EUROGRAPHICS 2016 / J. Jorge and M. Lin
(Guest Editors)

*Volume 35 (2016), Number 2*

# Fast and Robust Inversion-Free Shape Manipulation

Tiantian Liu[1], Ming Gao[2], Lifeng Zhu[1,4], Eftychios Sifakis[2] and Ladislav Kavan[1,3]

[1]University of Pennsylvania, [2]University of Wisconsin-Madison, [3]University of Utah, [4]Southeast University

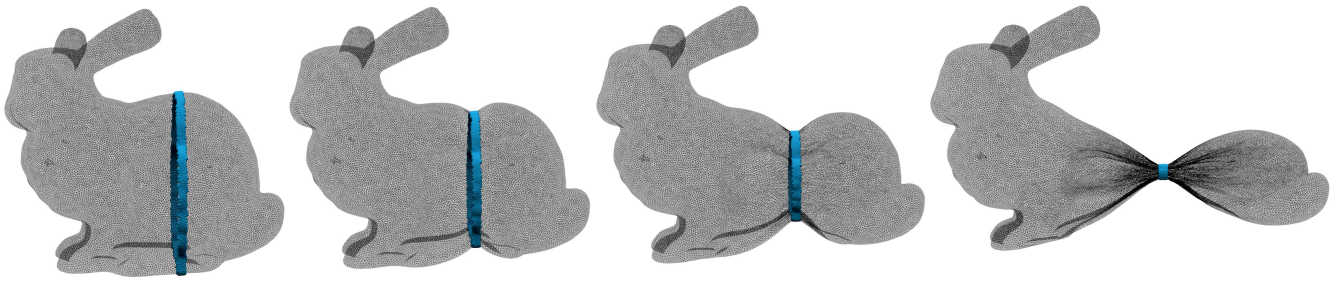


**Figure 1:** *A band of surface vertices on a tetrahedralized bunny is constricted to a tight noose. Even with severe distortion, the resulting mesh is inversion-free, i.e., all tetrahedral elements have positive volume.*

**Abstract**

*We present a shape manipulation technique capable of producing deformations of 2D and 3D meshes, guaranteeing that no elements will be inverted. We achieve this by augmenting the quadratic ex-rotated elastic energy with additional convex terms that penalize the presence of inverted elements. Using a schedule of increasing penalty coefficients, we efficiently and robustly converge to an inversion free state by solving a sequence of unconstrained convex minimization problems. This process can be interpreted as a special purpose Semi-Definite Programming (SDP) solver. We demonstrate that our method outperforms solvers used in previous work, including commercial-grade SDP software (MOSEK). As an additional benefit, our method also converges to the solution via a more intuitive path, which can be used for quick preview. We demonstrate the efficacy of our scheme in a number of 2D and 3D shapes undergoing moderate to drastic deformation.*

## 1. Introduction

Deformable solids may compress or stretch, but by no means vanish. This fact is obvious, but very difficult to guarantee in numerical simulations, where we discretize the simulated objects using *elements*, i.e., triangles in 2D or tetrahedra in 3D. Intuitively, inversion means that an element flips "inside-out," like an umbrella flipped in strong wind. Mathematically, the *non-inversion* constraint can be expressed by requiring the deformation gradient of each element to be positive, which yields a rather difficult non-convex set. This fact greatly complicates the design of numerical algorithms to avoid inverted elements.

This paper builds on related recent results [SKPSH13, KABL14] in order to develop a fast and versatile shape deformation method which avoids inverted elements, provides robustness guarantees, and does not rely on third-party optimization packages. We assume the user needs to manipulate a 2D or 3D volumetric shape using *handles* – subsets of vertices of our discretization which are directly coupled to a graphical user interface elements (manipulators). This is a common concept in skinning and shape deformation. Unfortunately, classical methods such as skinning with bounded biharmonic weights [JBPS11] quickly lead to inverted elements, even for moderately complex deformations.

To address this problem, we start by designing a convex quadratic deformation energy which produces results similar to skinning, but allows us to enhance the algorithm with non-inversion constraints. The key idea is to use bounded biharmonic weights to interpolate the *rotations* of each handle. Using this procedurally computed rotation field, we formulate a purely quadratic elastic energy which targets the externally provided rotations, and can be minimized to generate the deformed shape. Due to these explicitly specified rotations, we call this energy "ex-rotated elasticity." This concept is not new: the ex-rotated ideas appeared in [CGC*02, WPP07, WSLG07] as well as in more recent work [GMS14]. In this paper, we combine ex-rotated elasticity with an efficient method to prevent inversions.

The idea of ex-rotation fields is also utilized in the approach of [KABL14], who conservatively approximate the full non-convex non-inversion constraints with a more restrictive set of *semi-definite* constraints. Whereas Kovalsky et al. [KABL14] use this convex ap-

proximation step in an iterative process that ultimately converges to a solution of the original non-convex non-inversion constraints, we observe that for the types of deformations targeted by our work, a *single* ex-rotation field produces adequate results even though it imposes more conservative constraints than the non-convex non-inversion constraint would. This observation is a consequence of the fact that, for the types of deformation problems we target, we can procedurally furnish an ex-rotation field that provides a reasonable estimate of the final rotation field, similarly to previous skinning methods [CGC*02, WPP07, WSLG07]. Thus, we steer our attention to the solution process for this single semi-definite optimization problem, with the goal of producing a solver that is (i) simple to implement, without requiring delicate modeling steps or third-party software, (ii) provides intermediate iterates which can be utilized as early previews of the final result, and (iii) matches or even outperforms established *general-purpose* semi-definite programming packages. An example of a commercial solver package we use as a measure of reference is MOSEK, which in many of our examples would not yield competitive performance and would not be able to provide meaningful preview of the result.

Instead, we propose a specialized semi-definite solver, specifically designed for interactive inversion-free shape deformation. In contrast to interior point methods which are commonly used to solve semi-definite programs [BV04], we do not require an inversion-free starting point. Instead, our method works by progressively increasing the penalty term for inversion; upon convergence, all our elements are inversion-free. An interesting and somewhat counter-intuitive fact is that we can increase the weight of the penalty terms to extremely high values without any numerical difficulties. The convergence path of our method is quite natural, with early iterations providing a good approximation of the final result as inverted elements are progressively corrected. Thus, we can provide a meaningful preview to the user early on, in contrast to interior-point solvers whose intermediate iterates are typically not indicative of the final result.

In a number of practical shape deformation scenarios, we demonstrate that our method outperforms solvers used in previous work, including commercial semi-definite programming software (MOSEK). As an additional benefit, our method relies only on standard linear algebra libraries, we do not need any third-party optimization software. Our algorithm is much simpler to implement than modern interior point methods and we believe it will be a welcome addition among existing tools of geometry processing and physics-based simulation.

## 2. Related Work

In physics-based animation, it is common to design material models which penalize inversion [ITF04, CPSS10, SHST12, SWM*14, CFS14]. However, under larger external forces, these penalties may not be strong enough to prevent inversions and the resulting undesired self-overlaps. Specialized material models such as Neo-hookean elasticity [BW97] contain terms that grow to infinity as the area/volume of a deformed element approaches to zero. Unfortunately, these terms greatly complicate the numerics and may grind the classical Newton's method to a halt [SKPSH13]. Chen et al. [CZXZ14] discusses how to overcome these numerical difficulties in the context of inverse elastic shape design.

Physics-based simulation is known for its relatively high computational costs. *Skinning* techniques have been developed to produce plausible deformations quickly, fast enough even for real-time applications [JDKL14]. Our method builds on the concept of skinning with explicitly provided rotations for each element ("ex-rotations") [CGC*02, WPP07, WSLG07]. More recently, Gao et al. [GMS14] used ex-rotated skinning to speed up collision response. However, they detect and resolve collisions only at the boundary; internal elements are free to invert and self-overlap. In this paper, we focus on the orthogonal problem of element inversions. Indeed, if boundaries are deformed without self-intersections *and* all elements are non-inverted, the entire volumetric deformation is a bijective map; this result as well as alternative necessary and sufficient conditions were studied by Lipman [Lip14].

