorithm uses a reduced deformable model that generalizes the core technique for animation of skeletal deformation. As shown in previous work, this model can be inferred automatically by identifying clusters of triangles that exhibit near-rigid deformation in the example meshes [James and Twigg 2005]. Although the automatically generated control parameters do not immediately support intuitive mesh control, our algorithm shows that an animator can manipulate meshes directly without learning the relationship between control parameters and their effects. In doing so, we broaden the scope of the original model from compression and real-time playback, where control is not needed, to animation, where control is paramount.

3 Inverse Kinematics

Our inverse-kinematics algorithm provides intuitive and direct control of non-rigid shapes similar to a conventional inverse-kinematics algorithm for jointed rigid skeletons. It manages the geometric complexity of detailed meshes by using a reduced deformable model that parameterizes shape deformation with a compact set of control parameters. For example, our algorithm could be applied to manually skinned meshes whose deformations have been parameterized by the joint variables in the skeleton. More interestingly, reduced deformable models can be inferred automatically from a few example shapes to learn the demonstrated space of shape deformations. Further, we show how to select the optimal pose near the space of example shapes from the many valid configurations that meet the animator's manipulations.

3.1 Reduced Deformable Model

Reduced deformable models describe shape deformations with a compact set of control parameters. For example, a conventional skinning model may employ a skeleton to deform mesh vertices according to a weighted average of bone transformations. The animator can then change the shape by modifying individual joint variables explicitly or by using a traditional inverse-kinematics algorithm for jointed rigid bodies. However, the simple skinning model struggles to capture articulated shape deformation without significant visual artifacts [Mohr and Gleicher 2003].

A more powerful reduced deformable model can be obtained by generalizing the notion of a skeleton from a collection of hierarchically linked rigid bones to a set of abstract, non-hierarchical, non-rigid control transformations. With this generalization in place, a reduced deformable model can be built automatically by identifying portions of the mesh whose deformation in the examples is nearly rigid [James and Twigg 2005]. The estimated model consists of a set of affine control transformations $\{\mathbf{F}_i, \mathbf{d}_i\}$ and a set of skinning weights $\{\alpha_i(\mathbf{x})\}$ that depend on the coordinates of each vertex. Together these define each shape deformation as a weighted average of transformed vertex coordinates:

$$\mathbf{v} = \sum_i \alpha_i(\mathbf{x})(\mathbf{F}_i\mathbf{x} + \mathbf{d}_i), \tag{1}$$

where the coordinates $\mathbf{x}$ and $\mathbf{v}$ contain the position of the vertex before and after deformation, respectively. Though similar to the expression used in a more conventional skinning model, the abstract affine transformations enrich the space of representable shapes to allow for more complex deformations [James and Twigg 2005]. An animator could, in principle, deform the mesh by manually adjusting control transformations; however, the complex dependencies of the controls and the lack of a hierarchy make such indirect manipulation quite impractical. Instead, our inverse-kinematics algorithm offers direct control over mesh vertices.

3.2 Direct Manipulation

Direct manipulation eases shape deformation through intuitive control of individual mesh vertices. Instead of positioning a character's hands or feet by adjusting the joint angles of the elbow, shoulder, and so on, tasks such as lifting, reaching, and locomotion are described intuitively with direct control over hand or feet vertices. Direct manipulation is crucial for control of a reduced deformable model whose control parameters include numerous rotations, scales, and translations with no intuitive or semantic meaning. Not only is the total number of parameters overwhelming but also the changes in control parameters must be coordinated to induce meaningful deformations of the shape. Simply observing the

control transformations for each example is ineffective at coordinating their movement, because it does not capture the relationships among them.

This necessary coordination can be learned automatically through optimization. For example, a nonlinear blend of deformation gradients produces an effective objective function for direct manipulation of general shape deformations [Sumner et al. 2005]. However, direct application of deformation gradients is inefficient for geometrically detailed meshes because it requires examining every vertex of the model regardless of its motion complexity. We obtain a resolution-independent solution with approximations that replace groups of related mesh vertices with a single new estimate. This reduces the size of the numerical problem significantly while preserving the visual quality of the final result.

