Eurographics Symposium on Geometry Processing 2011
Mario Botsch and Scott Schaefer
(Guest Editors)

Volume 30 (2011), Number 5

As-Killing-As-Possible Vector Fields for Planar Deformation

Justin Solomon Mirela Ben-Chen Adrian Butscher Leonidas Guibas

Stanford University

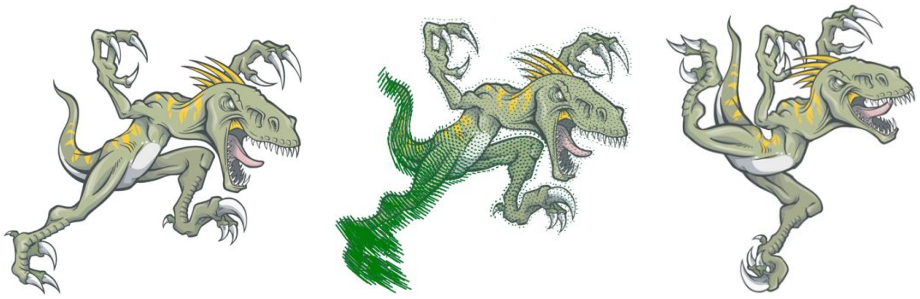


Figure 1: *A source model, an as-Killing-as possible vector field on that model, and a nearly-isometric deformation.*

Abstract

Cartoon animation, image warping, and several other tasks in two-dimensional computer graphics reduce to the formulation of a reasonable model for planar deformation. A deformation is a map from a given shape to a new one, and its quality is determined by the type of distortion it introduces. In many applications, a desirable map is as isometric *as possible. Finding such deformations, however, is a nonlinear problem, and most of the existing solutions approach it by minimizing a nonlinear energy. Such methods are not guaranteed to converge to a global optimum and often suffer from robustness issues. We propose a new approach based on* approximate Killing vector fields *(AKVFs), first introduced in shape processing. AKVFs generate near-isometric deformations, which can be motivated as direction fields minimizing an "as-rigid-as-possible" (ARAP) energy to first order. We first solve for an AKVF on the domain given user constraints via a* linear *optimization problem and then use this AKVF as the initial velocity field of the deformation. In this way, we transfer the inherent nonlinearity of the deformation problem to finding trajectories for each point of the domain having the given initial velocities. We show that a specific class of trajectories — the set of logarithmic spirals — is especially suited for this task both in practice and through its relationship to linear holomorphic vector fields. We demonstrate the effectiveness of our method for planar deformation by comparing it with existing state-of-the-art deformation methods.*

Categories and Subject Descriptors (according to ACM CCS): I.3.5 [Computer Graphics]: Computational geometry and object modeling—Geometric algorithms, Three-dimensional graphics and realism [I.3.7]: Animation—

1. Introduction

Planar shape deformation is a long-standing problem in computer graphics, with immediate uses in cartoon animation and image warping. The challenge in such applications is to build an intuitive tool that deduces desired deformations with minimal guidance while preserving small details.

The quality of a deformation can be measured by the distortion it introduces while attempting to satisfy user-specified constraints. In particular, a simple and reasonable model for planar deformation is that a shape should deform *isometrically* by preserving angles and areas. Strict planar isometries, however, only include rotations and translations, which are not rich enough to describe interesting manipulations. We thus seek *nearly*-isometric changes instead.

A deformation in the plane can be described as a map f applied to a domain Ω yielding a deformed shape $f(\Omega)$. A natural way to describe the distortion of f is to use its Jacobian, the 2×2 matrix of first partial derivatives J_f. This

matrix can be viewed as a transformation describing the local deformation applied at every point in Ω. Hence, if J_f at a point is orthogonal, the area near that point will undergo only rotation and translation. For this reason, many "as-rigid-as-possible" (ARAP) methods minimize the deviation of J_f from the set of rotation matrices. Inspired by a similar approach in shape interpolation [ACOL00], the seminal work by Igarashi et al. [IMH05] first used this idea for shape deformation. Their work has been extended in various ways in [SA07, SDC09, LZX*08, BCWG09, CPSS10] and others. The main obstacle to solving ARAP problems is that isometries are nonlinear. For example, the sum of two rotation matrices is not a rotation matrix. Thus, solving such problems requires nonlinear optimization, which can be slow and prone to local minima. See [BS08] for a discussion of the differences between linear and nonlinear deformation.

Other approaches make use of conformal maps, which guarantee that angles are preserved or, equivalently, that their Jacobians are similarity matrices. Conformal maps are linear in two dimensions, leading to efficient deformation methods. Their main downside, however, is that they do not preserve area, and enforcing zero angular distortion exactly can cause large global area scaling to occur.

We propose a new framework that uses small stepwise displacements rather than a global solve to generate new configurations. Instead of considering the map f directly, we solve for the initial velocities of deformation trajectories such that the resulting flow will be as rigid as possible. In this setup, the user specifies the velocities of the constrained points; in practice the interactive experience is similar to specifying target paths. The system solves an optimization problem to find a vector field $\vec{U}$ on Ω that is close to a Killing vector field [Pet10] while fulfilling the constraints. This optimization problem is *linear* in the velocities and thus can be solved robustly and efficiently. The original isometric deformation problem still is nonlinear, but we have shifted the nonlinearity to finding trajectories that integrate the field $\vec{U}$ over a short time step. Logarithmic spiral trajectories are suitable for this task, because they are flows of linear holomorphic vector fields and are natural extensions of rigid motions that nevertheless permit a controlled amount of dilation. In addition, matching logarithmic spirals to the required velocities can be expressed as a straightforward local problem.

Our framework for deformation is efficient and non-iterative, and it is comparable to state-of-the-art nonlinear systems. It is more robust than many conformal methods, avoiding their infamous large area distortions. We demonstrate the applicability of our method by comparing it with point-to-point Cauchy-Green coordinates [WBCG09] and as-rigid-as-possible deformation [LZX*08].

1.1. Contribution

We rephrase the problem of finding almost-isometric planar deformation in terms of velocities, resulting in a linear solve for the initial velocities of the deformation map combined with efficient logarithmic spiral trajectory fitting. We also show that logarithmic spirals are simply the flows of linear holomorphic vector fields, motivating their use for shape interpolation and other applications.

1.2. Previous Work

Shape deformation and specifically planar deformation have been active research topics in recent years, and a full survey of existing work is beyond the scope of this paper. We focus on recent approaches that are related to our own, which generally can be categorized as either *discrete* or *continuous*.