The problem of computing inversion-free mappings received considerable attention in recent years. Lipman [Lip12] reduced the 2D version of the problem to second order conic programming (SOCP), while Weber et al. [WMZ12] pointed out the connections to the theory of Teichmüller maps. In addition to non-inversion, these algorithms also strive to reduce conformal distortion but, unfortunately, both of these methods are limited to 2D. One year later, Aigerman and Lipman [AL13] generalized the previous approach to 3D, but only in the context of projections on an inversion-free configuration – arbitrary deformation energies were not supported. This limitation was lifted in [KABL14], proposing a general framework for controlling singular values. This framework embeds many different instances of inversion-free problems but requires us to solve a sequence of semi-definite programs (SDP). Even though robust and highly optimized SDP solvers exist (e.g., the commercial MOSEK package), their computational requirements are rather high even for modest meshes, hampering interactivity.

The long wait-times of existing solvers motivated Schüller et al. [SKPSH13] to develop Locally Injective Mappings (LIM), a custom barrier method which provides real-time feedback to the user. LIM was subsequently improved by online remeshing [JHT14]. Assuming the input configuration is inversion-free, LIM guarantees that no elements will ever invert. Unfortunately, there are no convergence guarantees and indeed, we observed that in certain cases LIM makes only very small progress towards the solution and requires a lot of iterations. Even upon convergence, global optimality is not guaranteed. In contrast, we design our problem to be convex, which allows us to reap the benefits of convex optimization: guaranteed convergence to a global optimum. Poranne and Lipman [PL14] investigated an interactive approach of inversion-free deformations with provable guarantees. While producing fast and smooth results, their method is limited to 2D deformations with limited number of degrees of freedom. Recently, Kovalsky et al. [KABL15] proposed another efficient algorithm to compute bounded distortion mappings based on projections onto approximate tangent planes. However, this method converges poorly in challenging deformation cases, and may produce oscillatory behaviors.

Element inversions are problematic also in mesh parameterization. Even though related, parameterization is not the main focus of this

paper and we refer to surveys [FH05, SPR06] for more complete literature review. Finding optimal parameterizations can be formulated as an optimization problem [HG00, DMK03] but finding accurate numerical solutions are difficult due to sharp non-linearities of the objective, similar to the Neo-hookean material in physics-based simulation. Schneider et al. [SHF13] constructed a bijective map between a source and target domain by creating a sequence of intermediate polygons. More recently, [WZ14] and [SH15] leveraged the Radó-Kneser-Choquet theorem which guarantees bijectivity of a harmonic map as long as the boundary of the target domain is convex. Even though preliminary 3D results were presented, these approaches are currently limited to 2D. In three and more dimensions, it is possible to guarantee bijectivity with integral curve coordinates [HG15]. However, all of the above described methods require the deformation of the entire boundary to be specified. This precludes these methods from applications in skinning and shape deformation, where we need to control the shape using internal handles, such as bones [JBPS11].

Similar problems arise also in quad meshing. Bommes et al. [BZK09] addresses element inversions using local stiffening, which is effective in many cases, but some inverted elements may remain. A more recent strategy is based on the idea of convexification [BCE*13], similar in spirit to [Lip12], but generalizable also to 3D.

A different approach to produce bijective maps is based on the fact that time integration of a smooth velocity field results in path-lines that do not intersect in space-time [vFTS06, AS07, ERF*11]. However, this fact is true only in the continuous settings. In practice, the performance and accuracy of this approach is limited by spatio-temporal discretization. Another possibility is to employ advanced deformation energies, e.g., involving the $l_\infty$ norm instead of the usual $l_2$ metric [LZ14]. This allows us to control the worst-case distortion but, unfortunately, element inversions can still occur.

## 3. Method

We assume that our domain is discretized using a triangle ($d = 2$) or tetrahedral mesh ($d = 3$). We did not investigate $d > 3$. We denote the current deformed configuration as $\mathbf{x} \in \mathbb{R}^{nd}$, where $n$ is the number of vertices. We can define the deformation gradient of element number $i$ as $\mathbf{F}_i \in \mathbb{R}^{d\times d}$, assuming a fixed rest pose configuration. We note that the deformation gradients are linear functions of $\mathbf{x}$. To reduce notation clutter we will write only $\mathbf{F}_i$ instead of denoting this dependence explicitly (such as $\mathbf{F}_i(\mathbf{x})$). We refer to [SB12] for details on Finite Element discretizations of deformable solids.

We use *ex-rotated elasticity* as our material model. The term ex-rotated elasticity was adopted by [GMS14] as a shorthand for "explicit rotations" of finite elements; similar ideas appeared in earlier work [CGC*02, WPP07, WSLG07]. Compared to the more commonly used co-rotated elasticity model [CPSS10] which defines element rotations implicitly, ex-rotated elasticity has a significant practical advantage: the energy function is convex. This property is key to our robust and efficient numerical solution procedures. Ex-rotated elasticity assumes a user provided rotation $\hat{\mathbf{R}}_i \in SO(d)$ for each element $i$. Because our goal is an interactive shape deformation system, we assume the user provides position and orientation of subsets of vertices, called *handles*. We use bounded biharmonic weights [JBPS11] to compute non-negative influence weights associated with each handle. The ex-rotation $\hat{\mathbf{R}}_i$ for each element $i$ is then calculated by quaternion blending of handle rotations [KŽ05], using the bounded biharmonic weights as blending coefficients. The ex-rotated elastic potential is then defined as:

$$E_{\text{ex}}(\mathbf{x}) = \sum_i ||\mathbf{F}_i - \hat{\mathbf{R}}_i||_F^2 \tag{1}$$

The constraints on positions and orientations of the handle vertices are affine, and therefore can be expressed as $\mathbf{Cx} = \mathbf{d}$. Even though these constraints can be enforced as hard constraints, we prefer to use only "soft" penalties which allow us to compromise the constraints if necessary:

$$E_{\text{tar}}(\mathbf{x}) = \alpha||\mathbf{Cx} - \mathbf{d}||^2 \tag{2}$$

where $\alpha > 0$ is a constant specifying the weight of this term. In all our experiments we use $\alpha = 10^4$. Because both $E_{\text{ex}}$ and $E_{\text{tar}}$ are convex quadratic functions, it is very easy and efficient to compute $\mathbf{x}$ which minimizes $E_{\text{ex}}(\mathbf{x}) + E_{\text{tar}}(\mathbf{x})$. This leads to an interactive skinning technique, producing results of quality not only comparable to skinning with bounded biharmonic weights [JBPS11], but in many cases even competitive with full co-rotated simulation, as shown by Gao et al. [GMS14]. Unfortunately, when the targeting constraints require larger deformations, this technique quickly starts to produce inverted elements, i.e., triangles/tets with negative area/volume, which lead to undesired self-intersections.

Assuming that all elements have positive area/volume in the rest pose, the condition to avoid inversions can be expressed as $\det(\mathbf{F}_i) \geq 0$. While some works attempted to satisfy this constraint directly [SKPSH13], the stumbling block is the fact that the set of matrices $\mathbf{F}_i \in \mathbb{R}^{d\times d}$ satisfying $\det(\mathbf{F}_i) \geq 0$ is not convex. In order to benefit from the robustness and efficiency of convex optimization methods, we follow [KABL14] and propose to further restrict our deformations to a convex subset of $\det(\mathbf{F}_i) \geq 0$. Because there are many convex subsets, we inform our choice by the ex-rotations $\hat{\mathbf{R}}_i$ which suggest the ideal desired rotation of element number $i$. Formally, we write:

$$\det(\mathbf{F}_i) = \det(\hat{\mathbf{R}}_i^\mathsf{T}\mathbf{F}_i) = \det(\mathbf{S}_i + \mathbf{A}_i) \geq \det(\mathbf{S}_i) \tag{3}$$

where $\mathbf{S}_i := \frac{1}{2}(\hat{\mathbf{R}}_i^\mathsf{T}\mathbf{F}_i + \mathbf{F}_i^\mathsf{T}\hat{\mathbf{R}}_i)$ is the symmetric part of $\hat{\mathbf{R}}_i^\mathsf{T}\mathbf{F}_i$ and $\mathbf{A}_i := \frac{1}{2}(\hat{\mathbf{R}}_i^\mathsf{T}\mathbf{F}_i - \mathbf{F}_i^\mathsf{T}\hat{\mathbf{R}}_i)$ is the anti-symmetric part. In other words, we decompose $\hat{\mathbf{R}}_i^\mathsf{T}\mathbf{F}_i$ into its symmetric and anti-symmetric parts and note that the anti-symmetric component can only increase the determinant, please see our supplemental document for a proof. Therefore, if we ensure $\det(\mathbf{S}_i) \geq 0$, we will automatically obtain non-inversion. This inequality is satisfied for all positive semi-definite matrices $\mathbf{S}_i$. Most importantly, the set of symmetric positive semi-definite matrices is convex and it can be shown that it is the maximal convex subset of $\det(\mathbf{F}_i) \geq 0$ containing our target ex-rotation $\hat{\mathbf{R}}_i$ [KABL14]. We note that, if the rotational component of $\mathbf{F}_i$, computed from the polar decomposition, matches $\hat{\mathbf{R}}_i$, Eq. 3 becomes an equality and our semi-definite constraint reduces to non-inversion. Since Eq. 1 encourages agreement between explicit and implicit rotations, we expect our semi-definite constraints to be not much tighter than $\det(\mathbf{F}_i) \geq 0$ in most cases. In contrast, the method of Kovalsky et al. [KABL14] iterates this convexification

