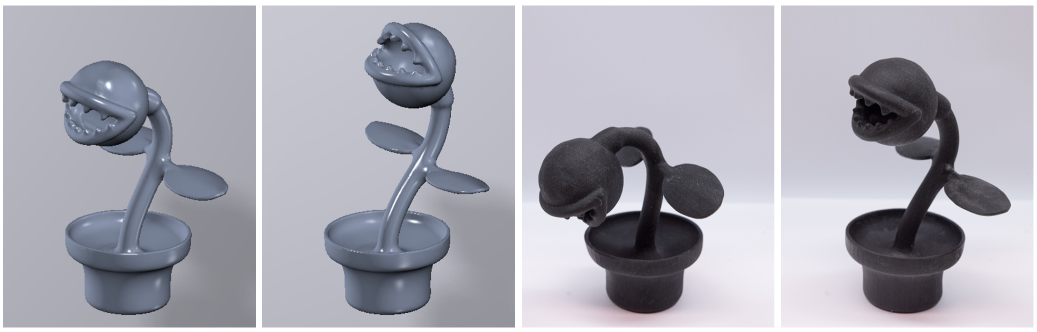
Interactive Two-Way Shape Design of Elastic Bodies

RAJADITYA MUKHERJEE, The Ohio State University
LONGHUA WU, The Ohio State University
HUAMIN WANG, The Ohio State University



(a) The target shape (b) The rest shape (c) The model made in the target shape (d) The model made in the rest shape

Fig. 1. Based on fast simulation and inverse simulation techniques, our system allows the designer to interactively edit and examine the quasistatic shape and the rest shape of a piranha model at the same time, as shown in (a) and (b). We show physical validation of our system by fabricating the model in both shapes using a rubber-like tango black material. Under the influence of gravity, the model fabricated in the rest shape in (b) sags to the desired quasistatic shape in (a), as predicted by our system.

We present a novel system for interactive elastic shape design in both forward and inverse fashions. Using this system, the user can choose to edit the rest shape or the quasistatic shape of an elastic solid, and obtain the other shape that matches under the quasistatic equilibrium condition at the same time. The development of this system is based on the discovery that inverse quasistatic simulation can be immediately solved by Newton's method with a direct solver. To implement our simulator, we propose a Jacobian matrix evaluation scheme for the inverse elastic problem and we present step length and matrix evaluation techniques that improve the simulation performance. While our simulator is efficient, it is still not fast enough for the system to generate the result in real time. Our solution is a shape initialization method using the recent projective dynamics technique. Shape initialization not only works as a fast preview function during the user editing process, but also speeds up the convergence of quasistatic or inverse quasistatic simulation afterwards. The use of a heterogeneous algorithm structure allows the system to further reduce its preview cost, by utilizing the power of both the CPU and the GPU. Our experiment demonstrates that the whole system is fast, robust, and convenient for the designer to use in both forward and inverse elastic shape design. It can handle a variety of nonlinear elastic material models, and its runtime performance has space for more improvement.

Authors' addresses: Rajaditya Mukherjee, The Ohio State University, rajaditya.mukherjee@gmail.com; Longhua Wu, The Ohio State University, wu.2724@osu.edu; Huamin Wang, The Ohio State University, wang.3602@osu.edu.

2577-6193/2018/5-ART11 $15.00
https://doi.org/10.1145/3203196

CCS Concepts: • **Computing methodologies → Physical simulation**;

Additional Key Words and Phrases: Quasistatic simulation, Newton's method, Inverse elastic problem, Heterogeneous structure, Hyperelasticity

ACM Reference Format:
Rajaditya Mukherjee, Longhua Wu, and Huamin Wang. 2018. Interactive Two-Way Shape Design of Elastic Bodies. *Proc. ACM Comput. Graph. Interact. Tech.* 1, 1, Article 11 (May 2018), 16 pages. https://doi.org/10.1145/3203196

1 INTRODUCTION

An elastic body deforms under external loads. This property makes elastic shape design uniquely challenging, due to the discrepancy between the shapes before and after applying loads. Traditionally, elastic shape design is handled in a forward fashion: the designer first models the body without considering external loads, and then runs quasistatic simulation to check how the body gets deformed under loads. The designer must repeat this process multiple times, until he/she becomes satisfied with the outcome. A forward elastic shape design system is straightforward to implement, given the fact that quasistatic simulation has been extensively studied for many years. However, the repeating process requires heavy user intervention and consumes considerable computational resources. Forward elastic shape design is also counter-intuitive, since the designer cannot directly interact the actual shape under the influence of external loads.

A more natural and intuitive strategy is to allow the designer to manipulate the deformed shape under influence first, and then estimate the rest shape that achieves the deformed shape later. This inverse design strategy is useful not only to designers, but also to animators and surgeons, who need to know the rest shapes of sagging objects captured from the real world, for animation production and virtual surgery. Unfortunately, inverse elastic shape design requires to solve the inverse problem of quasistatic simulation, which turns out to be significantly more difficult than quasistatic simulation itself. Previous research on this topic was largely focused on lower dimensional cases [8, 10], specific elastic material models [5, 32], or small deformation [23]. Inverse quasistatic simulation of generic elastic models remains as a challenging problem, as far as we know.

In this paper, we present a new elastic shape design system that utilizes the power of both the CPU and the GPU. The key features of our system are:

- **Two-way design.** The system enables elastic shape design in both forward and inverse ways. In other words, the designer can modify either the rest shape or the quasistatic shape, and the system generates the other shape on the fly.
- **Flexibility.** Our system flexibly handles a wide range of nonlinear elastic models, including the neo-Hookean model, the Mooney-Rivlin model, and the spline-based model using principal stretches [36].
- **Efficiency.** After user editing, the system typically finishes the modeling process of a mesh with 73K tetrahedra in 0.5 seconds, as Figure 1 shows. This is substantially faster than any existing system, especially for inverse shape design.

The technical contributions involved in the development of our system can be summarized as follows.

- **Inverse quasistatic simulation.** We present the Jacobian matrix evaluation scheme for inverse quasistatic simulation of generic elastic materials. Based on this scheme, we discovered that, inverse simulation, as a special shape optimization problem, can be directly solved by Newton's method.

- **Initialization by projective dynamics.** Although our simulators are efficient, they are still unable to follow fast shape changes made by the user. To address this problem, we propose a real-time shape initialization approach based on the recent projective dynamics technique.
- **A heterogeneous system.** To achieve interactive system performance, we develop a heterogenous structure that distributes the computational workload to both the CPU and the GPU. The use of this structure offers another level of parallelization among processors.

Our experiment demonstrates that the whole system is fast, robust, and convenient for the designer to use in both forward and inverse shape design cases. It flexibly handles a variety of nonlinear elastic material models, and its performance can be potentially improved by other numerical techniques in the future, such as multi-grid.

2 RELATED WORK

Elastic body simulation. Elastic body simulation has been an important graphics research topic, since the pioneer work by Terzopoulos and colleagues [28]. Today's physics-based simulators often use implicit time integration, since explicit time integration can cause the numerical instability issue [18, 26]. To use implicit time integration, a simulator needs the Jacobian matrix of the force with respect to the vertex position. Teran and colleagues [27] proposed a Jacobian matrix evaluation scheme for hyperelastic materials and applied it in quasistatic simulation. Another Jacobian matrix scheme based on principal stretches was proposed by Xu and collaborators [36].

Physics-based models are flexible and close to real-world materials, but their computational costs are not so affordable by real-time applications. Because of that, researchers are getting more interested in fast constraint-based simulation techniques, such as strain limiting [21, 29, 34], shape matching [17, 22], and position-based dynamics [12, 15, 16]. Liu and collaborators [13] found that the potential energy of a spring can be interpreted as a compliant edge constraint. This observation guided them to the development of an iterative local/global mass-spring simulator, whose result converges to the solution of implicit time integration. Bouaziz and colleagues [4] generalized this idea into projective dynamics, by formulating the elastic energy of an element as a geometric constraint. For highly stiff problems, Tournier and colleagues [30] combined forces and constraints into a better conditioned linear system with a larger problem size. To implement projective dynamics on the GPU, Wang [33] explored the Jacobi-preconditioned gradient descent method, and Fratarcangeli and colleagues [9] advocated the use of multi-color Gauss Seidel. Recently, Wang and Ying [35] generalized Jacobi-preconditioned gradient descent to accurately and efficiently simulate generic nonlinear elastic materials, thanks to a series of divergence avoidance techniques.