Deformation Gradients. Deformation gradients are convenient localized descriptions of arbitrary deformation [Barr 1984]. These simple 3×3 transformations can be interpolated and aggregated to reconstruct pleasing deformations from a few example shapes [Alexa et al. 2000; Sumner and Popović 2004]. We examine the deformation gradients of vertices, because the vertices affected by multiple control transformations establish additional relationships among the controls to ensure that they change in coordination.

The deformation gradient of a vertex in our reduced deformable model is computed by differentiating Eq. (1) with the product rule:

$$\mathrm{D}_{\mathbf{x}}\mathbf{v} = \sum_i \Big(\mathbf{F}_i \big(\mathbf{1}_{3\times 3}\alpha_i(\mathbf{x}) + \mathbf{x}\mathrm{D}_{\mathbf{x}}\alpha_i(\mathbf{x}) \big) + \mathbf{d}_i \mathrm{D}_{\mathbf{x}}\alpha_i(\mathbf{x}) \Big). \quad (2)$$

The derivatives of the skinning weights are non-zero because they depend on the position of each vertex. The non-negative skinning weights are computed in the construction of the reduced deformable model by minimizing the least-squares error between predicted vertex positions and their true location in the example shapes [James and Twigg 2005]. The derivative values can be precomputed using a finite difference approximation.

The deformation gradients of the vertices in each example shape allow us to define a nonlinear objective function that coordinates all control parameters to ensure meaningful deformation. For example, we can blend between the example shapes by solving for the control parameters that yield deformation gradients most similar to those of the blend:

$$\mathbf{t}^* = \arg\min_{\mathbf{t}} \|\mathbf{G}\mathbf{t} - \mathbf{m}(\beta)\|^2, \quad (3)$$

where the vector $\mathbf{t}$ stacks the unrolled control transformations and the matrix $\mathbf{G}$ is built from the coefficients in Eq. (2). The blending function $\mathbf{m}(\beta)$ combines the deformation gradients of the vertices of each example, where β consists of a blending weight for each example. Deformation gradients are factored into rotational and scale/shear terms prior to blending rotations with the nonlinear exponential map [Sumner et al. 2005].

Proxy Vertices. The objective function need not evaluate deformation gradients at every mesh vertex because reduced deformable models coordinate the motion of many vertices. This structure allows us to design an efficient algorithm whose complexity is independent of geometric detail. The procedure is most easily observed in vertices whose positions are influenced by only one control transformation and whose weight derivatives are zero. These vertices all share the identical deformation gradient. As a result, they contribute identical row entries to both the deformation-gradient matrix $\mathbf{G}$ and the nonlinear blend of feature vectors $\mathbf{m}(\beta)$. Eliminating this redundancy reduces the size of the problem and maintains the quality of the final result.

More generally, our formulation replaces each group of vertices affected by the same set of control parameters with a single proxy vertex, as shown in Figure 2. Intuitively, this process strives to map the objective function from the high-dimensional space of vertex positions to the low-dimensional space of control parameters. For example, replacing a group of vertices that are attached to one control transformation is analogous to evaluating the deformation gradient of that transformation, rather than of all the individual vertices. Other vertices that are affected by more than one control transformation establish additional relationships among controls to ensure that all control parameters change in coordination.

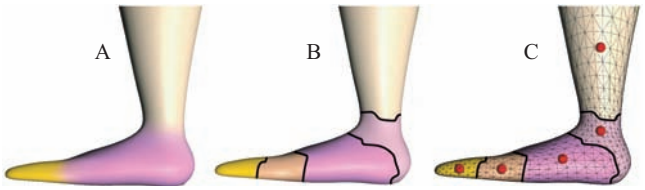

Figure 2: Our resolution-independent formulation evaluates deformation gradients at a few carefully chosen proxy vertices instead of the many mesh vertices. (A) A portion of a mesh is displayed, visualizing the influence of three control parameters and (B) five resulting vertex groups, each influenced by either one or two control parameters. (C) Each group is replaced by a proxy vertex, shown in red, located at the weighted centroid of its vertices.