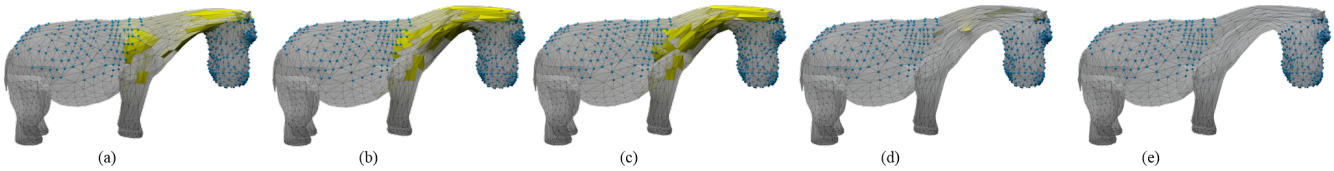



**Figure 2:** *Increasing the penalty parameter $\tau$ will correct the shape towards an uninverted state in an intuitive path, we show $\mathbf{x}^\star(\tau) = \arg\min_{\mathbf{x}} E(\mathbf{x},\tau)$ with different $\tau$ values using this hippo example: (a) $\tau = 1$, (b) $\tau = 10^3$, (c) $\tau = 10^4$, (d) $\tau = 10^6$, (e) $\tau = 10^{15}$, where (e) is the globally converged solution.*

process using the rotation field extracted from the previous iterate, ultimately converging to a solution of a non-convex optimization problem. Instead, we perform only a single convex (semi-definite) approximation using the procedurally computed ex-rotation field. Although the iterative update of the ex-rotation fields is quite justified in the problems targeted by Kovalsky et al. [KABL14], we observed that our (single) convex approximation yields quite satisfactory results for shape manipulation. Therefore, in the following we focus on improving the qualities of the numerical solver used for this semi-definite optimization problem.

The problem of minimizing $E_{\text{ex}} + E_{\text{tar}}$ subject to $\mathbf{S}_i$ being positive semi-definite is a semi-definite program (SDP), a classical convex optimization problem which can be solved using existing convex optimization software packages. Even though commercial-grade packages such as MOSEK [AA99] are robust and highly optimized, solving SDPs is time-consuming and precludes interactive user experience even with small meshes [KABL14]. In this paper, we therefore propose a different approach, which does not rely on third-party convex optimization software and is much faster in many practical situations. The basic idea is related to classical penalty methods [NW06]. Specifically, we introduce a term to penalize negative eigenvalues of $\mathbf{S}_i$:

$$E_{\text{pen}}(\mathbf{x}) = \sum_{i,j} p(\lambda_j(\mathbf{S}_i)) \tag{4}$$

where $\lambda_j$ for $j = 1,\ldots,d$ is a function which returns the $j$-th eigenvalue of the input matrix. Note that because the $\mathbf{S}_i$ matrices are always real and symmetric, all of their eigenvalues are real and therefore can be ordered, as is commonly done in numerical linear algebra subroutines. Similarly to deformation gradients, the matrices $\mathbf{S}_i$ are also linear functions of $\mathbf{x}$, even though, again, we do not denote this linear dependence explicitly.

The function $p : \mathbb{R} \to \mathbb{R}_+$ is a penalty function that will help us enforce $\lambda_j(\mathbf{S}_i) \geq 0$. A common strategy used in previous methods to achieve non-inversion [SKPSH13] as well as interior-point methods for convex optimization [BV04] is to employ a barrier function, such as $-\log(x)$, which is defined only for positive real numbers. An important distinction of our method is that our penalty function $p$ is finite and well-defined on the entire $\mathbb{R}$. One specific penalty function that works well is clamped quadratic:

$$p(x) = \begin{cases} (x-\varepsilon)^2 & \text{if } x < \varepsilon \\ 0 & \text{if } x \geq \varepsilon \end{cases} \tag{5}$$

where $\varepsilon > 0$ is a small constant which we use to cope with limited precision of floating point arithmetics. In all our experiments we use $\varepsilon = 10^{-6}$.

Using this penalty function, we define our final objective as:

$$E(\mathbf{x},\tau) = E_{\text{ex}}(\mathbf{x}) + \tau E_{\text{pen}}(\mathbf{x}) + E_{\text{tar}}(\mathbf{x}) \tag{6}$$

Note that as a function of $\mathbf{x}$, $E$ is finite and *convex* on the entire domain $\mathbb{R}^{nd}$. Convexity is easy to see for the terms $E_{\text{ex}}$ and $E_{\text{tar}}$, which are quadratic. The convexity of the $E_{\text{pen}}$ term follows from the convexity of $p$ and the *symmetry* of $\mathbf{S}_i$ [BL10,PB13]. This symmetry is the key difference between the ex-rotated and co-rotated elastic energies.

If we denote:

$$\mathbf{x}^\star(\tau) = \arg\min_{\mathbf{x}} E(\mathbf{x},\tau) \tag{7}$$

the desired solution can be written as:

$$\mathbf{x}^\star = \lim_{\tau\to\infty} \mathbf{x}^\star(\tau) \tag{8}$$

For any fixed $\tau$, the minimizer $\mathbf{x}^\star(\tau)$ can be found using Newton's method. At first sight, attempting to calculate $\mathbf{x}^\star$ according to (8) seems to be a folly, because large values of $\tau$ will inevitably lead to poorly conditioned Hessians $\nabla^2_{\mathbf{x}} E$. Indeed, if we execute standard Newton's method with backtracking line-search [BV04] on a problem with moderately high $\tau$, such as $10^6$, we often observe a very poor convergence behavior. Even though the objective is convex, the steep penalties force Newton's method to take only very small steps, resulting in an excessive number of iterations. Similar problems were observed by [SKPSH13].

An important observation is that $\mathbf{x}^\star$ can be computed efficiently and robustly by solving a sequence of relaxed problems with increasing complexity, using the solution of previous (simpler) problem to bootstrap the solve of the subsequent (harder) problem. This is a very broadly applicable principle, common to many efficient numerical algorithms, including multigrid and interior point methods. In our case, the parameter $\tau$ lends itself as a natural relaxation parameter. We start with $\tau = 0$, which corresponds to ignoring the semi-definiteness constraints. In this case, we are minimizing convex quadratic function which amounts to a single linear system solve. Of course, some elements of the resulting state $\mathbf{x}^\star(0)$ can be inverted. Therefore, we increase the $\tau$ to a small number, e.g., 1, and apply Newton's method. With this relatively small $\tau$, Newton's method converges quickly to the solution $\mathbf{x}^\star(1)$, typically in 20 or even much fewer iterations, regardless of the number of vertices. This is the expected (text-book) behavior of Newton's method [BV04]. A nice property of Newton's method is that it computes highly accurate solutions, limited only by machine precision. Subsequently, we increase the $\tau$ e.g. to 10 and calculate $\mathbf{x}^\star(10)$ using the previously computed $\mathbf{x}^\star(1)$ as an initial guess. We repeat the

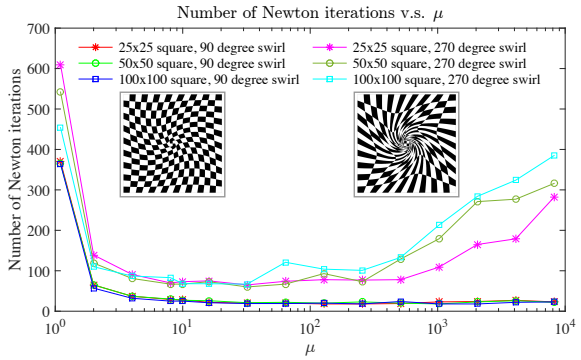



**Figure 3:** *Total number of iterations (y-axis) vs. the multiplier $\mu$ (x-axis). The total number of iterations is approximately constant over a large range of multipliers $\mu$.*

process for $\tau = 100, 1000, ..., 10^{15}$, stopping at $10^{15}$ only because at this point we start to approach the limits of double-precision floating point accuracy. Fig. 2 shows the converging sequence with increasing $\tau$. Surprisingly, despite the very steep penalties for higher $\tau$, the number of Newton iterations remains very small, often even much less than 20, even for high resolutions. The reason for this somewhat counter-intuitive behavior are the high quality initial starting points, computed with previous values of $\tau$.