Discrete methods are based on discretizing the deformation domain, usually with a triangulation or a quadrangular grid. One of the first discrete approaches was introduced by Igarashi et al. [IMH05]. They assert that a deformation is intuitive if triangles move rigidly, meaning they only rotate and translate. Hence, their algorithm finds a deformation that fulfills the user's constraints, and then it transforms the triangles to be rotations of the triangles from the input mesh. Finally, the new triangles are "stitched" together to generate the deformed mesh. Successive as-rigid-as-possible approaches, such as [SA07, LZX*08], repeat this process until convergence. [SA07] shows that the iteration reduces a global energy and hence converges to a local minimum. [CPSS10] presents an elastic deformation energy that is a continuous analog of the as-rigid-as-possible metric and can be reduced using a Gauss-Newton solver. Other approaches, including [WXW*06], optimize contrasting deformation energies with similar advantages and drawbacks.

While effective in many scenarios, as-rigid-as-possible approaches suffer from two main drawbacks. First, although the process is guaranteed to converge, it might reach a suboptimal local energy minimum. Many local minima provide plausible deformations, but the deformation might alternate between *different* solutions for similar user constraints, yielding an unstable deformation. Furthermore, the robustness of the method degrades as the mesh is refined, requiring more iterations for convergence. Since a low-resolution mesh will exhibit discretization distortions, the second issue can be problematic for planar deformation.

The second category of planar deformation methods includes the continuous and *cage-based* methods. These methods are based on defining the deformation of a bounded domain Ω in the plane without requiring its discretization. For instance, [SMW06] averages linear transformations induced by moving constraint points. In more recent work, the deformation at a point $p \in \Omega$ is defined as a linear combination of basis functions associated with elements of the boundary $\partial\Omega$. The basis functions commonly are called *barycentric coordinates* and have been formulated many different ways. Those used for deformation include [Flo03, LKCOL07, JMD*07, LLCO08, WBCG09, WG10, HS08, MS10,

JBPS11]. These methods are efficient and robust, but their output is not close to isometry because the induced maps are either affine-invariant, allowing shears, or merely conformal. In fact, such linear approaches cannot generate isometries by definition, since isometries are nonlinear. A recent approach combines cage-based and as-rigid-as-possible methods [BCWG09], generating near-isometric mappings, but it suffers from poor convergence for large numbers of rigidity constraints, similar to other as-rigid-as-possible algorithms.

Our method is based on approximate Killing vector fields (AKVFs), and thus we also mention previous work in this domain. An exact KVF is a tangent vector field on a surface that generates an isometric deformation. KVFs are intrinsic and were first introduced for geometry processing applications in [BCBSG10], where the surface in question is a closed surface in $\mathbb{R}^3$. Exact intrinsic isometries of curved surfaces are rare, but one can find *approximate* KVFs as the minimizers of an energy functional (the "Killing energy") that measures deviation from being an exact KVF. Such vector fields were used for mesh segmentation in [SBCBG11]. We apply the AKVF methodology to domains in the plane. That is, we find vector fields that best approximate exact KVFs while fulfilling the user constraints.

It is worth mentioning another vector field approach using *divergence-free* fields [vFTS06]. This method, however, only guarantees that the deformation is area-preserving and does not attempt to minimize angular distortion. Although exact KVFs are divergence-free, approximate KVFs strike a balance between requiring the deformation to be as area-preserving and as angle-preserving as possible.

1.3. Method Overview

We get as input a planar triangulated mesh discretizing a 2D domain. The user marks a few points as handles (these can change during the interaction), and drags them to new locations, effectively prescribing velocities as the points move. We use the velocities to solve a linear system whose output is a vector field that is as close as possible to a KVF while matching the user's constraints. In a second step, we locally fit logarithmic spiral trajectories for all the vertices using the vector field to obtain initial velocities. The new position of the mesh is computed by following these trajectories.

2. Algorithm

2.1. As-Killing-As-Possible Vector Fields

A planar deformation is a mapping $f:\Omega\to\mathbb{R}^2$ defined on a domain $\Omega\subset\mathbb{R}^2$; the deformed domain is simply $f(\Omega)$. Instead of considering a single such map, however, we can look at a *one-parameter family* of deformations $F:\Omega\times[0,1]\to\mathbb{R}^2$ where $F(x,t)$ gives the location of a point $x\in\Omega$ at time t. The deformed domain at time t is $F(\Omega,t)$, and the final deformed shape is $F(\Omega,1)$. The advantage of this representation is that it allows us to infer properties of the deformation from its *velocity field* $\vec{U}(p,t)=\frac{\partial F}{\partial t}(p,t)$.

For example, consider rotation by $\frac{\pi}{2}$ about the origin as part of a family of deformations given by $F(p,t)=\begin{pmatrix}\cos(t) & -\sin(t)\\ \sin(t) & \cos(t)\end{pmatrix}p$ with $t\in[0,\frac{\pi}{2}]$. The velocity of this family is then $\frac{\partial F}{\partial t}(p,t)=\begin{pmatrix}-\sin(t) & -\cos(t)\\ \cos(t) & -\sin(t)\end{pmatrix}p=RF(p,t)$, where R is a rotation by angle $\pi/2$ about the origin. So, the velocity of the family of deformations is the vector field $\vec{U}$ on the plane defined by $\vec{U}(p)=R\cdot p$ or $\vec{U}(x,y)=(-y,x)$. Note that the Jacobian $J_{\vec{U}}=R=\begin{pmatrix}0 & -1\\ 1 & 0\end{pmatrix}$ is anti-symmetric.

More generally, suppose f minimizes the as-rigid-as-possible (ARAP) energy $E(f)=\frac{1}{2}\int_\Omega\|J_f-R\|^2$ from [CPSS10], where $J_f:\Omega\to\mathbb{R}^{2\times2}$ is the Jacobian of f and $R:\Omega\to SO(2)$ minimizes $\|J_f-R\|$. As noted in [CPSS10], other ARAP algorithms including [SA07,IMH05] optimize a similar discrete energy. In the one-parameter case, for small Δt we define $f(p)=F(p,\Delta t)$, and thus $f=\mathrm{id}+\vec{U}$ for identity map id and small vector field $\vec{U}:\Omega\to\mathbb{R}^2$. Then, to first order ARAP minimizes $\int_\Omega\|J_{\vec{U}}-S\|^2$ for antisymmetric S. That is, ignoring regularization and data-matching, small ARAP displacements have Jacobians that are as antisymmetric as possible.