Each vertex group affected by more than one control transformation is represented by the weighted centroid of its vertices. We prefer centroids that are uniformly affected by all control transformations for this group. In a skeleton, such a centroid would approximate the location of a joint because vertices close to the joint are usually uniformly affected by both adjoining bones. As a result, our centroid computation assigns more importance to vertices with uniform skinning weights (smaller maximum weight $\bar{\alpha}(\mathbf{x}_i)$) than to vertices with non-uniform weights (larger maximum weight). The position of a new proxy vertex $\mathbf{f}$ for group V is given by:

$$\mathbf{f} = \frac{\sum_{i\in V} \mathbf{x}_i (1 - \bar{\alpha}(\mathbf{x}_i))}{\sum_{i\in V} (1 - \bar{\alpha}(\mathbf{x}_i))}. \quad (4)$$

The deformation gradient for each new $\mathbf{f}$ is computed using Eq. (2) after its skinning weights and derivatives have been computed. The required position of the proxy vertex in each example is also estimated using Eq. (4). When building the model, we limit the number of control parameters that can influence a single vertex and thus the number of new proxy vertices is small. The overall computation is greatly reduced since we use a $\mathbf{G}$ matrix that evaluates deformation gradients only for the proxy vertices.

Vertex Constraints. Direct manipulation of mesh vertices provides an intuitive framework for posing the mesh. Vertices positioned by the user supply constraints that limit the degrees of freedom of the control transformations. We express the posing problem according to the constrained optimization:

$$\mathbf{t}^*, \beta^* = \arg\min_{\mathbf{t},\beta} \|\mathbf{G}\mathbf{t} - \mathbf{m}(\beta)\|^2 \quad \text{such that} \quad \mathbf{C}\mathbf{t} = \mathbf{b}. \quad (5)$$

The user-controlled mesh vertices are expressed as linear equality constraints in $\mathbf{C}\mathbf{t} = \mathbf{b}$ where the vector $\mathbf{b}$ stacks the constrained vertex positions and the rows of the matrix $\mathbf{C}$ express the deformed vertex positions in terms of the coefficients shown in Eq. (1). While the objective function evaluates deformation gradients only for proxy vertices, the user is free to constrain any of the original mesh vertices. If the constraint matrix is overdetermined, the control parameters can be computed directly with a least-squares solution. In practice, this case arises only when many vertices are constrained over the entire mesh. A more common situation for mesh posing occurs at the opposite extreme when only a few constraints are

specified and one out of many valid mesh configurations must be selected [Grochow et al. 2004; Sumner et al. 2005].

We reformulate this problem into an unconstrained optimization by describing the space of all possible solutions:

$$\mathbf{t} = \mathbf{C}^{+}\mathbf{b} + \mathbf{N}\mathbf{t}_1, \tag{6}$$

where the matrices $\mathbf{C}^{+}$ and $\mathbf{N}$ are the pseudoinverse and the null space basis of the constraint matrix respectively, both computed via singular value decomposition. The vector $\mathbf{t}_1$ consists of free variables that parameterize the space of valid mesh configurations.

Inserting this expression into the constrained optimization of Eq. (5) allows us to compute the free control variables, yielding the control parameters that closely match the deformation gradients suggested by the example deformations:

$$\mathbf{t}_1^*, \beta^* = \arg\min_{\mathbf{t}_1, \beta} \|\mathbf{G}\mathbf{N}\mathbf{t}_1 - (\mathbf{m}(\beta) - \mathbf{G}\mathbf{C}^{+}\mathbf{b})\|^2. \tag{7}$$

An efficient numerical solution of this optimization problem is possible because the reduced model parameters comprise the unknowns and because the resolution-independent objective function need not evaluate deformation gradients at every mesh vertex.