There is of course nothing special about the multiplicative constant 10. Our experiments indicate that any multiplier $\mu$ within the range of, roughly, $5 - 500$ works well. For small multipliers, the initial guess for the subsequent problem will be very good because the problems are almost the same, however, we have to increase the multiplier many times, resulting in many total iterations. For too high multipliers (e.g., more than 1000), the optimization problem changes too dramatically and our initial guesses are no longer reliable, resulting in slow convergence of Newton's method. Interestingly, this behavior does not seem to vary much with resolution or complexity of the deformations, see Fig. 3. In all our experiments, we fix $\mu = 10$, which gives us a smoother convergence path than higher values of $\mu$.

Similar ideas appear also in classical interior point methods for semi-definite programming (SDP). However, compared to SDP solvers, our penalty approach behaves quite differently. One obvious difference is that our method is not an interior point method, because inequality constraints can be violated at intermediate iterations. A practical benefit is that there is no need to provide a feasible starting point – all points $\mathbf{x} \in \mathbb{R}^{nd}$ are feasible. Another advantage is that well-behaved elements (i.e., with eigenvalues of the $\mathbf{S}_i$ matrix greater then $\varepsilon$) are not penalized, because the function $p$ is constant zero. Therefore, unlike barrier methods which influence all elements, our method focuses only on elements which violate the semi-definite constraints. This means that a reasonable estimate of the solution is available well before convergence, which is very useful for interactive preview. The convergence path of interior point methods is much less intuitive as the shape of *all* elements changes, even well-behaved ones (see Fig. 8).

Another advantage of our method is speed. Even if we require full accuracy and dutifully increase our parameter $\tau$ up to its maximal level $10^{15}$, our algorithm often outperforms a commercial SDP solver (we used MOSEK for our experiments, see Sec. 4). This is remarkable, because our algorithm is much easier to implement than modern interior point methods and our prototype implementation is much less carefully optimized than commercial software.

**Algorithm 1:** Inversion-free Shape Deformation

**Given** a starting point $\mathbf{x} := \mathbf{x}_0$, $\tau := \tau_0$, $\mu > 1$, tolerance $\varepsilon > 0$, upper bound $\zeta > 0$
(default values: $\tau_0 = 1, \mu = 10, \varepsilon = 10^{-7}, \zeta = 10^{15}$)
**repeat**
  Compute Hessian $\mathbf{H} := \nabla^2_{\mathbf{x}} E(\mathbf{x}, \tau)$
  Compute gradient $\mathbf{g} := \nabla_{\mathbf{x}} E(\mathbf{x}, \tau)$
  Compute descent direction: $\Delta\mathbf{x} := -\mathbf{H}^{-1}\mathbf{g}$
  Linesearch: find step size $t$
  Update: $\mathbf{x} := \mathbf{x} + t\Delta\mathbf{x}$
  **if** $\mathbf{g}^T\mathbf{H}^{-1}\mathbf{g} < \varepsilon^2$ **then**
    increase $\tau$: $\tau := \mu\tau$
  **end**
**until** $\tau > \zeta$

## 4. Results

We tested our algorithm on various 2D and 3D shapes. In each example, we compare the performance of our method to MOSEK SDP (the same as used in [KABL14]), MOSEK SOCP solver used in [Lip12] (applicable only to 2D problems) and solving a dual problem formulated using CVX. The results can be found in Table 1. In addition, in Fig. 4 we show how the performance of our method scales when increasing the resolution of the mesh. In our current implementation, we use the soft formulation of targeting constraints, according to Equation (2). We also experimented with hard constraints incorporated using Lagrange multipliers (Newton's method with equality constraints [BV04]). Unfortunately, this

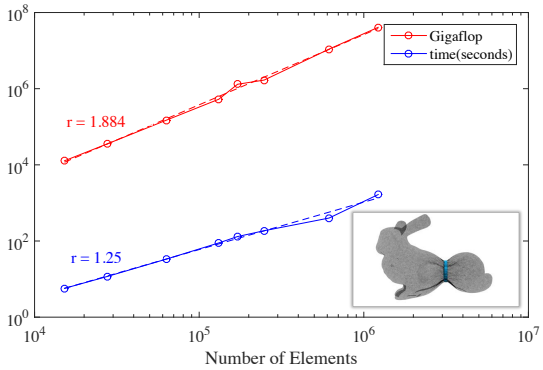



**Figure 4:** *Scalability of our method. We executed the bunny deformation example with resolutions ranging from $10^4$ to $10^6$ tets, measuring the number of floating point operations and compute time. Linear regression (dashed lines, slope r) shows that our method scales sub-quadratically.*

| model | $d$ | $n$ | $m$ | our method | | | YALMIP + MOSEK SDP | | | CVX+MOSEK Dual Solver | | | YALMIP + MOSEK SOCP | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | | #iter. | per-iteration time | total | modeling | solving | total | modeling | solving | total | modeling | solving | total |
| Arm | 2 | 2135 | 3945 | 20 | 20.4ms | 0.409s | 12.922s | 1.706s | 14.628s | 34.686s | 0.596s | 35.282s | 18.206s | 1.246s | 19.452s |
| Disc | 2 | 2483 | 4844 | 48 | 25.6ms | 1.228s | 15.962s | 5.663s | 21.625s | 48.064s | 0.843s | 48.907s | 23.036s | 1.628s | 24.664s |
| Caterpillar | 2 | 3744 | 7027 | 34 | 32.2ms | 1.096s | 30.295s | 7.685s | 37.980s | 87.501s | 1.123s | 88.624s | 41.457s | 2.372s | 43.829s |
| Worm | 2 | 10208 | 19888 | 25 | 84.2ms | 2.105s | 245.803s | 40.141s | 285.944s | 519.225s | 3.307s | 522.532s | 292.824s | 6.770s | 299.595s |
| Woody | 2 | 11290 | 22065 | 33 | 106.6ms | 3.518s | 291.825s | 50.053s | 341.878s | 660.936s | 4.538s | 665.474s | 366.795s | 10.522s | 377.318s |
| Yogurt Swirl | 2 | 13122 | 25600 | 43 | 98.2ms | 4.220s | 208.112s | 55.476s | 263.588s | 759.679s | 4.604s | 764.284s | 250.920s | 8.784s | 259.705s |
| Starry Night | 2 | 30502 | 60000 | 91 | 278.8ms | 25.369s | 1333.150s | 272.348s | 1605.498s | 5217.002s | 38.736s | 5255.739s | 2012.312s | 33.561s | 2045.873s |
| Sphere | 3 | 1061 | 4758 | 39 | 51.3ms | 2.002s | 69.995s | 9.760s | 79.756s | 65.531s | 3.886s | 69.417s | N/A | N/A | N/A |
| Hippo | 3 | 2387 | 8406 | 34 | 94.2ms | 3.202s | 203.202s | 24.273s | 227.475s | 169.738s | 19.996s | 189.735s | N/A | N/A | N/A |
| Bar | 3 | 2704 | 10800 | 58 | 123.0ms | 7.132s | 195.774s | 24.540s | 220.314x | 188.308s | 10.193s | 198.501s | N/A | N/A | N/A |
| Bunny | 3 | 4804 | 15202 | 36 | 0.159s | 5.706s | 908.782s | 43.438s | 952.220s | 248.285s | 14.251s | 262.536s | N/A | N/A | N/A |
| Bunny | 3 | 8700 | 27639 | 41 | 0.285s | 11.704s | 3411.283s | 176.116s | 3587.398s | 386.077s | 18.841s | 404.918s | N/A | N/A | N/A |
| Bunny | 3 | 19443 | 63040 | 47 | 0.710s | 33.374s | 20742.607s | 707.191s | 21449.798s | 2896.373s | 44.988s | 2941.360s | N/A | N/A | N/A |
| Bunny | 3 | 40084 | 131524 | 55 | 1.617s | 88.962s | >12hrs,quit | N/A | N/A | 21228.970s | 320.001s | 21548.971s | N/A | N/A | N/A |
| Bunny | 3 | 52434 | 171001 | 49 | 2.671s | 130.881s | >12hrs,quit | N/A | N/A | 36060.587s | 258.927s* | 36319.514s | N/A | N/A | N/A |
| Bunny | 3 | 76379 | 247884 | 58 | 3.207s | 186.016s | >12hrs,quit | N/A | N/A | >12hrs,quit | N/A | N/A | N/A | N/A | N/A |
| Bunny | 3 | 131451 | 615068 | 37 | 10.829s | 400.665s | >12hrs,quit | N/A | N/A | >12hrs,quit | N/A | N/A | N/A | N/A | N/A |
| Bunny | 3 | 261018 | 1223023 | 45 | 36.872s | 1659.248s | >12hrs,quit | N/A | N/A | >12hrs,quit | N/A | N/A | N/A | N/A | N/A |