Vector fields on a surface S with anti-symmetric Jacobians are called *Killing vector fields* (KVFs). One can show that any KVF generates an isometric deformation of S essentially by reversing the process described in the rotational example (i.e. integrating the KVF to find a one-parameter family of transformations whose velocity is the KVF). When $S=\mathbb{R}^2$, KVFs must have *constant* Jacobian [Pet10], giving $\vec{U}(x,y)=a(-y,x)+d$ for some $a\in\mathbb{R}$ and $d\in\mathbb{R}^2$. It is then easy to deduce that all Killing vector fields on $\mathbb{R}^2$ can be written:

$$\vec{U}(p)=\begin{cases}aR\cdot(p-c) & \text{if } a\neq0\\ d & \text{if } a=0.\end{cases}\tag{1}$$

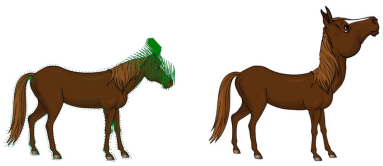
where $c=Rd/a$. The isometries generated by these KVFs are either $F(p,t)=c+R^{at}(p-c)$ or $F(p,t)=p+td$, where now we use the notation R^θ for the rotation by an angle θ about the origin. Thus, the vector fields in (1) generate either the rigid motions that rotate with angular velocity a about $c\in\mathbb{R}^2$ or that translate in the direction $d\in\mathbb{R}^2$, respectively.

Since KVFs are characterized by their anti-symmetric Jacobians, we can measure a vector field's deviation from being Killing using its *Killing energy*:

$$E_K(\vec{U})=\int_{p\in\Omega}\|J_{\vec{U}}(p)+J_{\vec{U}}(p)^T\|_{Fro}^2\tag{2}$$

This energy is a special case of the surface-based Killing energy of [BCBSG10]. It differs from the ARAP energy of [CPSS10] by a cubic term in $J_{\vec{U}}$, which is insignificant as $\vec{U}\to\vec{0}$. In particular, the integrand of (2) can be written $4\|J\|^2-\|J-J^T\|^2$. Since we are perturbing the identity map

we have $f = \mathrm{id}$, $J_f = I$, and $R = I$. The map f trivially is an isometry and thus is a zero of the first variation of the ARAP energy. Our energy E_K is a quadratic form from the second variation $\delta^2_{\vec{U},\vec{U}} E(f) = \int_\Omega \left[\langle J, J\rangle - \langle J, \delta_{\vec{U}} R\rangle\right]$ scaled by $1/4$ after applying (3) in [CPSS10], where $\langle A, B\rangle = \mathrm{tr}(A^T B)$.



(a) (b)

Figure 2: *Following the AKVF (a) for a short distance begins to straighten the horse's neck, but following the field too far without updating can create distortions (b).*

In a typical deformation scenario, the user poses a number of constraints, and then a nearly-isometric deformation is found fulfilling these constraints. If we use velocities for the underlying computations, specifying starting and ending positional constraints is not enough; for instance, Figure 2 shows an example in which following a single vector field too far can create unnecessary distortion. Instead, we use a deformation metaphor where the user prescribes *velocities* rather than positional constraints; these velocities are updated repeatedly as the user moves constrained vertices and are used to generate small stepwise displacements. In a single step, given the user's prescribed velocities $\vec{u}_i$ for a set of points $C = \{p_i \in \Omega\}$, we construct a velocity field which can be used to generate the deformation. Since we are aiming for close-to-isometric deformations, we search for a vector field which is "as-Killing-as-possible" given the constraints.

In particular, we can find an as-Killing-as-possible vector field while respecting given constraints by minimizing:

$$\vec{U}^\circ = \arg\min \left(E_K(\vec{U}) + \lambda \sum_{p_i \in C} \|\vec{U}(p_i) - \vec{u}_i\|^2 \right) \qquad (3)$$

where λ is a constant that balances the user's constraints and the isometry of the deformation.

(a)
(b)
(c)

Figure 3: *The field (3) moving the vertices in (a) to the locations in (b) creates singularities around the constraints; the Dirichlet solve (4) has a smoother displacement (c), preserving the boundary but relaxing the locations of the constraints to the red points. The solve here uses hard constraints to find $\vec{U}^\circ$; soft constraints yield similar singularities.*

The solution of this optimization problem is not guaranteed to be smooth near the constraints since their locations are sparse in Ω; Figure 3b shows the type of singularity that can occur. Therefore we use it to find the vector field on the *boundary*, where smoothness is guaranteed by elliptic regularity [GT01], and solve again using these boundary values as Dirichlet boundary conditions:

$$\vec{U}_{opt} = \arg\min \left(E_K(\vec{U}) \text{ s.t. } \vec{U} = \vec{U}^\circ \text{ on } \partial\Omega \right) \qquad (4)$$

Since this equation is the variational form of an elliptic partial differential equation with smooth boundary data, its solution must be smooth; an identical argument shows the smoothness of harmonic extensions of a function to the interior of a domain. This last step does not enforce the user's constraints directly, but we have found that the behavior of the boundary largely prescribes the behavior of the shape, thus approximately fulfilling the user's constraints as in Figure 3c. We can also show that our solution has an important reproduction property: if the constraints are compatible with a unique KVF $\vec{W}$ (and thus the user's constraints represent a rigid motion) then $\vec{U}_{opt} = \vec{W}$ and we reproduce that rigid motion using the proper integration method (Appendix A).

Figure 4 shows a few examples of such vector fields given user constraints. It is evident that the resulting vector field approximately fulfills the constraints and is restricted to the parts of the shape upon which it should act. For example, when trying to move the tail of the cat model, the vector field has nearly zero norm on the rest of the model.

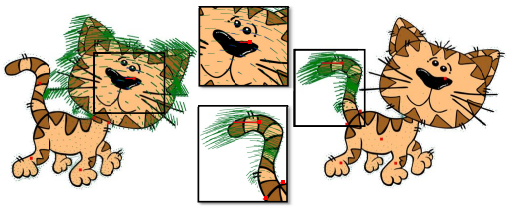


Figure 4: *Examples of as-Killing-as-possible vector fields (in green). The field computed from the user's constraints is in blue; red points are constrained. In both examples a single constraint was moved, either the head (left) or tail (right).*

2.2. Discretization

The procedure for finding $\vec{U}_{opt}$ can be implemented as follows. Given a mesh $M = (V, F, E)$ with vertices V, faces F, and edges E, the user selects a set of vertices $C = \{v_i \in V\}$ and drags them to a different location. As the vertices are moved, we interpret the difference between their current positions and the previous positions as the set of velocity constraints for a given time step. Hence, our deformation metaphor is slightly different from the standard one, but from the user's perspective, the manipulation is just as intuitive as the standard position-based one. See the attached video for several sample interactive sessions.

We discretize (3) and (4) using a standard finite element approach. The discrete vector field is given by two scalar

functions $U_1, U_2 : V \to \mathbb{R}$ interpolated piecewise-linearly to the whole domain. The partial derivatives of the vector field are given by the derivatives of its components, which can be computed using the gradient of a piecewise linear scalar function on triangles (see e.g. [BKP*10], Chapter 3, or [PP93]). These gradients are constant on the faces, and hence for each face we have four values: two components for each of the gradients of U_1 and U_2. These provide the ingredients for building an over-constrained sparse linear system whose solution is the optimal vector field $\vec{U}_{opt}$.