Elastic shape design. While forward elastic shape design can be solved by quasistatic simulation as many researchers explored, inverse elastic shape design, fundamentally as inverse quasistatic simulation, is a much more complex problem. If the elastic shape is made of strands and curves, this problem can be well solved under reduced dynamics models [8, 10]. When the goal does not require the rest shape result to be strictly consistent with the deformed shape input, the problem can be solved by augmented Lagrangian methods as Skouras and collaborators [24] suggested. For inverse quasistatic simulation of neo-Hookean volumetric models, Chen and colleagues [5] developed a Jacobian matrix evaluation formula and suggested to use an asymptotic numerical method. Wang and collaborators [32] found that the inverse problem can be solved as a backward simulation process for co-rotational linear materials. Shin and colleagues [23] studied inverse quasistatic simulation of generic elastic materials, when the deformation is small. Recently, Bartle and collaborators [1] designed a fixed point method to solve the inverse problem for cloth patches. Their method converges fast mostly in the first few iterations.

Graphics researchers have also studied inverse elastic shape design, when the shape is parameterized by edge lengths [31], control parameters [2, 7], or joints [20, 25]. Their techniques typically involve the use of the Gauss-Newton method, by formulating shape optimization as nonlinear least square problems. For inverse quasistatic simulation, Chen and colleagues [5] pointed out that the Gauss-Newton method is not a suitable option, since it is often too expensive to construct its matrix, especially when the mesh is in high resolution. Outside of the computer graphics community, researchers investigated the inverse elastic problem mostly for identifying material parameter distributions or buried objects [3]. The developed techniques are either limited to linear elasticity, or too computationally expensive for interactive applications.

3 BACKGROUND ON QUASISTATIC SIMULATION

Let $\mathbf{X} \in \mathbb{R}^{3N}$ and $\mathbf{x} \in \mathbb{R}^{3N}$ be rest and deformed vertex position vectors of an elastic body with N vertices. The quasistatic equilibrium state can be described using a nonlinear system:

$$\mathbf{f}(\mathbf{x}, \mathbf{X}) = \mathbf{0}, \tag{1}$$

in which the total force $\mathbf{f}$ is a nonlinear function of $\mathbf{x}$ and $\mathbf{X}$. Given $\mathbf{X}$, the goal of quasistatic simulation is to find $\mathbf{x}$ that satisfies Equation 1.

Numerical solutions. A typical solution to a nonlinear system is Newton's method, which successively refines the result $\mathbf{x}^{(k+1)}$ in the k+1-th iteration as:

$$\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} - \left[\mathbf{J}_\mathbf{x}^{(k)}\right]^{-1} \mathbf{f}(\mathbf{x}^{(k)}, \mathbf{X}), \tag{2}$$

where $\mathbf{J}_\mathbf{x} = \partial \mathbf{f} / \partial \mathbf{x}$ is the Jacobian matrix of $\mathbf{f}$ with respect to $\mathbf{x}$. Since we can define $\mathbf{f}$ as the negative gradient of the total potential energy, we can treat $\mathbf{J}_\mathbf{x}$ as the negative Hessian matrix of the potential energy. Therefore, $\mathbf{J}_\mathbf{x}$ must be symmetric.

The issue with Newton's method is that it is too computationally expensive to solve $\mathbf{J}_\mathbf{x}^{-1}\mathbf{f}$ in every iteration. A natural idea explored by quasi-Newton methods and conjugate gradient methods is to use an approximation of $\mathbf{J}_\mathbf{x}$, whose inverse can be easily calculated. These methods often require extensive use of dot product operations, making them unfriendly with parallelization. In a special case, if we replace $\mathbf{J}_\mathbf{x}$ by the identity matrix, we reduce Newton's method to the gradient descent method, whose convergence rate is known to be low. Alternatively, we can replace $\mathbf{J}_\mathbf{x}$ by a carefully tuned constant matrix, which can be pre-factored as a fast runtime solver. This approach has demonstrated its effectiveness on projective dynamics [4], but not so much on generic hyperelastic models [14]. Recently, Wang and Yang [35] proposed to replace $\mathbf{J}_\mathbf{x}$ by its diagonal. Their method can robustly and efficiently handle generic hyperelastic models, after using a dynamically adjusted step length.

Elastic force and matrix evaluation. A crucial step involved in many aforementioned methods is the evaluation of the internal elastic force and its Jacobian matrix. According to continuum mechanics, the elastic forces $\{\mathbf{g}_0, \mathbf{g}_1, \mathbf{g}_2, \mathbf{g}_3\}$ applied at the vertices of a tetrahedron is:

$$\mathbf{G} = [\mathbf{g}_1 \;\; \mathbf{g}_2 \;\; \mathbf{g}_3] = \mathbf{P}\mathbf{B}_m = \mathbf{P}\left[\mathbf{b}_1 \;\; \mathbf{b}_2 \;\; \mathbf{b}_3\right], \qquad \mathbf{g}_0 = -\sum_{i=1}^{3} \mathbf{g}_i, \tag{3}$$

in which $\mathbf{P}$ is the first Piola-Kirchhoff stress and $\mathbf{b}_i$ stands for the average area-weighted normal of the triangles adjacent to vertex i. Let $\mathbf{F} = \mathbf{D}_s\mathbf{D}_m^{-1}$ be the deformation gradient, where $\mathbf{D}_s$ and $\mathbf{D}_m$ are the edge vector matrix: $\mathbf{D}_s = [\mathbf{x}_1 - \mathbf{x}_0 \;\; \mathbf{x}_2 - \mathbf{x}_0 \;\; \mathbf{x}_3 - \mathbf{x}_0]$ and $\mathbf{D}_m = [\mathbf{X}_1 - \mathbf{X}_0 \;\; \mathbf{X}_2 - \mathbf{X}_0 \;\; \mathbf{X}_3 - \mathbf{X}_0]$. After singular value decomposition $\mathbf{F} = \mathbf{U}\mathbf{D}\mathbf{V}^\mathsf{T}$, we can compute the stress tensor $\mathbf{P}$ as:

$$\mathbf{P} = \mathbf{U}\mathbf{P}(\mathbf{D})\mathbf{V}^\mathsf{T}, \tag{4}$$

where the function $\mathbf{P}(\mathbf{D})$ represents the constitutive relationship between the principal stretches in $\mathbf{D}$ and the stress tensor. Given Equation 3 and 4, Xu and colleagues [36] proposed to evaluate the elastic Jacobian matrix from:

$$\frac{\partial \mathbf{G}}{\partial \mathbf{x}} = \frac{\partial \mathbf{G}}{\partial \mathbf{F}}\frac{\partial \mathbf{F}}{\partial \mathbf{x}} = \left(\frac{\partial \mathbf{P}}{\partial \mathbf{F}}\mathbf{B}_m\right)\frac{\partial \mathbf{F}}{\partial \mathbf{x}}. \tag{5}$$

Since both $\mathbf{B}_m$ and $\partial\mathbf{F}/\partial\mathbf{x}$ are invariant to $\mathbf{x}$, our focus is on $\partial\mathbf{P}/\partial\mathbf{F}$. By definition, we can formulate the derivative of $\mathbf{P}$ with respect to one variable of $\mathbf{F}$ as:

$$\frac{\partial \mathbf{P}}{\partial F_{ij}} = \mathbf{U}\left\{\Omega_{\mathbf{U}}^{ij}\mathbf{P}(\mathbf{D}) + \frac{\partial \mathbf{P}(\mathbf{D})}{\partial \mathbf{D}}\frac{\partial \mathbf{D}}{\partial F_{ij}} + \mathbf{P}(\mathbf{D})\Omega_{\mathbf{V}}^{ij}\right\}\mathbf{V}^{\mathsf{T}}, \tag{6}$$

in which,

$$\Omega_{\mathbf{U}}^{ij} = \mathbf{U}^{\mathsf{T}}\frac{\partial \mathbf{U}}{\partial F_{ij}}, \quad \Omega_{\mathbf{V}}^{ij} = \frac{\partial \mathbf{V}^{\mathsf{T}}}{\partial F_{ij}}\mathbf{V}. \tag{7}$$