**Table 1:** *Evaluation of our method on several 2D and 3D deformation examples. $d$ is dimension, $n$ is number of vertices and $m$ is number of elements. #iter. refers to the* total *number of Newton iterations in our method. In some high resolution Bunny examples, YALMIP or CVX failed to terminate within 12 hours of computation and therefore we were unable to run the solver on these examples. In YALMIP + SOCP, we followed the work of [Lip12] which considers 2D deformations only. *Interestingly, in this case the compute time slightly decreased after increasing the resolution. This is because the MOSEK solver terminated after fewer iterations.*

introduces potential infeasibility, because the targeting constraints may be incompatible with our ex-rotated non-inversion constraints. The soft formulation of the targeting constraints gracefully degrades when the user attempts to prescribe an infeasible configuration.

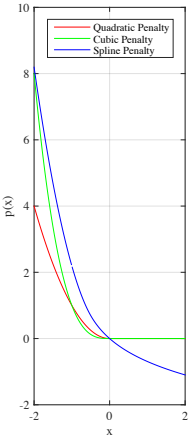
In addition to the quadratic penalty function (Eq. (5)), we experimented with several other penalty functions. In particular, we tested a cubic penalty function:

$$p_{\text{cubic}}(x,\tau) = \begin{cases} \tau(-(x-\varepsilon)^3) & \text{if } x < \varepsilon \\ 0 & \text{if } x \geq \varepsilon \end{cases}$$

which has the advantage of being $C^2$ continuous, instead of only $C^1$ continuous as Eq. (5). We also tried a "spline" penalty function:

$$p_{\text{spline}}(x,\tau) = \begin{cases} \frac{\tau(x-\varepsilon))^2 - \log(0.5) - 0.5}{0.5\tau} & \text{if } 2\tau(x-\varepsilon) < -1 \\ -\frac{log(\tau(x-\varepsilon)+1)}{\tau} & \text{if } 2\tau(x-\varepsilon) \geq -1 \end{cases}$$

In contrast to the previous two penalty functions, the spline penalty is strictly convex on the entire domain. The individual penalty functions are visualized in the inset figure to the right.




However, our experiments indicate that these special properties of the penalty functions do not make a significant difference in the results. This can be seen in Fig. 5, where we compare the three types of our penalty functions. The quadratic and spline penalties lead to approximately the same number of iterations, while the cubic penalty requires slightly more iterations. We use the quadratic penalty in all of our experiments,

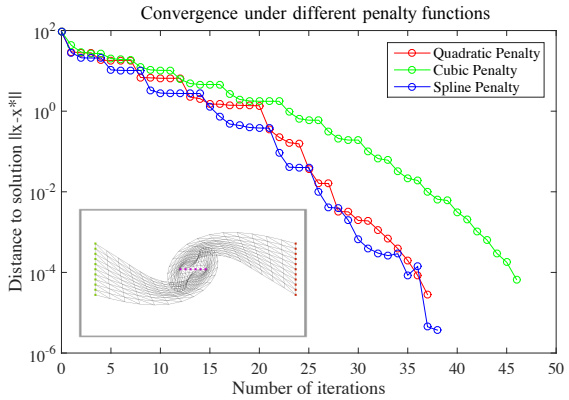



**Figure 5:** *Comparison of convergence for different penalty functions. The algorithm converges to the same result in all three cases.*

because its simple form opens up opportunities for performance optimization (which we defer to future work). Note that by "number of iterations" we always mean the *total* number of Newton steps, i.e., the number of iterations of lines 3-12 in Algorithm 1. Unless specified otherwise, we always use the default parameter values in Algorithm 1. The Hessian and gradient used in Algorithm 1 is computed analytically, using the implicit functions theorem to differentiate the eigen-decomposition [SB12].

An important question is how many iterations our algorithm needs as a function of the number of degrees of freedom (mesh vertices). In Fig. 6, we study the convergence of our algorithm on the same shape deformation scenario but with varying mesh resolutions. We can see that our method scales favorably with increasing numbers

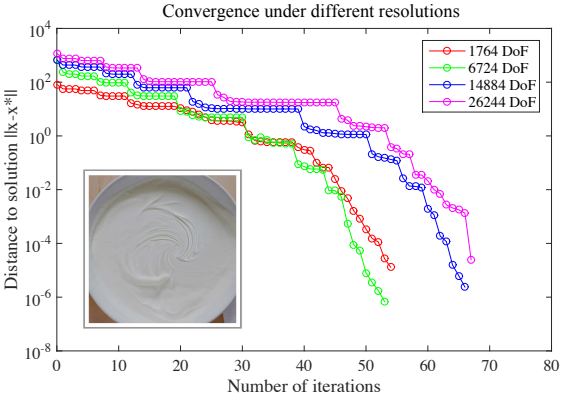



**Figure 6:** *Convergence of our method on a single deformation scenario with varying mesh resolutions.*

of degrees of freedom – the total number of iterations of our solver increases only slightly.

We also tested how the number of iterations changes on a fixed mesh, but with varying complexity of deformations. In Fig. 7, we examine progressively more and more difficult deformations of "yogurt swirl": 90, 180, and 270 rotation of the center of a square mesh. As expected, with increasing complexity, our algorithm requires more and more iterations to convergence. In the most difficult case, 270 degrees, the triangle mesh deformation is already close to its limits – going beyond 270 degrees would require us to compromise the targeting constraints. However, even in this challenging case, the total number of iterations is still relatively small: less than 70 in this case.

Our solver can be interpreted as a special-purpose semi-definite programming (SDP) method. This invites comparison with convex optimization software used in previous work. Following [KABL14], we use a combination of YALMIP [Lof04] and MOSEK. YALMIP is a modeling software which converts our

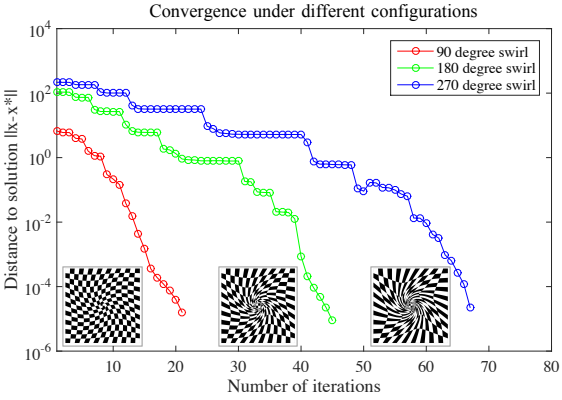



**Figure 7:** *Convergence of our method with a fixed mesh and three different configurations of the targeting constraints. Harder deformations require more iterations.*

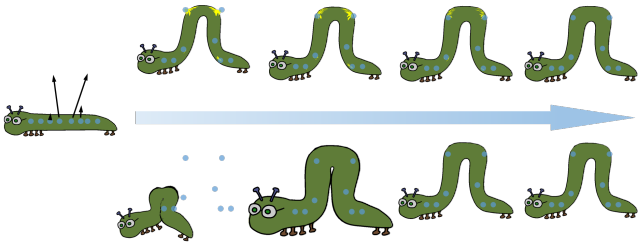


**Figure 8:** *The top row visualizes the convergence path of our method at iterations 5, 10, 25 and 33. The bottom row shows the convergence path of MOSEK at iterations 2, 5, 8, 15 (we attempted to find the most visually similar states).*

problem to a form acceptable by MOSEK. Of course, our problem has special structure and it would be possible to use a specialized modeling strategy, more efficient than general-purpose tools such as YALMIP. However, even if we completely disregard the time taken by YALMIP, our algorithm is still much faster than MOSEK, see Tab. 1. In all cases, both MOSEK and our method produce the same solution. Another advantage of our method is that even the intermediate iterates, i.e., before reaching convergence, are already visually similar to the final solution, which gives the user an opportunity to interactively preview whether the shape deformation behaves as intended. This is not the case with MOSEK, which approaches the solution via a much less natural convergence path, see Fig. 8. As we can see, the intermediate iterates of MOSEK scale the mesh in a non-intuitive way.