3.3 Numerical Methods

The nonlinear optimization in Eq. (7) solves for the pose that best matches the given examples, subject to user-specified vertex constraints. This formulation can be solved using an iterative Gauss-Newton algorithm [Gill et al. 1989]. The most critical step in each iteration is a solution to a dense set of normal equations, whose block Cholesky factorization enables efficient solution after appropriate precomputation.

Gauss-Newton Iteration. The optimization is nonlinear because $\mathbf{m}(\beta)$ combines the rotation portion of the deformation gradients nonlinearly. The Gauss-Newton method linearizes this function in each iteration to compute the change in blending weights, δ, by solving the normal equations:

$$\mathbf{A}^{\top}\mathbf{A}\begin{bmatrix}\mathbf{t}_1 \\ \delta\end{bmatrix} = \mathbf{A}^{\top}\left(\mathbf{m}(\beta) - \mathbf{G}\mathbf{C}^{+}\mathbf{b}\right), \tag{8}$$

where the dense matrix $\mathbf{A}$ is defined by:

$$\mathbf{A} = \left[\begin{array}{c|c} \mathbf{G}\mathbf{N} & -\mathrm{D}_{\beta}\mathbf{m}(\beta) \end{array}\right]. \tag{9}$$

The Gauss-Newton method iterates until the optimization converges, which is measured by observing the objective function, its gradient, and the change in blending weights. However, efficient optimization requires precomputing the Cholesky factorization of the largest block in the dense $\mathbf{A}$ matrix.

Cholesky Factorization. The size of $\mathbf{A}$ is linear in the number of proxy vertices. Although the number of control transformations is typically quite small (between 20 and 80 in our examples), the number of proxy vertices can be in the hundreds, preventing efficient solution without a carefully designed numerical method. A block Cholesky factorization, however, allows us to precompute the factors of the largest block in our system matrix:

$$\mathbf{U}_{11}^{\top}\mathbf{U}_{11} = \mathbf{N}^{\top}\mathbf{G}^{\top}\mathbf{G}\mathbf{N}, \tag{10}$$

because its value is constant for a given set of vertex constraints in every iteration of the Gauss-Newton algorithm. The remaining terms of the block Cholesky factorization,

$$\mathbf{A}^{\top}\mathbf{A} = \begin{bmatrix}\mathbf{U}_{11}^{\top} & \mathbf{0} \\ \mathbf{U}_{12}^{\top} & \mathbf{U}_{22}^{\top}\end{bmatrix}\begin{bmatrix}\mathbf{U}_{11} & \mathbf{U}_{12} \\ \mathbf{0} & \mathbf{U}_{22}\end{bmatrix},$$

are computed efficiently in each iteration. The block matrix $\mathbf{U}_{12}$ is computed with backsubstitution and the block matrix $\mathbf{U}_{22}$ is computed with the Cholesky factorization of a small dense matrix whose size depends on the number of example shapes. Given these Cholesky factors, the normal equations are solved efficiently with two solutions to triangular linear systems. The complexity of each solver iteration is $O(pce + e^3)$, where p is the number of proxy vertices, c the number of control transformations, and e the number of example shapes. When the number of proxy vertices is large compared to the number of examples, the computational penalty of adding additional examples (up to 27 in our experiments) is a small fraction of the total cost.

4 Results

Prior to interactive posing, our procedure learns a reduced deformable model from a few user-supplied examples. It relies on a skinning technique that recognizes near-rigid deformations in example shapes [James and Twigg 2005]. All reduced deformable models in our experiments were generated by the same four-step procedure. First, mean-shift clustering forms near-rigid groups of triangles that express similar rotation sequences in the example poses. Second, each near-rigid component is associated with an affine transformation that optimally deforms the triangle centroids in each example. Third, a small set of control transformations is selected for each vertex according to an error metric that measures the influence of that control on the vertex. And fourth, the affine skinning weights are computed with a solution to the non-negative least-squares optimization that maximizes the fit to example shapes. It is known that correct shape interpolation using deformation gradients may require that some rotations be manually adjusted to include rotation in a different range of 2π [Sumner et al. 2005; Sumner 2005]. We chose example shapes that did not require modification.