In contrast to the treatment of KVFs on curved surfaces in [BCBSG10, SBCBG11], the discretization in our case is straightforward, since the domain is planar and we can represent the vector field using its two components. Our optimization problem has $2n$ variables, where $n = |V|$. The gradient of a piecewise linear scalar function f can be written as $\nabla f = Gf$, where $G \in \mathbb{R}^{2m \times n}$, where $m = |F|$, yielding two components for the gradient for every face. The Jacobian of the vector field can thus be represented as the matrix concatenation $[GU_1; GU_2]$, where we have a 4×1 vector, instead of a 2×2 matrix, for every face.

Given a Jacobian matrix $J = \left(\begin{smallmatrix} a & b \\ c & d \end{smallmatrix}\right)$, the energy in our first optimization problem (3) tries to minimize the Frobenius norm of its symmetric part $J + J^T = \left(\begin{smallmatrix} 2a & b+c \\ b+c & 2d \end{smallmatrix}\right)$ subject to the user constraints. Hence, our discretization of (3) is:

$$\min \left\| \begin{bmatrix} P \\ \lambda I_k \end{bmatrix} \begin{bmatrix} U_1^\circ \\ U_2^\circ \end{bmatrix} - \begin{bmatrix} 0 \\ \lambda \tilde{U} \end{bmatrix} \right\|_{Fro}^2 \tag{5}$$

where

$$P = \begin{bmatrix} 2G & 0 \\ \sqrt{2}G & \sqrt{2}G \\ 0 & 2G \end{bmatrix}_{6m \times 2n} \quad I_k = \begin{bmatrix} I & 0 \\ 0 & I \end{bmatrix}_{2k \times 2n} \quad \tilde{U} = \begin{bmatrix} \tilde{U}_1 \\ \tilde{U}_2 \end{bmatrix}_{2k \times 1}. \tag{6}$$

Here $I \in \mathbb{R}^{k \times n}$ ($k = |C|$) contains the rows of an identity matrix corresponding to the constrained vertices, and $\tilde{U}_1, \tilde{U}_2$ are the components of the vector constraints. The minimizer of (5) is given by the solution of the normal equations:

$$\left[P^T P + \lambda^2 I_k^T I_k\right] \begin{bmatrix} U_1^\circ \\ U_2^\circ \end{bmatrix} = \lambda^2 I_k^T \tilde{U} \tag{7}$$

This is a sparse positive definite linear system, which can be solved efficiently using Cholesky factorization [CDHR08].

Next, given $\vec{U}^\circ = (U_1^\circ, U_2^\circ)$, we minimize the Killing energy again using $\vec{U}^\circ$ for Dirichlet boundary conditions:

$$\min \left\| P \begin{bmatrix} U_1 \\ U_2 \end{bmatrix} \right\|_{Fro}^2 \text{ s.t. } \begin{bmatrix} U_1 \\ U_2 \end{bmatrix}_{\mathcal{B}} = \begin{bmatrix} U_1^\circ \\ U_2^\circ \end{bmatrix}_{\mathcal{B}} \tag{8}$$

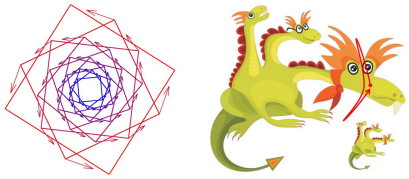
where $\mathcal{B}$ contains the indices of the boundary vertices and P is the same as in (5) and (7); the values of U° from the interior of the mesh are unused in this step. This problem can be solved efficiently by replacing the variables corresponding to the boundary vertices with constants determined by the boundary conditions and removing them from the system. Such boundary conditions yield a slightly different sparsity pattern for the normal equations from (8), although both this sparsity pattern and that of (7) depend only on mesh topology and (for (7)) the indices of the user-constrained vertices.

2.3. Logarithmic Spiral Trajectories

The last step in our method is to generate the deformed shape from the vector field $\vec{U}_{opt} = (U_1, U_2)$. We are now looking for trajectory curves $\gamma_p(t)$ for all the points p in the mesh, such that

$$\gamma_p(0) = p \text{ , and } \gamma_p'(0) = \vec{U}_{opt}(p) \tag{9}$$

A simple approach would be to set $\gamma_p(t) = p + t\vec{U}_{opt}(p)$. These linearized trajectories, however, have an important drawback: they cannot reproduce global rotations from rotational vector fields. Figure 5 shows examples of dilations resulting from the use of linear trajectories.



(a) (b)

Figure 5: *(a) The blue box is deformed using linear steps along the rotational field $\vec{U}(x,y) = (-y,x)$; the box grows due to the use of linear trajectories. (b) The monster's head (on the lower right) is deformed along the red path using small linear steps; the head grows considerably.*

We thus require more sophisticated trajectories that can reproduce isometric motion from exact rotational and translational Killing fields and that cleanly support deviations from isometry in approximate Killing fields. We use the following terminology: given a vector field $\vec{U} : \mathbb{R}^2 \to \mathbb{R}^2$, the *flow* of a point p under $\vec{U}$ is the curve $\gamma_p(t)$, such that:

$$\gamma_p(0) = p \quad \text{and} \quad \gamma_p'(t) = \vec{U}(\gamma_p(t)). \tag{10}$$

If $\vec{U}$ is a KVF, then its flows are either circles or straight lines. Thus, if we assumed that locally the vector field is given by an exact KVF, we would have an exact trajectory to follow. This constraint, however, is too strong, since it cannot allow even for small dilations. For example, given two points with arbitrary velocities, it is not possible to fit an exact KVF that generates these velocities, because already these two velocities can imply some dilation. An easy way to see this is to recall the general form of exact KVFs in the plane given in (1), which has at most three degrees of freedom. To fit two velocities at two points we require four.

Therefore, we suggest a more flexible approach fitting flows of a *linear holomorphic vector field* locally around p. Just as KVFs generate isometries, holomorphic vector fields

generate conformal maps. Conformal maps preserve angles but not areas and thus are more flexible and allow for dilations. However, since the vector field $\vec{U}_{opt}$ is as-Killing-as-possible, in general the dilations required are not expected to be big. Since the space of holomorphic vector fields is very large, we choose to fit *linear* holomorphic vector fields, which are the simplest holomorphic vector fields at our disposal, and, like KVFs, have constant Jacobian matrix.