Our optimization problem can be also solved using augmented Lagrangian methods. These methods are related to penalty methods, but they add an estimate of Lagrange multipliers, which is updated along with the penalty weights [NW06]. The advantage is that the resulting linear systems are better conditioned, as it is not necessary to increase the penalty weight $\tau$ to infinity. Two examples of augmented Lagrangian-type solvers are the Spectral quadratic-SDP method [ANTT04] and PENNON [KS03]. Both of them can be adapted to our problem, and we compare their performance to our

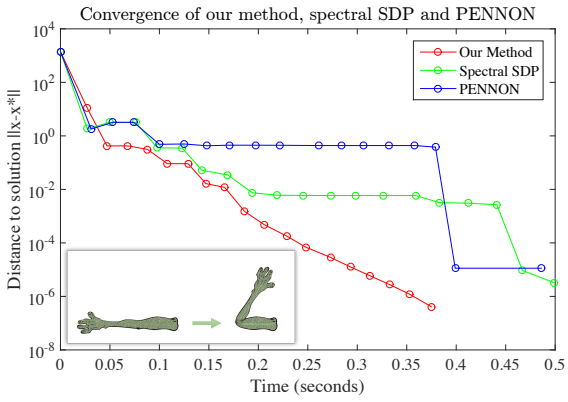



**Figure 9:** *Our method outperforms Augmented Lagrangian Methods: Spectral SDP [ANTT04] and PENNON [KS03].*

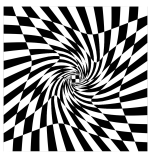
|  | model | $n$ | $m$ | Our method | | Spectral SDP | | PENNON | |
|---|---|---|---|---|---|---|---|---|---|
| | | | | #iter. | total time | #iter. | total time | #iter. | total time |
| | Quad20x20 | 882 | 1600 | 28 | 0.211s | 37 | 0.231s | 52 | 0.373s |
| | Quad40x40 | 3362 | 6400 | 25 | 0.606s | 64 | 1.63s | 49 | 1.512s |
| | Quad60x60 | 7442 | 14400 | 34 | 1.74s | 74 | 4.168s | 31 | 2.754s |
| | Quad80x80 | 13122 | 25600 | 41 | 3.991s | 63 | 6.313s | 35 | 5.485s |

**Table 2:** *Comparison to Augmented Lagrangian Methods: Spectral SDP [ANTT04] and PENNON [KS03]. We tested the quad example (shown on the left) using different resolution meshes. In all cases, our method is faster and enables interactive preview.*

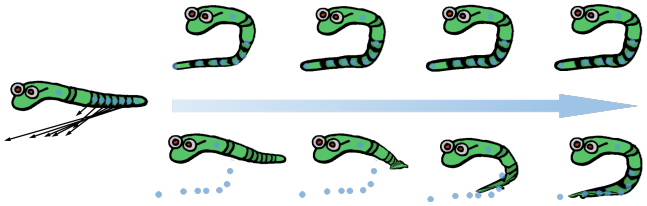


**Figure 10:** *The top row visualizes the convergence path of our algorithm at iterations 1, 5, 10, 25 (converged). The bottom row shows the convergence path of Locally Injective Mappings at iterations 1, 200, 1000, 3000 (not converged).*

method in Fig. 9 and Tab. 2. Although all methods converge to the same result and are relatively competitive, our approach is faster. The reason is that in our settings, the benefits of the augmented Lagrangian methods are diminished by the overheads associated with using and re-computing the estimates of Lagrange multipliers. Because we avoid this extra step, our method is easier to implement. We believe that this simplicity combined with high robustness and efficiency will prove useful in applications and future extensions.

Our method also invites comparison to Locally Injective Mappings (LIM) [SKPSH13], because LIM is another interactive shape deformation tool which avoids inversions. However, LIM is solving a different optimization problem: it imposes non-negativity constraints directly on $\det(\mathbf{F})$. This is more challenging because the resulting problem is non-convex. In situations such as in Fig. 10, LIM requires a very high number of iterations, where each step makes only minuscule progress towards the solution: we terminated LIM after 3000 iterations without reaching even a local minimum. Producing an analogous deformation with our method requires only 25 iterations using our algorithm (Fig. 10).

We also compared our method with Large-Scale Bounded Distortion Mappings [KABL15], a very recent inversion-free shape deformation method. For comparison purposes, we replaced the original quasi-conformal constraints with our ex-rotated non-inversion constraints. In deformation examples where most of the elements are not inverted, [KABL15] is significantly faster than our method. With moderate deformations, the performance of [KABL15] is comparable to our method. In challenging cases where the non-inversion constraints are hard to satisfy, [KABL15] may produce oscillatory behavior, while our method still converges reliably.

A logical question is whether our algorithm would directly extend to the non-convex $\det(\mathbf{F}) \geq 0$ regime. We attempted to apply our penalty functions directly to $\det(\mathbf{F})$. In addition to the fact that we lose global optimality guarantees (we can only hope to find a local minimum at best), we observed the number of iterations required to converge even to a local minimum increases drastically. In Fig. 12, we show an example where the above mentioned strategy failed to converge to a local minimum even after 1000 iterations, whereas our method converged to a global optimum in 50 iterations. This result highlights the benefits of our convex ex-rotated energy formulation.

It is straightforward to generalize our method to different convex quadratic elastic potentials. For example, we can add a linearized approximation of incompressibility [GMS14]:

$$\frac{1}{2}\sum_i \text{trace}^2(\hat{\mathbf{R}}_i^T \mathbf{F}_i - \mathbf{I}) \tag{9}$$

Adding this term to Eq. (1) encourages the elements preserve volume, as most real-world materials do. The problem is that upon significant extension, the material "shrinks" too much and many elements invert, see Fig. 11(b). Our method produces much more realistic results, shown in Fig. 11(c).

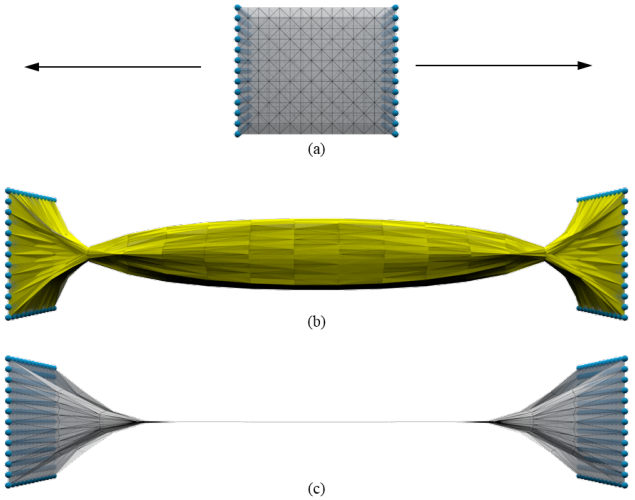



**Figure 11:** *Stretching an elastic bar (a) with ex-rotated elasticity and non-zero Poisson ratio produces many inverted elements (b). The same energy augmented with our ex-rotated non-inversion constraints produces a much more natural result (c).*