To evaluate Equation 6, we calculate $\mathbf{U}$, $\mathbf{V}$, $\mathbf{P}(\mathbf{D})$, and $\partial\mathbf{P}(\mathbf{D})/\partial\mathbf{D}$ by their definitions. The additional terms needed by Equation 6 are $\Omega_{\mathbf{U}}^{ij}$, $\Omega_{\mathbf{V}}^{ij}$, and $\partial\mathbf{D}/\partial F_{ij}$. According to [19], we have:

$$\mathbf{U}^{\mathsf{T}}\frac{\partial \mathbf{F}}{\partial F_{ij}}\mathbf{V} = \Omega_{\mathbf{U}}^{ij}\mathbf{D} + \frac{\partial \mathbf{D}}{\partial F_{ij}} + \mathbf{D}\Omega_{\mathbf{V}}^{ij}. \tag{8}$$

Since $\mathbf{D}$ is a diagonal matrix, $\partial\mathbf{D}/\partial F_{ij}$ is diagonal as well. Meanwhile, $\Omega_{\mathbf{U}}^{ij}$ and $\Omega_{\mathbf{V}}^{ij}$ must be skew-symmetric, because $\Omega_{\mathbf{U}}^{ij} + (\Omega_{\mathbf{U}}^{ij})^{\mathsf{T}} = \partial(\mathbf{U}^{\mathsf{T}}\mathbf{U})/\partial F_{ij} = \mathbf{0}$. Based on these observations, we obtain $\partial\mathbf{D}/\partial F_{ij}$ from the diagonal elements of $\mathbf{U}^{\mathsf{T}}(\partial\mathbf{F}/\partial F_{ij})\mathbf{V}$. The remaining off-diagonal elements form a linear system, whose solution gives the six unique elements of $\Omega_{\mathbf{U}}^{ij}$ and $\Omega_{\mathbf{V}}^{ij}$. To avoid the system from being singular when two or more principal stretches are close, we simply place a limit on the determinant of the system matrix.

4 INVERSE QUASISTATIC SIMULATION

In this section, we will study inverse quasistatic simulation, the key component in inverse elastic shape design. Specifically, given $\mathbf{x}$, we would like to find $\mathbf{X}$ that satisfies $\mathbf{f}(\mathbf{x},\mathbf{X}) = \mathbf{0}$. Before we discuss the applicability of the numerical methods, we will formulate the Jacobian matrix of the force with respect to the rest shape: $\mathbf{J}_{\mathbf{X}} = \partial\mathbf{f}/\partial\mathbf{X}$, based on our prior knowledge in Section 3.

4.1 Jacobian Matrix Evaluation

We consider two Jacobian matrices for the inverse problem: the Jacobian of the elastic force under a generic elastic material model and the Jacobian of the gravity force.

Elastic matrix evaluation. To evaluate the elastic Jacobian matrix for the inverse problem, our idea is to use the inverse of the deformation gradient:

$$\hat{\mathbf{F}} = \mathbf{F}^{-1} = \mathbf{D}_m\mathbf{D}_s^{-1} = \mathbf{V}\mathbf{D}^{-1}\mathbf{U}^{\mathsf{T}}, \tag{9}$$

which is a linear function of $\mathbf{X}$. We can obtain $\mathbf{J}_{\mathbf{X}}$ from:

$$\frac{\partial \mathbf{G}}{\partial \mathbf{X}} = \left(\frac{\partial \mathbf{P}}{\partial \hat{\mathbf{F}}}\mathbf{B}_m\right)\frac{\partial \hat{\mathbf{F}}}{\partial \mathbf{X}} + \mathbf{P}\frac{\partial \mathbf{B}_m}{\partial \mathbf{X}}, \tag{10}$$

where only $\partial\hat{\mathbf{F}}/\partial\mathbf{X}$ is invariant to $\mathbf{X}$. To evaluate the first component in Equation 10, we compute the derivative of $\mathbf{P}$ with respect to one variable of $\hat{\mathbf{F}}$:

$$\frac{\partial \mathbf{P}}{\partial \hat{F}_{ij}} = \mathbf{U}\left\{\hat{\Omega}_{\mathbf{U}}^{ij}\mathbf{P}(\mathbf{D}) + \frac{\partial \mathbf{P}(\mathbf{D})}{\partial(\mathbf{D}^{-1})}\frac{\partial(\mathbf{D}^{-1})}{\partial \hat{F}_{ij}} + \mathbf{P}(\mathbf{D})\hat{\Omega}_{\mathbf{V}}^{ij}\right\}\mathbf{V}^{\mathsf{T}}, \tag{11}$$

in which,

$$\hat{\Omega}_{\mathbf{U}}^{ij} = \mathbf{U}^{\mathsf{T}} \frac{\partial \mathbf{U}}{\partial \hat{F}_{ij}}, \quad \hat{\Omega}_{\mathbf{V}}^{ij} = \frac{\partial \mathbf{V}^{\mathsf{T}}}{\partial \hat{F}_{ij}} \mathbf{V}. \tag{12}$$

By taking the derivative of the singular value decomposition of $\hat{\mathbf{F}}$ and multiplying the result with $\mathbf{V}^{\mathsf{T}}$ and $\mathbf{U}$, we get:

$$\left(\mathbf{V}^{\mathsf{T}} \frac{\partial \hat{\mathbf{F}}}{\partial \hat{F}_{ij}} \mathbf{U}\right)^{\mathsf{T}} = \hat{\Omega}_{\mathbf{U}}^{ij} \mathbf{D}^{-1} + \frac{\partial \mathbf{D}^{-1}}{\partial \hat{F}_{ij}} + \mathbf{D}^{-1} \hat{\Omega}_{\mathbf{V}}^{ij}. \tag{13}$$

As before, $\partial \mathbf{D}^{-1} / \partial \hat{F}_{ij}$ is diagonal, and both $\hat{\Omega}_{\mathbf{U}}^{ij}$ and $\hat{\Omega}_{\mathbf{V}}^{ij}$ are skew-symmetric. Therefore, we calculate $\partial \mathbf{D}^{-1} / \partial \hat{F}_{ij}$ from the diagonal elements of $(\mathbf{V}^{\mathsf{T}} \partial \hat{\mathbf{F}} / \partial \hat{F}_{ij} \mathbf{U})^{\mathsf{T}}$, and $\hat{\Omega}_{\mathbf{U}}^{ij}$ and $\hat{\Omega}_{\mathbf{V}}^{ij}$ from the remaining off-diagonal elements after solving a linear system.

The evaluation of the second component in Equation 10 needs $\partial \mathbf{B}_m / \partial \mathbf{X}$. Since $\mathbf{B}_m$ is made of area-weighted triangle normals and each area-weighted normal is equal to the cross product of two edge vectors, we formulate the derivative of one normal first:

$$\frac{\partial (\mathbf{X}_3 - \mathbf{X}_2) \times (\mathbf{X}_1 - \mathbf{X}_2)}{\partial \mathbf{X}_1} = \mathbf{R}_{32}^{*}, \tag{14}$$

in which $\mathbf{R}_{32}^{*}$ is the skew-symmetric cross product matrix of edge $\mathbf{X}_3\mathbf{X}_2$. Once we obtain all of these cross product matrices, we assemble them into $\partial \mathbf{B}_m / \partial \mathbf{X}$ and obtain the second component.