Like KVFs, holomorphic vector fields are characterized by a condition on their Jacobian matrices, the Cauchy-Riemann equations. This condition can be written:

$$J + J^T - Trace(J) \cdot I = 0 \tag{11}$$

where I is the identity matrix. Thus, the Jacobian of a holomorphic vector field must be of the form $J = \begin{pmatrix} A & -B \\ B & A \end{pmatrix}$ at each point, and a linear holomorphic vector field must have the form $(U_1(x,y), U_2(x,y)) = (d_1 + Ax - By, d_2 + Bx + Ay)$, where $A, B, d_1, d_2 \in \mathbb{R}$. Simple manipulations allow us to put linear holomorphic vector fields in a form similar to (1):

$$\vec{U}_{a,\theta,c}(p) = \begin{cases} aR^{\theta}(p-c) & \text{if } A, B \text{ are not both zero} \\ d & \text{otherwise.} \end{cases} \tag{12}$$

Here $a, \theta \in \mathbb{R}$ and $d \in \mathbb{R}^2$. Since we allow an arbitrary rotation, we now have an additional degree of freedom. This allows us to better fit trajectories given velocities at two points.

Using complex numbers to simplify the notation (via the association of $(x,y) \in \mathbb{R}^2$ with $x + iy \in \mathbb{C}$), we can write linear holomorphic vector fields in one of two forms. The first, when A or B is nonzero, is most interesting. We can write

$$\vec{U}_{s,c}(p) = s(p-c) \text{ where } p, c, s \in \mathbb{C} \tag{13}$$

where $s \neq 0$. Thus $\vec{U}_{s,c}$ generates a combination of rotation and scaling of the complex plane. The flow of a point p under this vector field is given by the curve $\gamma_{c,s,p}$ such that $\gamma_{c,s,p}(0) = p$ and $\gamma'_{c,s,p}(t) = s(\gamma_{c,s,p}(t) - c)$. It is easy to check that the solution is:

$$\gamma_{c,s,p}(t) = c + e^{st}(p-c). \tag{14}$$

These curves are known as *logarithmic spirals*, or "natural curves" due to their abundance in nature [Coo14]. Recently, they have been used for interpolating 2D cartoons in [WNS*10]. It is interesting to note that "aesthetic" curves for animation are in fact the flows of linear holomorphic vector fields. Figure 6(a) shows an example of a log spiral, with parameters c, a and θ marked, where $a = \log(\mathbf{Re}(s))$ and $\theta = \mathbf{Im}(s)$. Figure 6(b) shows a family of such spirals.

The second form taken by a linear holomorphic vector field occurs when $A = B = 0$ and thus $\vec{U}_d(p) = d$ where $d = d_1 + id_2$. This corresponds to translations in the complex plane and the associated trajectories are $\gamma_p(t) = p + td$.

We now show how we fit the trajectories above to the vector field $\vec{U}_{opt}$. Suppose that at two neighboring points p_1 and p_2, the vector field has values $\vec{U}_{opt}(p_1) = u_1$ and

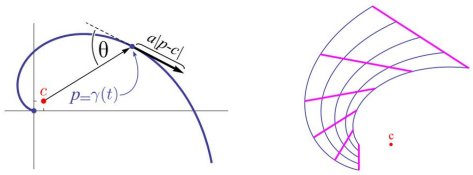


Figure 6: *(a) A logarithmic spiral; (b) a family of log spirals which are the flows of the holomorphic vector field $\vec{U}_{c,s}$.*

$\vec{U}_{opt}(p_2) = u_2$ where $u_1, u_2 \in \mathbb{C}$. We want to find the parameters s and c of a linear holomorphic vector field $\vec{U}_{s,c}$ such that $\vec{U}_{s,c}(p_1) = u_1$, and $\vec{U}_{s,c}(p_2) = u_2$. This boils down to solving two linear equations in two complex variables, yielding:

$$s = \frac{u_1 - u_2}{p_1 - p_2} \quad \text{and} \quad c = p_1 + u_1/s \tag{15}$$

In the special case $u_1 = u_2$, the linear holomorphic vector field reduces to the translation $\vec{U} = u_1$.

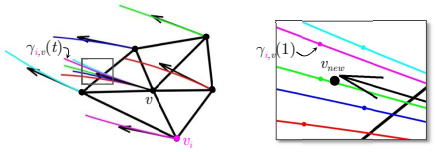


Figure 7: *Finding the new position of a vertex by fitting a linear holomorphic vector field to each edge, computing the spiral flows $\gamma_{i,v}$, and averaging the resulting positions.*

We were left with the problem of finding the deformed shape Ω_{new}, which we accomplish as follows. For each point $v \in V$ and one of its neighboring vertices v_i, we have found a linear holomorphic vector field that coincides with $\vec{U}_{opt}$ at these two points along with a corresponding flow. Thus each (v, v_i) pair yields a possible position for v_{new}, namely $\gamma_{i,v}(1)$; note we evaluate at $t = 1$ to have a time step for which the velocity constraints coincide reasonably well with differences in vertex displacements along their prescribed motion paths. Hence we average to find the final position:

$$v_{new} = \frac{1}{d(v)} \sum_{i=1}^{d(v)} \gamma_{i,v}(1) \tag{16}$$

where $d(v)$ is the degree of v. Figure 7 shows an illustration of this process. Note that we could have inverted the order of operations, averaging parameters s and c rather than endpoints; this can be unstable when vector fields are nearly translational, as the centers of rotation c may vary considerably while yielding the same approximately linear trajectory.

If $\vec{U}_{opt}$ is a global KVF, then it is also a special case of a linear holomorphic vector field. Thus, applying (15), the parameters c_i, s_i will be constant and unique across the mesh. This implies that our fitting process will reconstruct the motion of the mesh under this KVF, effectively guaranteeing that we reconstruct rotations and translations exactly.

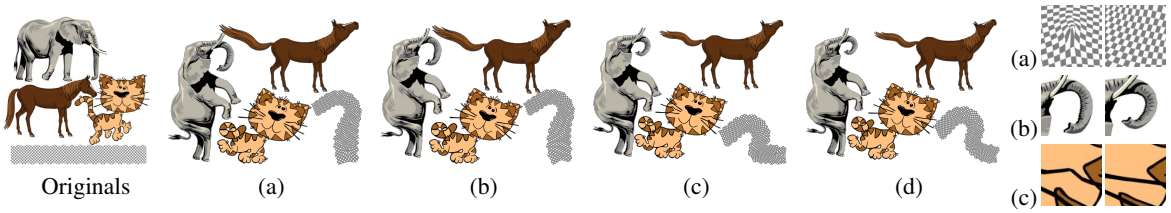


Figure 8: *(a) Deformations produced using linear steps along (3), (b) deformations with linear steps using Dirichlet fields (4), (c) deformations with log spiral steps along (3), (d) deformations produced using both improvements. Deformations in each column were produced using identical constraints at each step recorded from an interactive session using the full implementation. The rightmost column zooms in on typical distortions in output of (a), (b), and (c) and compares with the output of (d).*

There are curves other than logarithmic spirals that could be considered as potential trajectories. No other class, however, is a clear alternative. As discussed earlier, straight-line trajectories induce undesirable artifacts, and even circular trajectories are too rigid to allow for most deviations from isometry. Polynomial curves $\gamma(t) = (\sum_n a_n x^n, \sum_m b_m x^m)$ have high degrees of freedom and cannot reproduce rotations; conic sections can reproduce rotations but have many degrees of freedom and are not easily parameterizable. In general, logarithmic spirals have not only theoretical justification through their connection to linear holomorphic vector fields but also work well in practice, providing adequate trajectories and fast computations using equation (15).