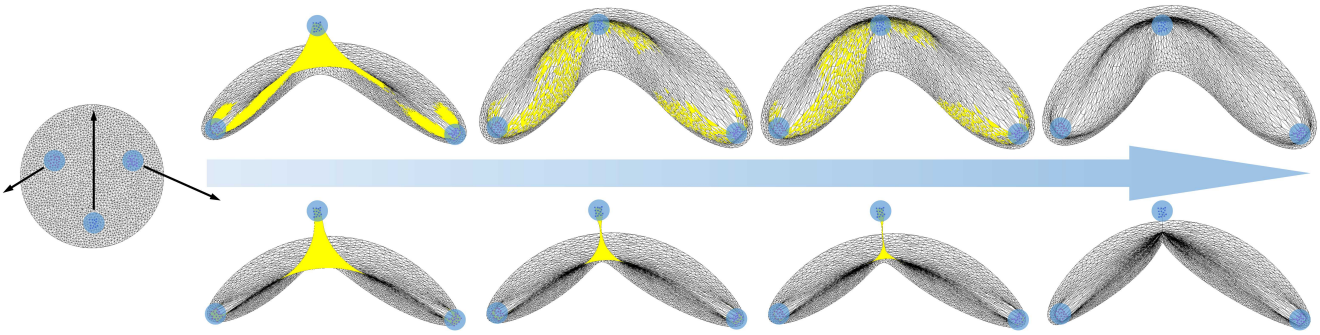


**Figure 12:** *The top row shows the convergence path of our method at iterations 5, 15, 30, 50 (converged). The bottom row shows the convergence path of a modified version of our method which applies the quadratic penalty functions directly to* $\det(\mathbf{F})$. *We show results at iterations 5, 15, 30, 1000 (still not converged).*

In some 3D shape deformation cases, correcting the interior inverted elements also changes the surface mesh. Fig. 13 shows an example of a better surface mesh when we enable our non-inversion constraints.

Finally, inspired by van Gogh's Starry Night painting, we attempted to create a similar effect by deforming a photograph. Unlike ex-rotated skinning, our inversion-free approach is able to produce the desired result, see Fig. 14.

## 5. Limitations and Future Work

One limitation of our method is the assumption that an appropriate ex-rotation field is specified by the user. In our current implementation we use handle rotations interpolated using bounded biharmonic weights, which works very well in skinning, but may not be ideal in other applications which require inversion-free mappings.

Our algorithm does not rely on any initial guess provided by the user, because we always start from ex-rotated skinning. On one hand, this means that we are not taking advantage of previously computed states. On the other hand, if we are rendering an entire animation, this "history-independece" allows us to compute each animation frame in parallel – a coveted feature in production environments. This trivial parallelism is typically elusive in physics-based animation where the result of the next frame depends on the solution of the previous frame.

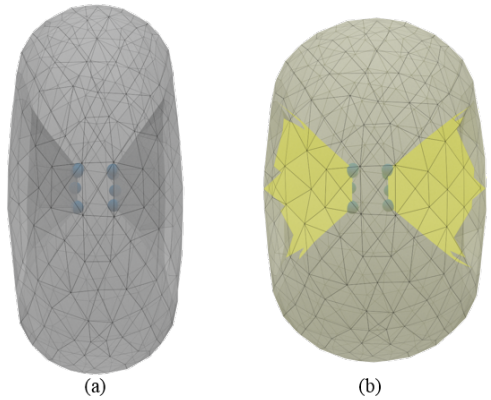



**Figure 13:** *Pinching an elastic sphere: (a) our result shows natural deformation without inverted elements, (b) ex-rotated skinning exhibiting inversions.*

In the future we would like to conduct a more rigorous study of feasibility. Our preliminary experiments indicate that in some contrived cases (with very poor tessellation and highly varying ex-rotations) our semi-definite problem may be infeasible, even with soft targeting constraints. We never encountered this situation in practice even when we experimented with randomly generated ex-rotations. In future work we would like to investigate sufficient and necessary conditions for feasibility, especially when accounting also for hard targeting constraints.

As noted already in recent related work [SKPSH13, KABL14], preventing element inversions does not guarantee that there will be no self-intersections of the boundary. To avoid boundary self-intersections, we could employ standard collision detection and response methods. Modern global collision processing algorithms are very robust, but also time consuming [HVS*09, HPSZ11]. In the future, we plan to combine our algorithm with a boundary collision response method, e.g., building on the results of Gao et al. [GMS14].

## 6. Acknowledgement

Our special thanks belong to Nathan Marshak for initial investigation of this problem within the scope of his Masters thesis. We thank the anonymous reviewers for many useful comments and Alec Jacobson, Shahar Kovalsky, Yaron Lipman, Daniele Panozzo, and Steven Wright for fruitful discussions. We also thank Harmony Li for narrating the accompanying video. This research was supported by NSF CAREER Award IIS-1350330 and NSF Grants IIS-1253598, IIS-1407282. Lifeng Zhu has been partially supported by the National Science Foundation of China (NSFC) under Grant No. 61502096 and the Natural Science Foundation of Jiangsu Province under Grant No. BK20150634.

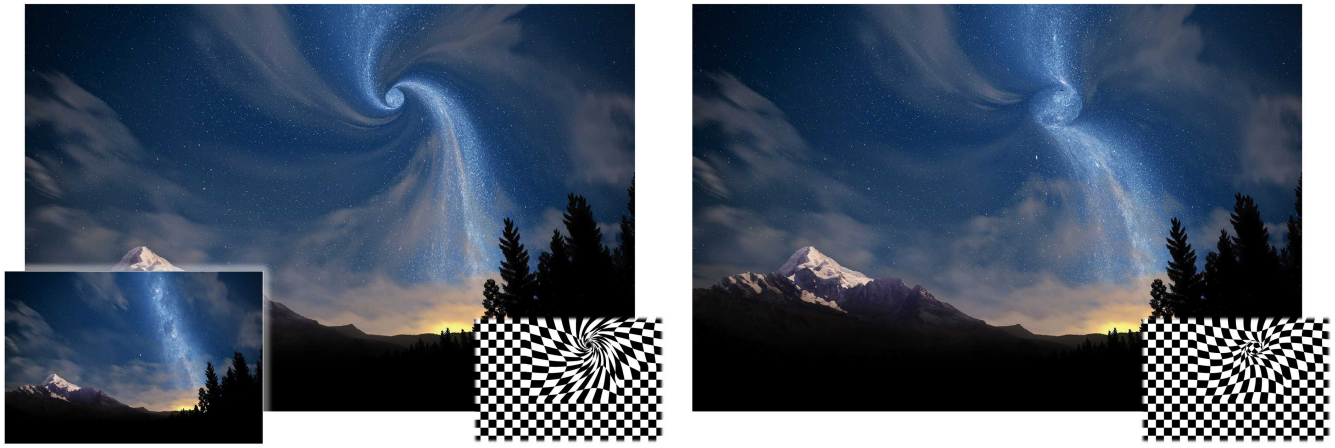



**Figure 14:** *Deformation of a photograph (shown in bottom-left) inspired by van Gogh's Starry Night painting: (a) our result, (b) ex-rotated skinning. Ex-rotated skinning fails to produce the desired effect due to the presence of inversions.*

## References


[AA99] ANDERSEN E. D., ANDERSEN K. D.: The mosek interior point optimization for linear programming: an implementation of the homogeneous algorithm. Kluwer Academic Publishers, pp. 197–232. 4

[AL13] AIGERMAN N., LIPMAN Y.: Injective and bounded distortion mappings in 3d. *ACM Trans. Graph. 32*, 4 (2013), 106:1–106:14. 2

[ANTT04] APKARIAN P., NOLL D., THEVENET J.-B., TUAN H. D.: A spectral quadratic-sdp method with applications to fixed-order $H_2$ and $H_\infty$ synthesis. *Eur. J. Control 10*, 6 (2004), 527–538. 7, 8

[AS07] ANGELIDIS A., SINGH K.: Kinodynamic skinning using volume-preserving deformations. In *Proceedings of the ACM SIGGRAPH/Eurographics Symposium on Computer Animation* (2007), SCA '07. 3

[BCE*13] BOMMES D., CAMPEN M., EBKE H.-C., ALLIEZ P., KOBBELT L.: Integer-grid maps for reliable quad meshing. *ACM Trans. Graph. 32*, 4 (2013), 98. 3

[BL10] BORWEIN J. M., LEWIS A. S.: *Convex analysis and nonlinear optimization: theory and examples*. Springer Science & Business Media, 2010. 4

[BV04] BOYD S., VANDENBERGHE L.: *Convex optimization*. Cambridge university press, 2004. 2, 4, 5

[BW97] BONET J., WOOD R.: *Nonlinear Continuum Mechanics for Finite Element Analysis*. Cambridge University Press, 1997. 2