Once the reduced deformable model is constructed, our procedure estimates the proxy vertices needed for the resolution-independent formulation. Proxy vertices, which summarize the coordinated vertex movement, are computed for each vertex group according to Eq. (4). The third and fourth steps of the model construction process are repeated to compute the skinning weights and their derivatives for each proxy vertex.

Mesh Posing. The resolution-independent formulation supports fast and fluid mesh posing independent of the geometric complexity. The animator selects a few mesh vertices, drags them with a mouse-based interface, and the entire shape is posed in a natural way as suggested by the examples. The real-time video on the conference DVD demonstrates the ease of interaction in this system.

In Figure 3 (A), our system learns the natural space of deformations for a horse mesh from the provided examples. The user selects a handle on two hooves and moves them to deform the mesh. The horse bends its legs at the appropriate joints and extends all its limbs when the handles encourage the horse into a galloping pose. Similar interaction is shown in Figure 3 (B), in which an elephant mesh with over 42,000 vertices is posed at interactive rates. The user constrains the elephant's feet and trunk to induce an expressive charging pose. Such fluid interaction would not be possible without our resolution-independent formulation.

Our system is particularly powerful for meshes that exhibit diverse and expressive deformations, such as the human character shown in Figure 3 (C). A set of scanned and reconstructed human shapes [Anguelov et al. 2005] allows our system to easily learn a space of natural human deformations. By selecting a few vertex handles such as on the hands or feet, the entire character is posed intuitively based on the handle positions. The limbs bend, flex, and extend naturally according to the constraints, and subtle deformations such as muscle bulges are implicitly captured by draw-

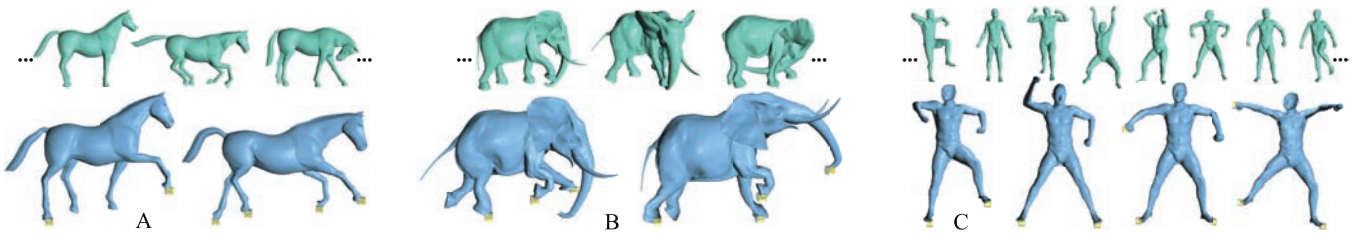

Figure 3: Top row: Example shapes for three different meshes. Bottom row: New meshes interactively posed using our system. (A) A few vertex handles are selected on the horse's hooves to induce the animal into stepping and galloping poses. (B) An elephant with over 42,000 vertices is sculpted into a charging pose at equivalent interactive rates. (C) Scanned human shapes allow our system to learn the space of natural human poses. The posed character's limbs bend and extend naturally to accommodate the user-provided vertex constraints.