2.4. Summary

The algorithm above can be summarized as follows:

1. User chooses $C = \{v_i\}$
2. User drags points $\{v_i\}$ to $\{\tilde{v}_i\}$
3. Set $\tilde{U}_i = \tilde{v}_i - v_i$
4. Build P and solve (7) and (8) to find $\vec{U}_{opt}$
5. Fit locally linear holomorphic fields to $\vec{U}_{opt}$ using (15)
6. Move vertices by averaging log spirals as in (16)

This process is repeated in real-time for every mouse move.

Overall, we make three main improvements beyond simply iterating small as-rigid-as-possible solves:

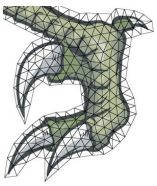
- Replacement of the ARAP nonlinear optimization with a linear vector field solve, justified not only as an approximation to ARAP but also in the language of AKVFs.
- A second vector field solve guaranteeing smoothness of the resulting displacement.
- An efficient method for integrating AKVFs using logarithmic spirals avoiding distortions caused by linear steps.

Figure 8 isolates the effect of each change, showing that all three are necessary for the formulation of an efficient and intuitive deformation method. In general, deformation using fields from the first step (3) rather than (4) creates singularity points around the constraints, while using linear displacements instead of logarithmic spirals can dilate parts if they are rotated during the deformation.

3. Analysis

3.1. Computational Complexity

The slowest step in our algorithm is solving the linear systems (7) and (8). Since our mesh is updated constantly and the expression for P depends on vertex positions, the Cholesky factorization is not constant, and we need to recompute it for each solve. In practice even a non-optimized implementation performs reasonably well, yielding interactive rates (about 30 fps) for models of a few thousand vertices on an Intel core i7 2.13 GHz processor using a single thread; a typical triangulation is shown in the inset figure. Extra speedup might be gained by separating symbolic and numerical factorization and performing the former only once. This optimization is possible since the sparsity structure of both least-squares systems does not change. Finally, with the recent release of sparse Cholesky factorization in NVidia's CUDA language [NVi10], a GPU-based implementation of our method should be possible and potentially more efficient.



3.2. Limitations

Our velocity-based deformation metaphor is not the same as the positional constraint approach used in most deformation methods. Although in our experiments the two seem equally intuitive, further study might be required to validate that this is the case. Similar study also might be needed to verify that soft rather than hard velocity constraints are acceptable, although this relaxation is used by many other approaches and is known to produce much more pleasing output even if the user's target positions or velocities are not matched exactly.

Because we use velocities rather than positions, the final deformation of a shape depends on the paths of the constrained vertices. While experimenting with our system we were able to generate a variety of deformations without noticing path dependence, if the deformation is pushed to be highly non-isometric, stretching and other non-isometric changes may accumulate (see accompanying video for an example); one could argue, however, that this type of path-dependence should occur since the user has expressed a de-

formation that changes the structure of the underlying shape. As with constraint-based methods, using velocities also does not guarantee that the resulting deformation is bijective, although in practice we have observed almost no instances of localized overlapping triangles.

Finally, our method requires a discretization of the domain Ω. Continuous methods only require a discretization of the boundary and thus may be more efficient. It might be possible to rephrase our method to use boundary element methods for the solution instead of finite elements, but we have not investigated this direction yet; such a discretization would be nonlocal in the sense that two points that are close through the interior of Ω but far along the boundary $\partial\Omega$ may interact. It also is worth noting that methods based on BEM have an expensive pre-processing step requiring the inversion of a dense matrix. This matrix typically changes with the choice of constrained vertices, and hence the user is required to commit to a predefined set of handles. Our method, on the other hand, is more flexible, as the user can change the placement of the constrained vertices during the interaction.

3.3. Comparisons

We compare our method to two representatives of recent research in planar deformation. One is a continuous conformal method that allows for point-to-point manipulation [WBCG09], and the other is a discrete iterative as-rigid-as-possible deformation [LZX*08]. The latter is a recent approach to this problem published as a parameterization method that can also be used for deformation. We note that the most recent continuous ARAP methods [BCWG09, CPSS10] are similar in concept to the discrete method in that they also are iterative and minimize a nonlinear energy; for these reasons they have similar drawbacks. We do not compare with non-iterative ARAP methods like [IMH05], which generate comparable if not somewhat less desirable output. The most recent conformal method is [WG10], which requires the user to modify the cage directly and hence is less intuitive to use. Thus, we found the methods [WBCG09] and [LZX*08] to be the most appropriate for comparison and used implementations provided by the authors.

We compare the methods using three models, one synthetic (a rectangular bar with detail on its boundary) and two more complicated ones (artist-drawn cartoons). Since the deformation metaphor is slightly different for the three deformation approaches, we try to achieve the same pose on all three models to facilitate comparison. We compare the results qualitatively in Figure 9 and quantitatively in Table 1, based on distortion measures from [LZX*08, HG99]:

$$E_{area} = \sum_{i=1}^{m} \rho_i \left(\sigma_i^1 \sigma_i^2 + \frac{1}{\sigma_i^1 \sigma_i^2} \right)$$

$$E_{angle} = \sum_{i=1}^{m} \rho_i \left(\frac{\sigma_i^1}{\sigma_i^2} + \frac{\sigma_i^2}{\sigma_i^1} \right)$$

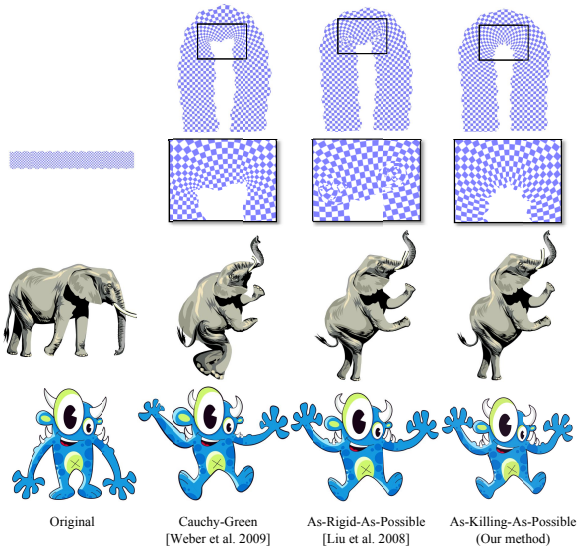


Figure 9: *Comparison of our method with Cauchy-Green and ARAP deformation methods.*

where $\rho_i = A_i / \sum_j A_j$ for triangle area A_i and σ_i^j is the jth singular value of the Jacobian of the map for triangle i; optimally both values equal 2. Our area distortion measure is symmetric, penalizing both shrinkage and expansion.