[BZK09] BOMMES D., ZIMMER H., KOBBELT L.: Mixed-integer quadrangulation. In *ACM Trans. Graph.* (2009), vol. 28, p. 77. 3

[CFS14] CIVIT-FLORES O., SUSÍN A.: Robust treatment of degenerate elements in interactive corotational fem simulations. In *Computer Graphics Forum* (2014), vol. 33, pp. 298–309. 2

[CGC*02] CAPELL S., GREEN S., CURLESS B., DUCHAMP T., POPOVIĆ Z.: Interactive skeleton-driven dynamic deformations. *ACM Trans. Graph. 21*, 3 (2002), 586–593. 1, 2, 3

[CPSS10] CHAO I., PINKALL U., SANAN P., SCHRÖDER P.: A simple geometric model for elastic deformations. *ACM Trans. Graph. 29*, 4 (July 2010), 38:1–38:6. 2, 3

[CZXZ14] CHEN X., ZHENG C., XU W., ZHOU K.: An asymptotic numerical method for inverse elastic shape design. *ACM Trans. Graph. 33*, 4 (2014). 2

[DMK03] DEGENER P., MESETH J., KLEIN R.: An adaptable surface parameterization method. In *Proc. International Meshing Roundtable* (2003). 3

[ERF*11] ESTURO J. M., RÖSSL C., FRÖHLICH S., BOTSCH M., THEISEL H.: Pose correction by space-time integration. In *VMV* (2011), pp. 33–40. 3

[FH05] FLOATER M. S., HORMANN K.: Surface parameterization: a tutorial and survey. In *Advances in Multiresolution for Geometric Modelling* (2005), Springer. 3

[GMS14] GAO M., MITCHELL N., SIFAKIS E.: Steklov-Poincaré Skinning. In *Eurographics/ ACM SIGGRAPH Symposium on Computer Animation* (2014), The Eurographics Association. 1, 2, 3, 8, 9

[HG00] HORMANN K., GREINER G.: MIPS: An efficient global parametrization method. In *Curve and Surface Design: Saint-Malo 1999*, Innovations in Applied Mathematics. Vanderbilt University Press, 2000. 3

[HG15] HUYNH L., GINGOLD Y.: Bijective Deformations in $\mathbb{R}^n$ via Integral Curve Coordinates. *ArXiv e-prints* (2015). 3

[HPSZ11] HARMON D., PANOZZO D., SORKINE O., ZORIN D.: Interference-aware geometric modeling. In *ACM Trans. Graph.* (2011), vol. 30, p. 137. 9

[HVS*09] HARMON D., VOUGA E., SMITH B., TAMSTORF R., GRINSPUN E.: Asynchronous contact mechanics. In *ACM Trans. Graph.* (2009), vol. 28, p. 87. 9

[ITF04] IRVING G., TERAN J., FEDKIW R.: Invertible finite elements for robust simulation of large deformation. In *Proceedings of the 2004 ACM SIGGRAPH/Eurographics Symposium on Computer Animation* (2004), SCA '04, pp. 131–140. 2

[JBPS11] JACOBSON A., BARAN I., POPOVIĆ J., SORKINE O.: Bounded biharmonic weights for real-time deformation. *ACM Trans. Graph. 30*, 4 (2011), 78. 1, 3

[JDKL14] JACOBSON A., DENG Z., KAVAN L., LEWIS J.: Skinning: Real-time shape deformation. In *ACM SIGGRAPH 2014 Courses* (2014). 2

[JHT14] JIN Y., HUANG J., TONG R.: Remeshing-assisted optimization for locally injective mappings. *Computer Graphics Forum 33*, 5 (2014), 269–279. 2

[KABL14] KOVALSKY S. Z., AIGERMAN N., BASRI R., LIPMAN Y.:

Controlling singular values with semidefinite programming. *ACM Trans. Graph. 33*, 4 (2014), 68:1–68:13. 1, 2, 3, 4, 5, 7, 9

[KABL15] KOVALSKY S. Z., AIGERMAN N., BASRI R., LIPMAN Y.: Large-scale bounded distortion mappings. *ACM Trans. Graph. 34*, 6 (Oct. 2015), 191:1–191:10. URL: http://doi.acm.org/10.1145/2816795.2818098, doi:10.1145/2816795.2818098. 2, 8

[KS03] KOČVARA M., STINGL M.: Pennon: A code for convex nonlinear and semidefinite programming. *Optimization methods and software 18*, 3 (2003), 317–333. 7, 8

[KŽ05] KAVAN L., ŽÁRA J.: Spherical blend skinning: a real-time deformation of articulated models. In *Proceedings of the 2005 symposium on Interactive 3D graphics and games* (2005), ACM, pp. 9–16. 3

[Lip12] LIPMAN Y.: Bounded distortion mapping spaces for triangular meshes. *ACM Trans. Graph. 31*, 4 (2012), 108:1–108:13. 2, 3, 5, 6

[Lip14] LIPMAN Y.: Bijective mappings of meshes with boundary and the degree in mesh processing. *SIAM Journal on Imaging Sciences 7*, 2 (2014), 1263–1283. 2

[Lof04] LOFBERG J.: Yalmip : a toolbox for modeling and optimization in matlab. In *Computer Aided Control Systems Design, 2004 IEEE International Symposium on* (2004), pp. 284–289. 7

[LZ14] LEVI Z., ZORIN D.: Strict minimizers for geometric optimization. *ACM Trans. Graph. 33*, 6 (2014), 185. 3

[NW06] NOCEDAL J., WRIGHT S.: *Numerical optimization*. Springer Science & Business Media, 2006. 4, 7

[PB13] PARIKH N., BOYD S.: Proximal algorithms. *Foundations and Trends in optimization 1*, 3 (2013), 123–231. 4

[PL14] PORANNE R., LIPMAN Y.: Provably good planar mappings. *ACM Trans. Graph. 33*, 4 (2014), 76:1–76:11. 2

[SB12] SIFAKIS E., BARBIČ J.: FEM simulation of 3D deformable solids: a practitioner's guide to theory, discretization and model reduction. In *ACM SIGGRAPH 2012 Courses* (2012), p. 20. 3, 6

[SH15] SCHNEIDER T., HORMANN K.: Smooth bijective maps between arbitrary planar polygons. *Computer Aided Geometric Design* (2015). 3

[SHF13] SCHNEIDER T., HORMANN K., FLOATER M. S.: Bijective composite mean value mappings. In *Proceedings of the Eleventh Eurographics/ACMSIGGRAPH Symposium on Geometry Processing* (2013), SGP '13, pp. 137–146. 3

[SHST12] STOMAKHIN A., HOWES R., SCHROEDER C., TERAN J. M.: Energetically consistent invertible elasticity. In *Proceedings of the ACM SIGGRAPH/Eurographics Symposium on Computer Animation* (2012), SCA '12, pp. 25–32. 2

[SKPSH13] SCHÜLLER C., KAVAN L., PANOZZO D., SORKINE-HORNUNG O.: Locally injective mappings. *Computer Graphics Forum 32*, 5 (2013), 125–135. 1, 2, 3, 4, 8, 9

[SPR06] SHEFFER A., PRAUN E., ROSE K.: Mesh parameterization methods and their applications. *Found. Trends. Comput. Graph. Vis. 2*, 2 (2006). 3

[SWM*14] SETALURI R., WANG Y., MITCHELL N., KAVAN L., SIFAKIS E.: Fast grid-based nonlinear elasticity for 2d deformations. In *Symposium on Computer Animation* (2014). 2

[vFTS06] VON FUNCK W., THEISEL H., SEIDEL H.-P.: Vector field based shape deformations. *ACM Trans. Graph. 25*, 3 (2006), 1118–1125. 3

[WMZ12] WEBER O., MYLES A., ZORIN D.: Computing extremal quasiconformal maps. *Computer Graphics Forum 31*, 5 (2012). 2

[WPP07] WANG R. Y., PULLI K., POPOVIĆ J.: Real-time enveloping with rotational regression. *ACM Trans. Graph. 26*, 3 (2007). 1, 2, 3

[WSLG07] WEBER O., SORKINE O., LIPMAN Y., GOTSMAN C.: Context-aware skeletal shape deformation. *Computer Graphics Forum 26*, 3 (2007), 265–274. 1, 2, 3

[WZ14] WEBER O., ZORIN D.: Locally injective parametrization with arbitrary fixed boundaries. *ACM Trans. Graph. 33*, 4 (2014), 75. 3