Gravitational matrix evaluation. The gravity force, as a function of the rest volume, also depends on $\mathbf{X}$. Therefore, we should consider its contribution to the Jacobian matrix of the total force as well. For simplicity, we use a lumped mass model, which equally distributes the mass of a tetrahedron to its four vertices. Based on this model, we calculate the Jacobian matrix of the gravity force contributed by tetrahedron t to every vertex i as:

$$\mathbf{J}_{\mathbf{X}}^{t,i} = \frac{\rho_t}{24} \mathbf{g} \frac{\partial \mathtt{det}(\mathbf{D}_m)}{\partial \mathbf{X}}, \tag{15}$$

in which ρ_t is the mass density, $\mathbf{g}$ is the gravity acceleration, and $\frac{1}{6}\mathtt{det}(\mathbf{D}_m)$ gives the tetrahedron volume. By definition, we can calculate the derivative of the determinant from its 2×2 sub-matrices. Once we calculate each $\mathbf{J}_{\mathbf{X}}^{t,i}$, we sum them up to form the whole gravitational Jacobian matrix.

The importance of matrix components. Our research reveals that the first component of the elastic matrix dominates the total Jacobian matrix. In average, it contributes more than 99.7 percent of the total Jacobian matrix, while the other two components contribute less than 0.3 percent. This phenomenon makes us wonder whether we can ignore the other components in inverse quasistatic simulation. Figure 2 demonstrates that the answer is no. When we ignore the second elastic component and the gravitational component, Newton's method fails to converge. When we ignore the gravitational component only, Newton's method converges within 34 iterations. Finally, when we use all of three components, Newton's method converges within 27 iterations. Since the computation overhead of the two additional components is small, we choose to evaluate the whole Jacobian matrix in practice.

4.2 Numerical Solutions

An interesting question is whether the numerical methods used for quasistatic simulation are still applicable to the inverse problem. Here we must keep in mind that the Jacobian matrix of the inverse problem cannot be interpreted as the Hessian matrix of a potential energy. Therefore, it may not be symmetric.

Our first experiment is to test nonlinear descent methods on the GPU, including nonlinear conjugate gradient and preconditioned gradient descent, both of which have demonstrated their

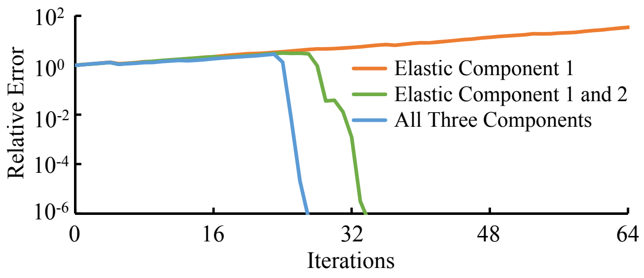


Fig. 2. The effects of Jacobian matrix components on the convergence of Newton's method. This plot shows that it is necessary to evaluate all of the three Jacobian matrix components, even though the contributions of the second elastic component and the gravitational component are small. By default, we choose the armadillo example for evaluation purposes in this paper.

robustness and performance in quasistatic simulation [35]. Figure 3 shows that gradient descent is effective in the first few iterations. But once the error becomes small, it diverges if we do not significantly reduce the step length. The performance of nonlinear conjugate gradient is even worse, probably because conjugate gradient is not suitable for asymmetric matrices. We note that here the cost of a gradient descent iteration is higher than that cost in [35], since gradient descent must evaluate the Jacobian matrix in every iteration to avoid more severe divergence issues.

Since some nonlinear descent methods can be considered as applying one step of an iterative solver in every iteration of Newton's method, our next experiment is to choose Newton's method and try different iterative solvers and different numbers of iterative steps. Figure 3 shows that if we use 128 steps of a Jacobi solver in each Newton iteration, the error curve behaves just like that of preconditioned gradient descent: the error decreases only in the first few iterations. The reason it runs faster is because it does not need frequent force or matrix evaluation. The asymmetric nature of the system matrix inspires us to test the biconjugate gradient stabilized method next. Figure 3 shows that this method still cannot reach a small error, even if we use 1,024 iterative steps per Newton iteration. It also verifies the high computational cost of the biconjugate gradient stabilized method on the GPU, due to extensive use of inner and matrix-vector products.

In the end, we believe that Newton's method with a direct solver provides the best way to solve inverse quasistatic simulation. Our simulator chooses to evaluate forces and matrices on the GPU, and solve each linear system by a LU solver on the CPU. In most cases, the simulator can reduce the error to a low level within one second, as Figure 3 shows. To further improve its robustness and efficiency, we will investigate error metrics, step lengths, and matrix evaluations in the rest of this subsection.

Errors and step lengths. When the initialization is away from the solution, Newton's method needs a sufficiently small step length to protect it from the divergence issue. One way to find the step length is to perform backtracking line search based on the Armijo rule, which requires the system energy to decrease monotonically. Since we cannot define inverse quasistatic simulation as an energy minimization problem, we replace the energy in the Armijo rule by the residual error, i.e., the magnitude of the residual force in Equation 1. We also change the rule to accept a step length, as long as it does not introduce too much error increase. The reason we do not strictly enforce an error decrease condition is because the error behaves differently from the energy and a strict condition can cause an unnecessarily small step length, as we experienced in our experiment. Figure 3 demonstrates that although the error reported from Newton's method increases in the first few iterations, it quickly drops afterwards.

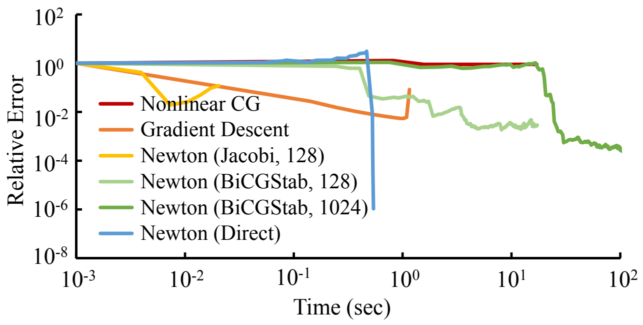


Fig. 3. The performances of different numerical methods solving the same inverse simulation problem. Due to the complex nature of the problem, many methods fail to converge. In contrast, Newton's method with a direct solver converges within one second.

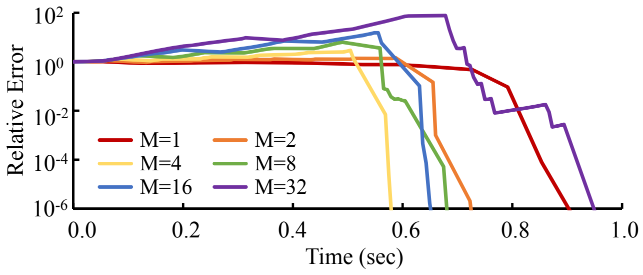


Fig. 4. The performances of Newton's method when it chooses different M values. Newton's method converges slowly when M is too small or too large. It converges the fastest when $M = 4$.

Skipping matrix evaluations. Newton's method spends a large portion of its computational time on Jacobian matrix evaluation and factorization. Therefore, a natural idea of improving its performance is to skip matrix evaluations and reuse factored matrices from previous iterations. Researchers have explored this idea as high-order Newton's methods [6] before. In this work, we simply perform matrix evaluation and factorization every M iterations, and use the same factored matrix to construct the linear systems in the M iterations. Figure 4 shows that Newton's method converges the fastest when $M = 4$. If M is too large, this approach will be not efficient either, since the error cannot decrease sufficiently in many iterations.

5 AN INTERACTIVE SYSTEM

The goal of our system is to achieve interactive elastic shape design in two ways. In the forward way, the user interacts with the rest shape and the system updates the quasistatic shape in real time. In the inverse way, the user modifies the quasistatic shape and the system generates the rest shape accordingly.

Given the techniques described in Section 3 and 4, it is straightforward to implement basic functionalities of our system. Specifically, we use the preconditioned gradient descent method [35] to handle quasistatic simulation in forward elastic shape design. For inverse quasistatic simulation, we use Newton's method solved by a direct solver, as mentioned in Subsection 4.2. Our system can adopt any tool for interactive shape editing. In practice, we simply use another quasistatic simulator, which allows the user to modify the shapes by adding new fix constraints (in red), as shown in Figure 5.