Model	E_{area}			E_{angle}		
	CG	ARAP	AKAP	CG	ARAP	AKAP
Bar	2.356	2.042	2.070	2.001	2.036	2.062
Elephant	4.605	2.019	2.030	2.151	2.033	2.054
Alien	2.596	2.012	2.016	2.004	2.013	2.017

Table 1: *Distortions for the deformations from Figure 9*

Compared to conformal maps, our method has the advantage that it considerably reduces area distortion at the expense of some non-uniform scaling. It is worth noting that the angular distortion of the conformal method converges to zero as the density of the triangulation grows. Since the "elephant" model has somewhat larger triangles, the angular errors of the conformal method are bigger for this model.

Compared to global ARAP methods that map directly from the starting shape to the final configuration, we have the advantage that taking small steps avoids nonlinear solves that may reach only local minima. Although following the AKVFs for small time steps can be viewed as "incremental ARAP" approximately completing an ARAP Newton iteration per vector field computation, the smoothing step and spiral trajectories prevent distortion that occurs in the version without these improvements. Of course, careful analysis of the cubic difference between the AKAP energy and that in [CPSS10] would be needed to establish the exact relationship, although for sufficiently small steps they clearly coincide. Regardless, the close relationship to KVFs shows that such an incremental method is theoretically justified.

Although our linear solve is more expensive than the iterations in ARAP methods, we have the advantage of allowing the user to change which vertices are pinned as part of the interaction. While experimenting with the system we have found this freedom especially helpful, since it allows us to start with a small number of constraints to achieve the general shape of the deformation and add constraints as we proceed to the more detailed changes. For example, for the "raptor" model from Figure 1, we can move the leg to the required position using a single pin and then fix it and add other pins for deforming the claws.

We also compared our output to that of the Moving Least Squares (MLS) approach in [SMW06, MS10]. MLS works well for deforming full images, but it does not produce as effective output on meshes with complex boundaries because it uses Euclidean distances. For instance, the inset figure shows an MLS deformation of the bar from Figure 9; although the original boundary is almost convex, the output does not appear natural around the bend ($E_{area} = 4.951, E_{angle} = 8.158$).

Figures 1, 10, and the accompanying video show additional deformations generated by our algorithm.

4. Conclusions and Discussion

We have presented a novel almost-isometric deformation method based on as-Killing-as-possible vector fields. The method is non-iterative and produces results comparable to iterative, nonlinear, as-rigid-as-possible methods. Furthermore, we have demonstrated the relationship between logarithmic spirals and linear holomorphic vector fields, motivating their use for shape interpolation applications.

A natural direction for future work would be to generalize our deformation method to deformations of 3D volumes. The KVF energy is well defined in any dimension; however, it is not clear that a generalization to 3D of a logarithmic spiral is a strong choice of trajectory. Furthermore, the computational load of our method is heavier in three-dimensions, and thus a GPU-based implementation could be necessary.

An additional interesting direction would be to investigate the properties of as-Killing-as-possible vector fields. For example, when solving the linear system with Dirichlet boundary conditions, we are effectively finding the vector field of the interior vertices as linear combination of the vectors on the boundary vertices. This might imply that we could define "as-Killing-as-possible" barycentric coordinates. Investigating the properties of such coordinates could reveal additional properties of close-to-isometric deformations.

Acknowledgments

This research was made with government support under and awarded by DoD, Air Force Office of Scientific Research, National Defense Science and Engineering Graduate (NDSEG) Fellowship, 32 CFR 168a. The authors also would like to acknowledge the following grants: the Hertz Foundation Fellowship, the NSF GRF program, the Weizmann Institute WiS award, NSF grant FODAVA 0808515 and IIS 0914833, and a seed grant from the Stanford Computer Science Department.

Appendix A: Reproduction of Global Rigid Motions

Here we provide a short proof that the method for computing the vector field $\vec{U}_{opt}$ will recover rigid motions.

Lemma 1 If the constraints are compatible with a vector field $\vec{W}$ inducing a global rigid motion, then $\vec{U}_{opt} = \vec{W}$.

Proof Consider a triangle with vertices p_i, p_j, and p_k, area A_T and vectors v_i, v_j, and v_k. Using the formula in [BKP*10] for the gradient of a piecewise-linear function on a triangle, a vector field with a Jacobian $J = \left(\begin{smallmatrix} a & b \\ c & d \end{smallmatrix}\right)$ would be in the null space of P exactly when:

$$\begin{aligned}
0 =& 2A_T a = v_{ix}(p_{jy} - p_{ky}) + v_{jx}(p_{ky} - p_{iy}) + v_{kx}(p_{iy} - p_{jy}) \\
0 =& 2A_T(b + c) \\
=& v_{ix}(p_{kx} - p_{jx}) + v_{jx}(p_{ix} - p_{kx}) + v_{kx}(p_{jx} - p_{ix}) \\
&+ v_{iy}(p_{jy} - p_{ky}) + v_{jy}(p_{ky} - p_{iy}) + v_{ky}(p_{iy} - p_{jy}) \\
0 =& 2A_T d = v_{iy}(p_{kx} - p_{jx}) + v_{jy}(p_{ix} - p_{kx}) + v_{ky}(p_{jx} - p_{ix})
\end{aligned}$$

For a single triangle, there are six unknowns (two for v at each of the three vertices), so for points in general position this system defines a three-dimensional linear subspace.

From Equation 1, rigid motion fields can be written $v(x,y) = v_0 + a(-y,x)$ with $v_0 \in \mathbb{R}^2$ and $a \in \mathbb{R}$. Plugging into the right-hand sides above shows that any $v(p)$ is in null(P). Since $v(x,y)$ has three linear parameters (one in a and two in v_0) and the system has co-rank 3, no other vector fields can be in the null space of P.

The argument above shows that if a vector field is in null(P) for an entire mesh, it must induce rigid motions of each of its triangles. We need to show that the choice of a and v_0 is the same for each triangle. Fortunately, it is clear from the expression for $v(x,y)$ that a rigid motion vector field is (over-)determined by its value at two points. So, any two triangles that share an edge must have the same a and v_0, which thus are constant for any path-connected mesh.