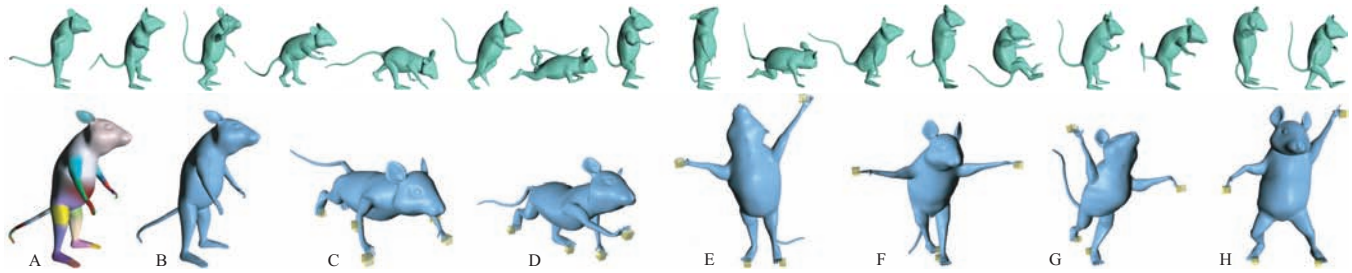

Figure 4: Top row: Mouse example poses. (A) Control transformations are inferred from the examples. (C–E) Constraining several vertices poses the mouse into a push-up position, followed by a crawl and a grandstand. (F–G) The mouse is posed as if on a balance beam and then falling off. (H) A classic disco pose is induced by only four handle vertices.

Mesh	Verts	Ex	Ct	Pxy	Total	Solve	MIK	$E_{full}(E_{proxy})$
Horse	8431	12	33	195	0.019	0.007	0.424	3.01(3.00)
Human	12500	10	42	471	0.031	0.013	0.675	6.01(5.95)
Mouse	13188	26	48	278	0.037	0.019	1.351	1.63(1.73)
Dragon	15560	7	72	380	0.046	0.023	0.590	0.96(0.97)
Gorilla	25436	27	28	198	0.045	0.013	2.962	4.38(4.33)
Elephant	42321	11	23	210	0.058	0.008	1.928	5.38(5.46)

Table 1: Number of vertices, examples, control transformations, and proxy vertices in our meshes. We show the computation time for our solver compared to that of Sumner and colleagues (MIK) [2005]. All timings are in seconds. Comparison of percent error distortion E between a full vertex representation and usage of proxy vertices shows that the latter is comparably accurate.

ing upon the examples. Our abstract, non-hierarchical, non-rigid controls can capture these complex deformations much more faithfully than could inverse kinematics on a conventional, rigid skeleton [Anguelov et al. 2005; James and Twigg 2005].

Other anthropomorphic characters are successfully manipulated using our system with highly expressive results, such as the gorilla in Figure 1 and the mouse in Figure 4. In the case of both characters, meaningful deformations are represented by both bipedal and quadrupedal poses. The user can deform the mesh anywhere in this space of natural poses by constraining several vertices. In (4 C–D), the mouse is manipulated into a push-up pose and then the handles are moved to change the pose into a quadrupedal crawl. Lifting a single vertex on each hand moves the character into a bipedal grandstanding pose (4 E). Direct manipulation of vertices easily aligns the mouse's feet, as if the character were walking on a balance beam (4 F). Further manipulation makes the mouse look as though it just fell off (4 G). Finally, only a few vertex handles are required to induce a classic disco pose (4 H).

Resolution-Independent Interaction. Table 1 provides statistics about our experiments, including the number of mesh vertices, examples, control transformations, and proxy vertices. The running times provided are given for one iteration of our nonlinear solver; all timing was measured on a 3.4 GHz Pentium 4 PC with 2GB of RAM. The "Solve" column indicates the amount of time needed for all computation in one Gauss-Newton iteration excluding vertex skinning and display, while the "Total" column indicates the total time for one iteration. Convergence was typically reached in 3 to 7 iterations. These timings reflect the fact that the computation is proportional only to the number of examples and proxy vertices, which is independent of the mesh's geometric complexity. In the case of the horse and elephant, both quadrupeds possess similar movement complexity, and thus their solve times are nearly identical even though the elephant's geometric complexity is far greater.

We use a software implementation of the skinning algorithm to position the vertices after the transformations have been found, but this functionality is greatly accelerated when implemented in hardware via matrix palette skinning [Lindholm et al. 2001]. Finally, we compare one iteration of our solving time to one Gauss-Newton iteration of the mesh-based inverse kinematics solver (MIK) [Sumner et al. 2005] used with the same mesh and examples. In all cases, our solver is at least an order of magnitude faster. The elephant example requires merely 8ms for one iteration, compared with nearly two seconds for mesh-based inverse kinematics. Similarly, the SCAPE algorithm [Anguelov et al. 2005] requires approximately one second to generate new meshes from the set of human poses, whereas our system allows posing and shape blending more than an order of magnitude faster.