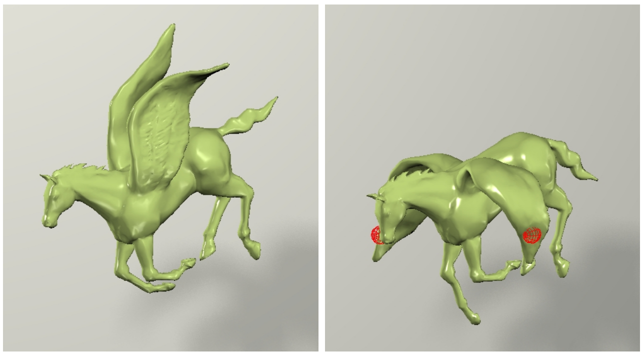

(a) The rest shape (b) The quasistatic shape

Fig. 5. The pegasus example. Our system allows the user to interactively modify the quasistatic shape by new fix constraints (in red) and examine the rest shape at the same time.

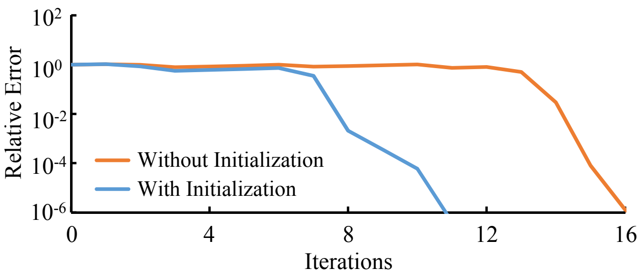


Fig. 6. The convergence of Newton's method with and without using shape initialization. Based on projective dynamics, our shape initialization technique speeds up inverse quasistatic simulation by approximately 30 percent, as shown in this plot.

A basic way of implementing our system is to run interactive shape editing and forward/inverse quasistatic simulation at the same time. However, such a system is highly susceptible to the divergence issue, since the simulation result in the current iteration can become a bad initialization in the next iteration, when the designer makes dramatic shape changes. Even though this issue can be addressed by using small step lengths, it reduces the system performance and makes divergence avoidance sophisticated. Our solution is a projective dynamics initialization method for estimating shape deformation in both forward and inverse directions. This method not only allows the designer to preview the outcome in real time, but also speeds up the convergence of simulation and inverse simulation after each user modification.

5.1 Shape Initialization by Projective Dynamics

Without loss of generality, here we study the initialization during inverse elastic shape design. It is straightforward to adjust the proposed method for forward elastic shape design, by simply swapping the quasistatic shape with the rest shape.

Let $\mathbf{x}_0$ and $\mathbf{X}_0$ be a pair of the quasistatic shape and the rest shape obtained from the last inverse simulation process. Let $\mathbf{x}$ be the newly modified quasistatic shape and $\mathbf{X}$ be the corresponding rest shape to be found. We argue that the transformation from $\mathbf{x}$ to $\mathbf{X}$ should be similar to the transformation from $\mathbf{x}_0$ to $\mathbf{X}_0$. For every tetrahedron, this similarity can be measured by the difference between two deformation gradients:

$$E_t^{\text{sim}} = \left\| \mathbf{D}_m \mathbf{D}_s^{-1} - \mathbf{D}_{m0} \mathbf{D}_{s0}^{-1} \right\|^2 = \left\| \mathbf{A}_t \mathbf{X} - \mathbf{F}_{t0}^{-1} \right\|^2, \tag{16}$$

where $\mathbf{A}_t$ is the matrix that converts $\mathbf{X}$ into the deformation gradient and $\mathbf{F}_{t0}^{-1}$ is the deformation gradient from $\mathbf{x}_0$ to $\mathbf{X}_0$. We also argue that the tetrahedron should not be too stretched or compressed from its input shape. This strain limiting condition can be quantified using another energy metric:

$$E_t^{\text{limit}} = \|\mathbf{B}_t \mathbf{X} - \mathbf{p}_t(\mathbf{B}_t \mathbf{X})\|^2, \tag{17}$$

in which $\mathbf{B}_t \mathbf{X}$ is the deformation gradient of the tetrahedron from the input shape to $\mathbf{X}$, and $\mathbf{p}_t(\mathbf{B}_t \mathbf{X})$ is its projection after isotropic strain limiting [34]. Finally, we use another energy to keep the vertices fixed in simulation remain fixed in initialization:

$$E^{\text{fix}} = \left(\mathbf{X} - \mathbf{X}^{(0)}\right)^{\mathsf{T}} \mathbf{S} \left(\mathbf{X} - \mathbf{X}^{(0)}\right), \tag{18}$$

in which $\mathbf{X}^{(0)}$ is the starting $\mathbf{X}$, and $\mathbf{S}$ is a diagonal matrix containing nonnegative fixing weights. By putting the three energies together, we form an energy minimization problem that finds an optimal $\mathbf{X}$ as: $\mathbf{X} = \arg\min \left\{ \frac{k_0}{2} \sum_t E_t^{\text{sim}} + \frac{k_1}{2} \sum_t E_t^{\text{limit}} + \frac{1}{2} E^{\text{fix}} \right\}$, which leads to a nonlinear system:

$$\begin{aligned} \mathbf{M}\mathbf{X} &= \mathbf{S}\mathbf{X}^{(0)} + \sum_t \left(k_0 \mathbf{A}_t^{\mathsf{T}} \mathbf{F}_{t0}^{-1} + k_1 \mathbf{B}_t^{\mathsf{T}} \mathbf{p}_t \right), \\ \mathbf{M} &= \mathbf{S} + \sum_t \left(k_0 \mathbf{A}_t^{\mathsf{T}} \mathbf{A}_t + k_1 \mathbf{B}_t^{\mathsf{T}} \mathbf{B}_t \right). \end{aligned} \tag{19}$$

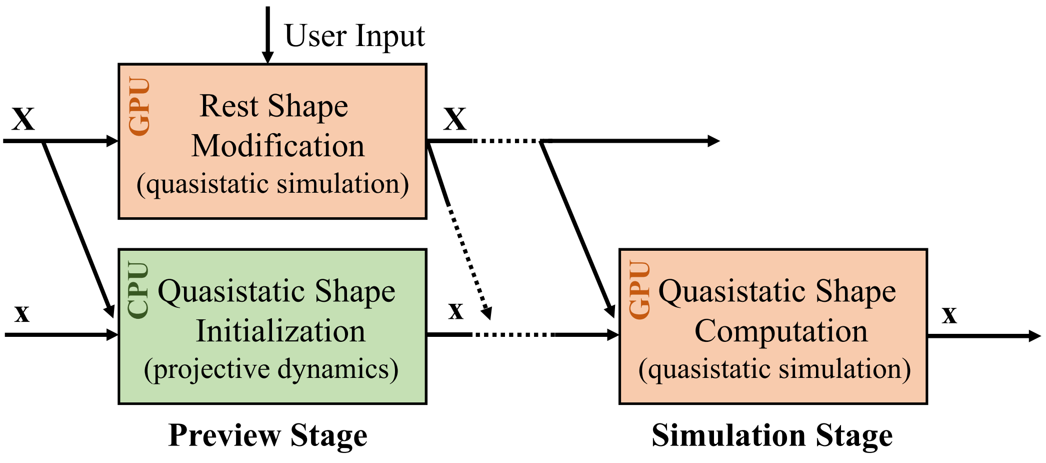
Here the nonlinearity comes from the dependence of $\mathbf{p}_t$ on $\mathbf{X}$. The projective dynamics technique [4] proposes to solve Equation 19 as a linear system multiple times, by treating $\mathbf{p}_t$ as constant each time. Projective dynamics is fast not only because $\mathbf{M}$ is independent of $\mathbf{X}$ and it can be pre-factored by Cholesky decomposition for less runtime cost, but also because the linear system can be separated into three for the three coordinates of $\mathbf{X}$. Since the initialization is called repetitively during user interaction and strain limit violation is uncommon, we found it is sufficient to solve Equation 19 just once per frame on the CPU. Note that $\mathbf{M}$ is symmetric positive definite, so the linear system must have a solution. Figure 6 reveals the effect of shape initialization on the convergence of Newton's method handling the inverse simulation process that follows.