Having characterized null(P), we proceed to showing $\vec{W} = \vec{U}_{opt}$. By the above argument about P, since $\vec{W}$ is rigid and satisfies the constraints, it has zero energy in Equation 3 and is the unique such vector field; any other is either non-rigid with positive E_K or does not satisfy the constraints exactly. Thus, $\vec{U}^\circ = \vec{W}$. Since the boundary of the mesh has at least three vertices, the vectors of $\vec{W}$ still induce the same unique rigid motion, so by an identical argument applied to Equation 4, $\vec{U}_{opt} = \vec{U}^\circ = \vec{W}$, as desired. □

References

[ACOL00] ALEXA M., COHEN-OR D., LEVIN D.: As-rigid-as-possible shape interpolation. SIGGRAPH '00, ACM Press, pp. 157–164. 2

[BCBSG10] BEN-CHEN M., BUTSCHER A., SOLOMON J., GUIBAS L. J.: On discrete killing vector fields and patterns on surfaces. *Comp. Graph. Forum 29*, 5 (2010), 1701–1711. 3, 5

[BCWG09] BEN-CHEN M., WEBER O., GOTSMAN C.: Variational harmonic maps for space deformation. *ACM Trans. Graph. 28* (July 2009), 34:1–34:11. 2, 3, 8

Figure 10: *A few deformations generated by our method. The original models are shown on the top left.*

[BKP*10] BOTSCH M., KOBBELT L., PAULY M., ALLIEZ P., LEVY B.: *Polygon Mesh Processing*. AK Peters, 2010. 5, 9

[BS08] BOTSCH M., SORKINE O.: On linear variational surface deformation methods. *IEEE Trans. Vis. Comp. Graph. 14* (2008), 213–230. 2

[CDHR08] CHEN Y., DAVIS T. A., HAGER W. W., RAJAMANICKAM S.: Algorithm 887: CHOLMOD, supernodal sparse cholesky factorization and update/downdate. *ACM Trans. Math. Softw. 35* (October 2008), 22:1–22:14. 5

[Coo14] COOK T. A.: *The Curves of Life*. Dover, 1914. 6

[CPSS10] CHAO I., PINKALL U., SANAN P., SCHRÖDER P.: A simple geometric model for elastic deformations. *ACM Trans. Graph. 29* (July 2010), 38:1–38:6. 2, 3, 4, 8, 9

[DoC76] DOCARMO M.: *Differential Geometry of Curves and Surfaces*. Prentice Hall, 1976.

[Flo03] FLOATER M. S.: Mean value coordinates. *Comp. Aided Geom. Des. 20* (March 2003), 19–27. 2

[GT01] GILBARG D., TRUDINGER N. S.: *Elliptic partial differential equations of second order*. Classics in Mathematics. Springer-Verlag, Berlin, 2001. Reprint of the 1998 edition. 4

[HG99] HORMANN K., GREINER G.: MIPS: An efficient global parameterization method. 153–162. 8

[HS08] HORMANN K., SUKUMAR N.: Maximum entropy coordinates for arbitrary polytopes. In *Proc. SGP* (2008), pp. 1513–1520. 2

[IMH05] IGARASHI T., MOSCOVICH T., HUGHES J. F.: As-rigid-as-possible shape manipulation. *ACM Trans. Graph. 24* (July 2005), 1134–1141. 2, 3, 8

[JBPS11] JACOBSON A., BARAN I., POPOVIĆ J., SORKINE O.: Bounded biharmonic weights for real-time deformation. SIGGRAPH '11 (to appear). 2

[JMD*07] JOSHI P., MEYER M., DEROSE T., GREEN B., SANOCKI T.: Harmonic coordinates for character articulation. *ACM Trans. Graph. 26* (July 2007). 2

[LKCOL07] LIPMAN Y., KOPF J., COHEN-OR D., LEVIN D.: GPU-assisted positive mean value coordinates for mesh deformations. In *Proc. SGP* (2007), pp. 117–123. 2

[LLCO08] LIPMAN Y., LEVIN D., COHEN-OR D.: Green coordinates. *ACM Trans. Graph. 27* (Aug. 2008), 78:1–78:10. 2

[LZX*08] LIU L., ZHANG L., XU Y., GOTSMAN C., GORTLER S. J.: A local/global approach to mesh parameterization. In *Proc. SGP* (2008), pp. 1495–1504. 2, 8

[MS10] MANSON J., SCHAEFER S.: Moving least squares coordinates. *Comp. Graph. Forum 29*, 5 (2010), 1517–1524. 2, 9

[NVi10] NVIDIA CORPORATION: *CUSPARSE Library*. Tech. rep., August 2010. 7

[Pet10] PETERSEN P.: *Riemannian Geometry*. Springer, 2010. 2, 3

[PP93] PINKALL U., POLTHIER K.: Computing discrete minimal surfaces and their conjugates. *Experimental Mathematics 2* (1993), 15–36. 5

[SA07] SORKINE O., ALEXA M.: As-rigid-as-possible surface modeling. In *Proc. SGP* (2007), pp. 109–116. 2, 3

[SBCBG11] SOLOMON J., BEN-CHEN M., BUTSCHER A., GUIBAS L.: Discovery of intrinsic primitives on triangle meshes. *Comp. Graph. Forum 30*, 2 (2011). 3, 5

[SDC09] SÝKORA D., DINGLIANA J., COLLINS S.: As-rigid-as-possible image registration for hand-drawn cartoon animations. In *Proc. NPAR* (2009), NPAR '09, ACM, pp. 25–33. 2

[SMW06] SCHAEFER S., MCPHAIL T., WARREN J.: Image deformation using moving least squares. *ACM Trans. Graph. 25* (July 2006), 533–540. 2, 9

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

[WBCG09] WEBER O., BEN-CHEN M., GOTSMAN C.: Complex barycentric coordinates with applications to planar shape deformation. *Comp. Graph. Forum 28*, 2 (2009), 587–597. 2, 8

[WG10] WEBER O., GOTSMAN C.: Controllable conformal maps for shape deformation and interpolation. *ACM Trans. Graph. 29* (July 2010), 78:1–78:11. 2, 8

[WNS*10] WHITED B., NORIS G., SIMMONS M., SUMNER R. W., GROSS M. H., ROSSIGNAC J.: BetweenIT: An interactive tool for tight inbetweening. *Comp. Graph. Forum 29*, 2 (2010), 605–614. 6

[WXW*06] WENG Y., XU W., WU Y., ZHOU K., GUO B.: 2d shape deformation using nonlinear least squares optimization. *Vis. Comp. 22* (September 2006), 653–660. 2