We validate our approximations of the example meshes by treating them as a mesh sequence and measuring error in terms of percent distortion [Karni and Gotsman 2004], shown in the last column of Table 1. Example reconstruction using deformation gradients for all vertices incurs error due to approximation with controls. There is no significant difference in error using proxy vertices.

Highly Non-Rigid Deformations. Many control parameters are required to faithfully capture highly non-rigid deformations, such as the dragon's tail which curls smoothly (Figure 5). Mean-shift parameters can be adjusted to yield more controls, better reproducing deformations. Though many controls will slow our method, it

will remain faster than MeshIK [Sumner et al. 2005] until deformations are so complex that controls express deformations no more compactly than individual vertices.

Figure 5: A storybook dragon, requiring many controls in the highly non-rigid tail and neck, is deformed into several poses.

5 Conclusion

Reduced deformable models provide a means to separate the ever increasing geometric complexity of animated meshes from the complexity of their movement. Although current models are effective for compression and hardware rendering, they impede user control since the model parameters often do not match the user's semantic understanding of character movement. Our work provides interactive control of reduced deformable models via an intuitive inverse-kinematics framework. A collection of transformations compactly represents articulated character movement and can be derived automatically from example data. We formulate the inverse kinematics problem in this reduced space to achieve resolution-independent performance: the speed of the posing task is a function of the model parameters rather than of character geometry.

Our choice of control transformations as a reduced deformable model incurs some limitations. Noise and cloth wrinkles in the scanned human dataset yield a dense sampling of controls. Our method would benefit from a more accurate and controllable way to estimate transformations. Furthermore, since our overall optimization framework applies to alternate reduced models, exploring different ones may lead to more efficient formulations or solutions appropriate for different types of deformations.

While we show how to separate deformation complexity from geometric complexity for efficient mesh posing, movement and geometry are not entirely separate: the reduced deformable model is tailored to a particular mesh's shape and mesh structure. Future work might develop ways to automatically extract a reduced deformable model that encapsulates one character's movement, and apply it to a different character. A new mesh could be posed without needing to repeat the process of example creation and model estimation.

6 Acknowledgements

We would like to thank Dragomir Anguelov for the human dataset, the Rutgers Image Understanding Laboratory for the mean-shift implementation, and the anonymous reviewers for their feedback. This work was partially supported by the John Reed UROP Fund and a donation from Pixar Animation Studios.

References

ALEXA, M., AND MÜLLER, W. 2000. Representing animations by principal components. *Computer Graphics Forum 19*, 3 (Aug.), 411–418.

ALEXA, M., COHEN-OR, D., AND LEVIN, D. 2000. As-rigid-as-possible shape interpolation. In *Proceedings of ACM SIGGRAPH 2000*, Annual Conference Series, 157–164.

ANGUELOV, D., SRINIVASAN, P., KOLLER, D., THRUN, S., RODGERS, J., AND DAVIS, J. 2005. SCAPE: shape completion and animation of people. *ACM Transactions on Graphics 24*, 3 (Aug.), 408–416.

BARR, A. H. 1984. Global and local deformations of solid primitives. In *Computer Graphics (Proceedings of ACM SIGGRAPH 84)*, vol. 18, 21–30.

GILL, P. E., MURRAY, W., AND WRIGHT, M. H. 1989. *Practical Optimization*. Academic Press, London.

GROCHOW, K., MARTIN, S. L., HERTZMANN, A., AND POPOVIĆ, Z. 2004. Style-based inverse kinematics. *ACM Transactions on Graphics 23*, 3 (Aug.), 522–531.

JAMES, D. L., AND PAI, D. K. 2002. DyRT: Dynamic response textures for real time deformation simulation with graphics hardware. *ACM Transactions on Graphics 21*, 3 (July), 582–585.