5.2 A Heterogeneous Structure

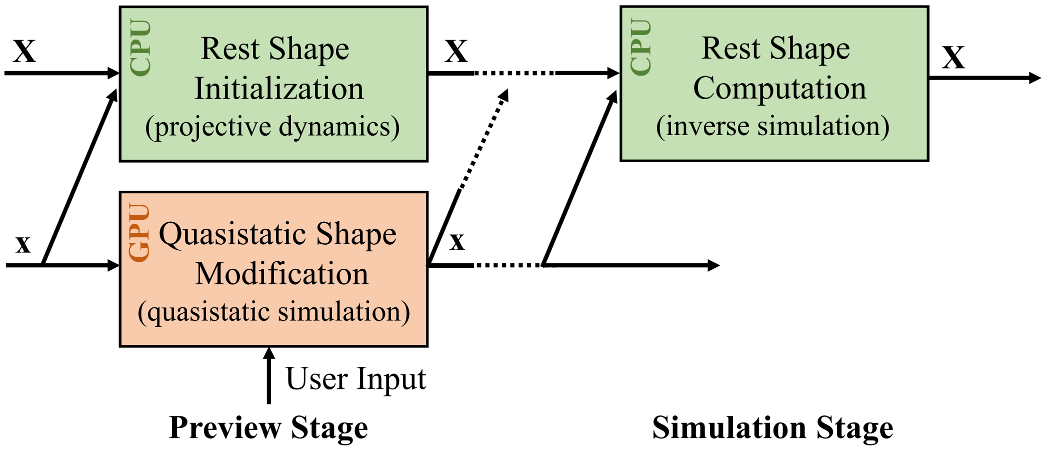
Based on the initialization method proposed in Subsection 5.1, we present our elastic shape design system as Figure 7 shows. This system divides the shape design process into two stages. In the preview stage, the system uses the initialization technique to estimate the other shape from the modified shape, for real-time preview purposes. Once the user stops editing and the modified shape converges, the system switches to the simulation stage and runs quasistatic or inverse quasistatic simulation to obtain the final optimized result.

An interesting feature of our system is that shape modification is performed on the GPU, while shape initialization is performed on the CPU[1]. This provides another level of parallelization between the processors in the preview stage, so they do not need to wait each other. For the armadillo example, shape modification takes 4.1ms per frame and shape initialization takes 12.6ms per frame in average. The theoretical upper bound on the speedup caused by CPU-GPU parallelization is 24.5 percent, while the actual speedup we observed from the experiment is approximately 18.0 percent, due to the overhead of multi-threading. We expect this speedup to be greater in the future, once the performance of a GPU receives more improvement than that of a CPU.

[1]Like inverse simulation, shape initialization runs the linear solver on the CPU. It still performs force and matrix evaluation on the GPU.



(a) Forward elastic shape design



(b) Inverse elastic shape design

Fig. 7. The system pipeline. When the user performs modification on one shape during the preview stage, the system utilizes both the CPU and the GPU to initialize the other shape in real time. After that, it runs quasistatic or inverse quasistatic simulation to reach the final result of the other shape.

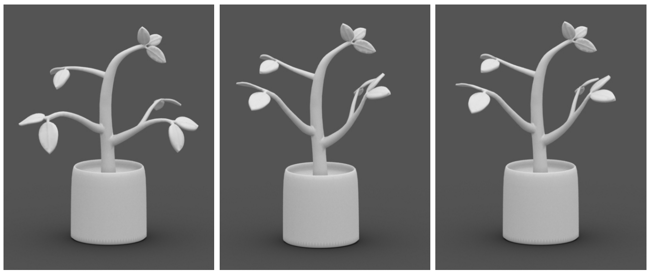
6 RESULTS AND DISCUSSIONS

(Please watch the supplemental video for animation examples.) The implementation of our system uses the Intel MKL PARDISO library and the CUDA library. Our experiment runs on an Intel Core i7-5930K 3.5GHz processor and an NVIDIA GeForce GTX TITAN X Graphics Card. Table 1 summarized the statistics and the timings of our examples. In each frame, our quasistatic simulator runs 96 gradient descent iterations for shape modification, or 128 gradient descent iterations for forward simulation. The other two simulators, i.e., the projective dynamics simulator and the inverse simulator, run only one Newton iteration in each frame. By default, we choose the neo-Hookean model for our examples.

Comparison to asymptotic numerical methods (ANM).. To verify the correctness of our inverse simulator, we compare the visual results with that of an asymptotic numerical method [5] by using

Name (#vert, #ele)	Preview Costs		Simulation Costs	
	Before	After	Forward	Inverse
Pegasus (12K, 49K)	14.9ms	12.1ms	12.0ms	15.8ms
Plant (14K, 47K)	14.8ms	12.1ms	11.8ms	15.5ms
Armadillo (14K, 55K)	16.7ms	13.7ms	12.6ms	20.4ms
Statue (16K, 60K)	19.0ms	14.7ms	14.1ms	23.0ms
Piranha (20K, 73K)	23.3ms	17.1ms	16.9ms	29.0ms

Table 1. Statistics and timings of our examples. This table summarizes the computational cost per frame in each stage. Here the two preview costs are the costs before and after implementing our CPU-GPU parallelization concept.



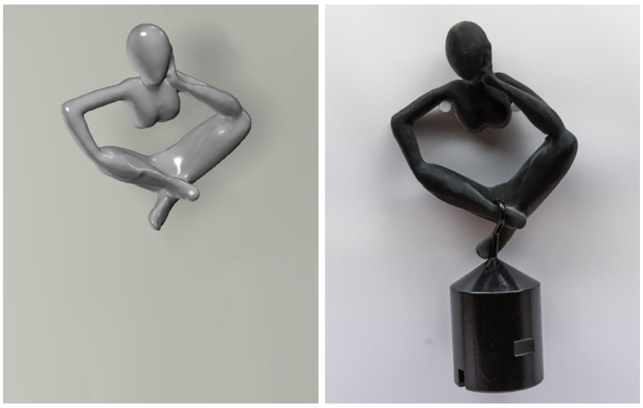
(a) The target shape (b) Our result (c) The result of ANM

Fig. 8. The plant example. The result of our inverse simulator in (b) is identical to that of the asymptotic numerical method in (c).

the same example as shown in Figure 8 on both the methods. Our experiments shows that our simulator can generate its result in 0.6 seconds, while ANM needs 3.5 seconds. Both methods use the Intel MKL PARDISO library on the CPU. We note that ANM is currently designed for the neo-Hookean model only.

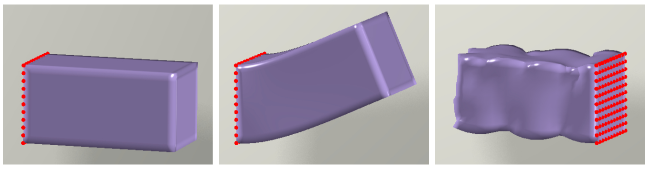
Each step of ANM can be considered as a combination of three steps: computation of an expansion coefficient using a linear solver, a residual estimation based on Brent's method and finally some (less than 3) Newton Raphson iterations. Because of the first two steps, the guesses are very close to the actual solution and so few steps are required for convergence. Our method guarantees the same assumption using the Projective Dynamics based initialization step which is performed only once at the start of the simulation stage. Additionally note that in our method, we can skip the costly hessian evaluation step occasionally and a part of the force and matrix computation is offloaded to the GPU which brings on another layer of parallelism as compared to ANM. A combination of these factors is responsible for the speed advantage compared to ANM.

Validation by 3D printing. We validate the correctness of our simulation results by 3D printing, as shown in Figure 1 and 9. In both examples, we use a Stratasys Objet 30 Prime Printer with a rubber-like tango black material. Since we do not exactly know the physical properties of *tango black*, we print the model in the target shape (in Figure 1c) and estimate material parameters manually from forward simulation. Given these parameters, we run inverse simulation on the target shape (in Figure 1a) to get the rest shape (in Figure 1b). The printed model in the rest shape sags to the desired target shape, as shown in Figure 1d. The statue example demonstrates that our inverse simulator can handle additional loads, which are invariant to the rest shape. In this example, we treat the input shape as the rest shape in Figure 9a, and run forward simulation to get



(a) The rest shape (b) The model made in the rest shape

Fig. 9. The statue example. Our inverse quasistatic simulator can successfully recover the rest shape of this model in (a) from the deformed shape under an additional 500g load shown in (b).



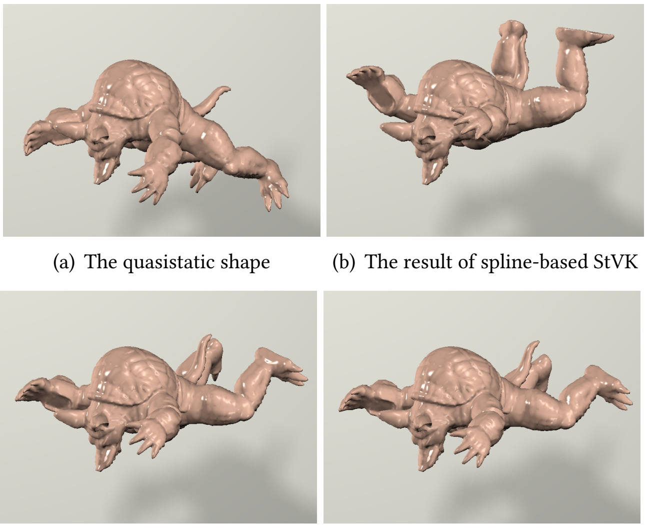
(a) The target shape (b) The rest shape (c) The inverted shape

Fig. 10. The box example. This example reveals that our inverse quasistatic simulation method cannot easily handle inverted element cases, as shown in (c).

the deformed shape in quasistatic equilibrium as Figure 9b shows. By running inverse simulation on the deformed shape, our system successfully recovers the rest shape.

Inverted elements. Since many elastic material models do not consider inverted element cases, researchers developed a variety of techniques to help quasistatic simulation handle inverted elements in the past. An interesting question is whether these techniques can help inverse quasistatic simulation handle inverted elements as well. Figure 10 shows the results of a box simulated by our inverse simulator. When the initialization is completely mirrored, the simulator fails to recover the mesh shown in Figure 10c, as expected. Unfortunately, techniques that work in quasistatic simulation, such as clamping the energy or the principal stretches [27, 35], do not work in inverse quasistatic simulation as far as we have experienced. While this issue can be easily addressed by placing a strain limit on the initial mesh, we would like to know whether a better solution exists in the future.

Nonlinear elastic models. Figure 11 demonstrates the capability of our system in handling the armadillo example under different nonlinear elastic models. In this experiment, we treat the input shape as the quasistatic shape as shown in Figure 11a. Since our spline-based stVK model [36] is the most compliant one, its rest shape differs the most from the quasistatic shape as Figure 11b shows. We note that the original StVK model has little resistance to compression, so we incorporate an additional volumetric strain energy as proposed in [11]. Figure 11c and 11d show that the results under the neo-Hookean model and the Mooney-Rivlin model are similar, because the system uses



(a) The quasistatic shape (b) The result of spline-based StVK

(c) The result of neo-Hookean (d) The result of Mooney-Rivlin

Fig. 11. The armadillo example. This example compares the rest shape results of our inverse quasistatic simulator under different nonlinear elastic models.

the same elastic modulus values. The Mooney-Rivlin result looks slightly closer to the quasistatic shape, due to an additional energy term.

Limitations. Perhaps the biggest limitation of our system is its requirement on the mesh quality. When the quality of the rest mesh is lower, forward elastic shape design needs smaller step lengths and more computational time to avoid the divergence issue. Interestingly, the mesh quality also affects inverse elastic shape design, if the modified mesh leads to a bad rest mesh. For example, when the user lifts the beam in the quasistatic shape as shown in Figure 12, the top white vertices will become more squeezed in the rest shape to count against gravity. This will cause oscillation or even divergence during the transition from the preview stage to the simulation stage, unless we eliminate the preview stage and always run Newton's method for inverse simulation. Our solution to this problem is to relax the fixing weights of the fixed vertices according to their distances to the free vertices, so that the free vertices do not get squeezed too much. Our experiment indicates that this solution effectively avoids oscillation or divergence issues.

Other limitations of the system include interruptive transition from the preview stage to the simulation stage, the need of a balanced workload between the CPU and the GPU, and lower convergence rates when it deals with stiffer or more nonlinear materials.

7 CONCLUSIONS AND FUTURE WORK

In this paper, we find that inverse quasistatic simulation can be immediately solved by Newton's method with a direct solver. Based on this observation, we develop an interactive two-way elastic shape design system, using a novel shape initialization method and a heterogeneous structure that utilizes both the CPU and the GPU.

In the future, we will explore better ways to handle inverted elements in inverse quasistatic simulation and we will improve the performance of our system using other acceleration techniques, such as multi-grid. We also would like to integrate adaptive remeshing into our system, in case the user performs significant changes to the rest shape or the quasistatic shape. Finally, we are

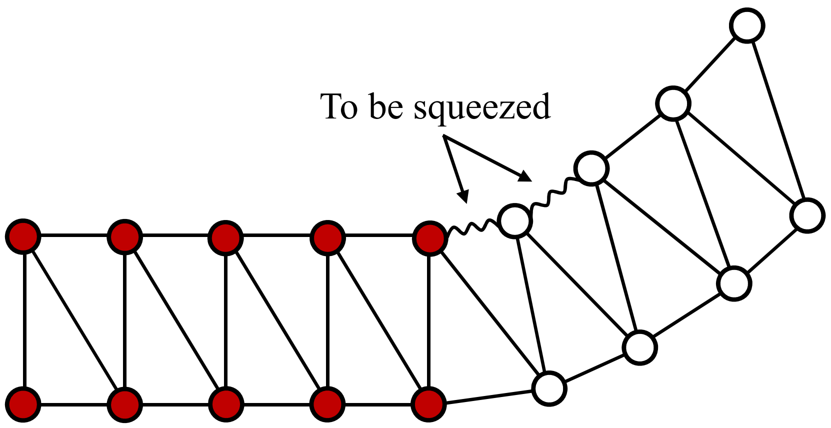


Fig. 12. A challenging example. When the red vertices are fixed and the white vertices get lifted in the quasistatic shape, the white vertices must be lifted even more in the rest shape to fight against gravity. This causes the top white vertices to be squeezed and deteriorates the rest mesh quality.

interested in developing fast simulation systems on other platforms, such as field-programmable gate array (FPGA).

ACKNOWLEDGMENTS

This work was funded by NSF grant CHS-1524992. The authors would also like to thank Adobe Research and NVIDIA Research for additional equipment and funding supports.