JAMES, D. L., AND TWIGG, C. D. 2005. Skinning mesh animations. *ACM Transactions on Graphics 24*, 3 (Aug.), 399–407.

KARNI, Z., AND GOTSMAN, C. 2004. Compression of soft-body animation sequences. *Computers & Graphics 28*, 1, 25–34.

KOBBELT, L., CAMPAGNA, S., VORSATZ, J., AND SEIDEL, H.-P. 1998. Interactive multi-resolution modeling on arbitrary meshes. In *Proceedings of ACM SIGGRAPH 98*, Annual Conference Series, 105–114.

LEWIS, J. P., CORDNER, M., AND FONG, N. 2000. Pose space deformations: A unified approach to shape interpolation and skeleton-driven deformation. In *Proceedings of ACM SIGGRAPH 2000*, Annual Conference Series, 165–172.

LINDHOLM, E., KILGARD, M. J., AND MORETON, H. 2001. A user-programmable vertex engine. In *Proceedings of ACM SIGGRAPH 2001*, Computer Graphics Proceedings, Annual Conference Series, 149–158.

LIPMAN, Y., SORKINE, O., COHEN-OR, D., LEVIN, D., RÖSSL, C., AND SEIDEL, H.-P. 2004. Differential coordinates for interactive mesh editing. In *Proceedings of Shape Modeling International*, 181–190.

LIPMAN, Y., SORKINE, O., LEVIN, D., AND COHEN-OR, D. 2005. Linear rotation-invariant coordinates for meshes. *ACM Trans. Graph. 24*, 3, 479–487.

LOUNSBERY, M., DEROSE, T. D., AND WARREN, J. 1997. Multiresolution analysis for surfaces of arbitrary topological type. *ACM Transactions on Graphics 16*, 1 (Jan.), 34–73.

MOHR, A., AND GLEICHER, M. 2003. Building efficient, accurate character skins from examples. *ACM Transactions on Graphics 22*, 3 (July), 562–568.

PENTLAND, A., AND WILLIAMS, J. 1989. Good vibrations: Modal dynamics for graphics and animation. In *Computer Graphics (Proceedings of SIGGRAPH 89)*, vol. 23, 215–222.

SLOAN, P.-P. J., ROSE, III, C. F., AND COHEN, M. F. 2001. Shape by example. In *Symposium on Interactive 3D Graphics (I3D)*, ACM Press, 135–144.

SORKINE, O. 2005. State-of-the-art report: Laplacian mesh processing. In *Eurographics 2005—State of the Art Reports*, The Eurographics Association, Dublin, Ireland, Eurographics, 53–70.

SUMNER, R. W., AND POPOVIĆ, J. 2004. Deformation transfer for triangle meshes. *ACM Transactions on Graphics 23*, 3 (Aug.), 399–405.

SUMNER, R. W., ZWICKER, M., GOTSMAN, C., AND POPOVIĆ, J. 2005. Mesh-based inverse kinematics. *ACM Transactions on Graphics 24*, 3 (Aug.), 488–495.

SUMNER, R. W. 2005. *Mesh Modification Using Deformation Gradients*. PhD thesis, Massachusetts Institute of Technology.

YU, Y., ZHOU, K., XU, D., SHI, X., BAO, H., GUO, B., AND SHUM, H.-Y. 2004. Mesh editing with poisson-based gradient field manipulation. *ACM Transactions on Graphics 23*, 3 (Aug.), 644–651.

ZHANG, L., SNAVELY, N., CURLESS, B., AND SEITZ, S. M. 2004. Spacetime faces: high resolution capture for modeling and animation. *ACM Transactions on Graphics 23*, 3 (Aug.), 548–558.

ZORIN, D., SCHRÖDER, P., AND SWELDENS, W. 1997. Interactive multiresolution mesh editing. In *Proceedings of ACM SIGGRAPH 97*, Annual Conference Series, 259–268.