REFERENCES

[1] Aric Bartle, Alla Sheffer, Vladimir G. Kim, Danny M. Kaufman, Nicholas Vining, and Floraine Berthouzoz. 2016. Physics-driven Pattern Adjustment for Direct 3D Garment Editing. *ACM Trans. Graph. (SIGGRAPH)* 35, 4, Article 50 (July 2016), 11 pages.
[2] Bernd Bickel, Peter Kaufmann, Mélina Skouras, Bernhard Thomaszewski, Derek Bradley, Thabo Beeler, Phil Jackson, Steve Marschner, Wojciech Matusik, and Markus Gross. 2012. Physical Face Cloning. *ACM Trans. Graph. (SIGGRAPH)* 31, 4, Article 118 (July 2012), 118:1–118:10 pages.
[3] Marc Bonnet and A. Constantinescu. 2005. Inverse problems in elasticity. *Inverse Problems* 21, 2 (2005), R1–R50.
[4] Sofien Bouaziz, Sebastian Martin, Tiantian Liu, Ladislav Kavan, and Mark Pauly. 2014. Projective Dynamics: Fusing Constraint Projections for Fast Simulation. *ACM Trans. Graph. (SIGGRAPH)* 33, 4, Article 154 (July 2014), 11 pages.
[5] Xiang Chen, Changxi Zheng, Weiwei Xu, and Kun Zhou. 2014. An Asymptotic Numerical Method for Inverse Elastic Shape Design. *ACM Trans. Graph. (SIGGRAPH)* 33, 4, Article 95 (July 2014), 11 pages.
[6] Alicia Cordero, José L. Hueso, Eulalia Martínez, and Juan R. Torregrosa. 2010. New Modifications of Potra-PtáK's Method with Optimal Fourth and Eighth Orders of Convergence. *J. Comput. Appl. Math.* 234, 10 (Sept. 2010), 2969–2976.
[7] Stelian Coros, Sebastian Martin, Bernhard Thomaszewski, Christian Schumacher, Robert Sumner, and Markus Gross. 2012. Deformable Objects Alive! *ACM Trans. Graph. (SIGGRAPH)* 31, 4, Article 69 (July 2012), 9 pages.
[8] Alexandre Derouet-Jourdan, Florence Bertails-Descoubes, Gilles Daviet, and Joëlle Thollot. 2013. Inverse Dynamic Hair Modeling with Frictional Contact. *ACM Trans. Graph. (SIGGRAPH Asia)* 32, 6, Article 159 (Nov. 2013), 10 pages.
[9] Marco Fratarcangeli, Valentina Tibaldo, and Fabio Pellacini. 2016. Vivace: a Practical Gauss-Seidel Method for Stable Soft Body Dynamics. *ACM Trans. Graph. (SIGGRAPH Asia)* 35, 6 (Nov. 2016), 214:1–214:9.
[10] Sunil Hadap. 2006. Oriented Strands: Dynamics of Stiff Multi-body System. In *Proceedings of SCA*. 91–100.
[11] Ryo Kikuuwe, Hiroaki Tabuchi, and Motoji Yamamoto. 2009. An Edge-based Computationally Efficient Formulation of Saint Venant-Kirchhoff Tetrahedral Finite Elements. *ACM Trans. Graph.* 28, 1, Article 8 (Feb. 2009), 13 pages.
[12] Tae-Yong Kim, Nuttapong Chentanez, and Matthias Müller-Fischer. 2012. Long Range Attachments - A Method to Simulate Inextensible Clothing in Computer Games. In *Proceedings of SCA*. 305–310.
[13] Tiantian Liu, Adam W. Bargteil, James F. O'Brien, and Ladislav Kavan. 2013. Fast Simulation of Mass-Spring Systems. *ACM Transactions on Graphics* 32, 6 (Nov. 2013), 209:1–7. http://cg.cis.upenn.edu/publications/Liu-FMS Proceedings

of ACM SIGGRAPH Asia 2013, Hong Kong.

[14] Tiantian Liu, Sofien Bouaziz, and Ladislav Kavan. 2016. Towards Real-time Simulation of Hyperelastic Materials. *arXiv preprint arXiv:1604.07378* (2016).

[15] Matthias Müller. 2008. Hierarchical Position Based Dynamics. In *Proceedings of VRIPHYS*. 1–10.

[16] Matthias Müller, N. Chentanez, T.Y. Kim, and M. Macklin. 2014. Strain Based Dynamics. In *Proceedings of SCA*. 21–23.

[17] Matthias Müller, Bruno Heidelberger, Matthias Teschner, and Markus Gross. 2005. Meshless Deformations Based on Shape Matching. *ACM Trans. Graph. (SIGGRAPH)* 24, 3 (July 2005), 471–478.

[18] James F. O'Brien, Adam W. Bargteil, and Jessica K. Hodgins. 2002. Graphical modeling and animation of ductile fracture. *ACM Trans. Graph. (SIGGRAPH)* 21, 3 (July 2002), 291–294.

[19] Théodore Papadopoulo and Manolis I. A. Lourakis. 2000. Estimating the Jacobian of the Singular Value Decomposition: Theory and Applications. In *Proceedings of the 6th European Conference on Computer Vision-Part I*. 554–570.

[20] Jesús Pérez, Bernhard Thomaszewski, Stelian Coros, Bernd Bickel, José A. Canabal, Robert Sumner, and Miguel A. Otaduy. 2015. Design and Fabrication of Flexible Rod Meshes. *ACM Trans. Graph. (SIGGRAPH)* 34, 4, Article 138 (July 2015), 12 pages.

[21] Xavier Provot. 1996. Deformation Constraints in a Mass-Spring Model to Describe Rigid Cloth Behavior. In *Proceedings of Graphics Interface*. 147–154.

[22] Alec R. Rivers and Doug L. James. 2007. FastLSM: Fast Lattice Shape Matching for Robust Real-time Deformation. *ACM Trans. Graph. (SIGGRAPH)* 26, 3, Article 82 (July 2007).

[23] Hijung V. Shin, Christopher F. Porst, Etienne Vouga, John Ochsendorf, and Frédo Durand. 2016. Reconciling Elastic and Equilibrium Methods for Static Analysis. *ACM Trans. Graph.* 35, 2, Article 13 (Feb. 2016), 13:1–13:16 pages.

[24] Mélina Skouras, Bernhard Thomaszewski, Bernd Bickel, and Markus Gross. 2012. Computational Design of Rubber Balloons. *Comput. Graph. Forum* 31, 2pt4 (May 2012).

[25] Mélina Skouras, Bernhard Thomaszewski, Stelian Coros, Bernd Bickel, and Markus Gross. 2013. Computational Design of Actuated Deformable Characters. *ACM Trans. Graph. (SIGGRAPH)* 32, 4, Article 82 (July 2013), 10 pages.

[26] J. Teran, S. Blemker, V. Ng Thow Hing, and R. Fedkiw. 2003. Finite Volume Methods for the Simulation of Skeletal Muscle. In *Proceedings of SCA*. 68–74.

[27] Joseph Teran, Eftychios Sifakis, Geoffrey Irving, and Ronald Fedkiw. 2005. Robust Quasistatic Finite Elements and Flesh Simulation. In *Proceedings of SCA*. 181–190.

[28] Demetri Terzopoulos, John Platt, Alan Barr, and Kurt Fleischer. 1987. Elastically Deformable Models. *SIGGRAPH Comput. Graph.* 21, 4 (Aug. 1987), 205–214.

[29] Bernhard Thomaszewski, Simon Pabst, and Wolfgang Strasser. 2009. Continuum-based Strain Limiting. *Computer Graphics Forum (Eurographics)* 28, 2 (2009), 569–576.

[30] Maxime Tournier, Matthieu Nesme, Benjamin Gilles, and François Faure. 2015. Stable Constrained Dynamics. *ACM Trans. Graph. (SIGGRAPH)* 34, 4, Article 132 (July 2015), 10 pages.

[31] Christopher D. Twigg and Zoran Kačić-Alesić. 2011. Optimization for Sag-free Simulations. In *Proceedings of SCA*. 225–236.

[32] Bin Wang, Longhua Wu, KangKang Yin, Uri Ascher, Libin Liu, and Hui Huang. 2015. Deformation Capture and Modeling of Soft Objects. *ACM Trans. Graph. (SIGGRAPH)* 34, 4, Article 94 (July 2015), 12 pages.

[33] Huamin Wang. 2015. A Chebyshev Semi-iterative Approach for Accelerating Projective and Position-based Dynamics. *ACM Trans. Graph. (SIGGRAPH Asia)* 34, 6, Article 246 (Oct. 2015), 9 pages.

[34] Huamin Wang, James O'Brien, and Ravi Ramamoorthi. 2010. Multi-resolution isotropic strain limiting. *ACM Trans. Graph. (SIGGRAPH Asia)* 29, 6, Article 156 (Dec. 2010), 156:1–156:10 pages.

[35] Huamin Wang and Yin Yang. 2016. Descent Methods for Elastic Body Simulation on the GPU. *ACM Trans. Graph. (SIGGRAPH Asia)* 35, 6, Article 212 (Nov. 2016), 10 pages.

[36] Hongyi Xu, Funshing Sin, Yufeng Zhu, and Jernej Barbič. 2015. Nonlinear Material Design Using Principal Stretches. *ACM Trans. Graph. (SIGGRAPH)* 34, 4, Article 75 (July 2015), 11 pages.