Eurographics Symposium on Geometry Processing 2009
Marc Alexa and Michael Kazhdan
(Guest Editors)

Volume 28 (2009), Number 5

Energy-Based Image Deformation

Z. Karni [†1] D. Freedman[‡1] C. Gotsman[§2]

[1]Hewlett-Packard Laboratories, Haifa, Israel
[2]Computer Science Dept, Technion – Israel Institute of Technology, Haifa, Israel

Abstract

We present a general approach to shape deformation based on energy minimization, and applications of this approach to the problems of image resizing and 2D shape deformation. Our deformation energy generalizes that found in the prior art, while still admitting an efficient algorithm for its optimization. The key advantage of our energy function is the flexibility with which the set of "legal transformations" may be expressed; these transformations are the ones which are not *considered to be distorting. This flexibility allows us to pose the problems of image resizing and 2D shape deformation in a natural way and generate minimally distorted results. It also allows us to strongly reduce undesirable foldovers or self-intersections. Results of both algorithms demonstrate the effectiveness of our approach.*

Categories and Subject Descriptors (according to ACM CCS): I.3.3 [Computer Graphics]: Picture/Image Generation—Line and curve generation

1. Introduction

We consider the problem of computing a natural shape deformation, where the "naturalness" is guided not only in terms of pure geometric considerations, but also by external data, such as an image in which the shape lives. To illustrate, consider the following scenario: we are given an image and an object segmented from the image. The user interface allows us to adjust the positions of some "control points" within the object; given this user input, we would like to compute a natural deformation of the object which conforms to the new positions of the control points (see Fig. 1 (right)). Clearly, the image content should play a role in the deformation. Where the image is highly textured, we would expect a fairly minor geometric deformation, namely, something which is close to isometric; whereas in homogeneous regions, where it is less noticeable, we might be willing to tolerate a greater amount of deformation.

In this paper, we present a framework to approach shape deformation problems with external or image-based considerations. The framework consists of two components: a deformation energy, and an algorithm for minimizing this energy. The deformation energy captures the image-based considerations through *sets of allowed transformations*, which specify – for each point in the shape – which types of deformations are deemed acceptable. The deformation energy then measures the quality of a given deformation by computing, for each point, the deviation of the deformation from the set of allowed transformations, and summing over all points. It turns out that this energy can be optimized using a simple alternating least-squares algorithm of the type used in [SA07], *inter alia*, sometimes called a "local-global" algorithm. The latter name derives from the fact that the algorithm consists of two stages per iteration, where the first stage involves local computations on the individual elements of the domain discretization (usually triangles or quads), and the second stage involves the solution of a global linear system.

One important aspect of our approach is its ability to incorporate the external or image-based considerations into the *local step* of the algorithm. As we shall show, it is actually more natural to define the deformation energy in such a way that these considerations affect the global step. However, this leads to two key drawbacks. First, it generates deformations which do not depend strongly on their local image-based terms; this is because the global step is essentially a Poisson-type system, with a corresponding averaging operation, so that the local effects of image-based terms are averaged out. Second, it tends to cause *foldovers* (or self-intersections) of the underlying mesh, a critical numerical problem which produces unacceptable artifacts in the results, for which existing works have proposed only heuristic remedies, e.g. [WTSL08]. By contrast, when the image-based considerations are incorporated directly into the local step, both of these problems are strongly mitigated.

A second important aspect of the approach is its flexibility, hence its applicability to a variety of applications. We present two such applications: image resizing, and image-sensitive object deformation. In image resizing (see Fig. 1 left), the objective is to modify the aspect ratio of an image in as natural a way as possible, i.e. preserve, as much as possible, the correct (i.e. original) aspect ratio for the "important" sections of the image. This problem can be cast in our framework, and we show that the resulting algorithm im-

[†] e-mail:zachi.karni@hp.com
[‡] e-mail:daniel.freedman@hp.com
[§] e-mail:gotsman@cs.technion.ac.il

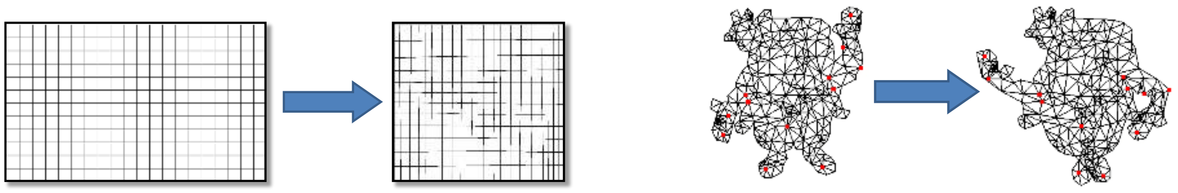

Figure 1: *Left: Image resizing. Right: 2D shape defomation; the user constraints are shown as red points.*

proves upon the results of the state of the art, [WTSL08] and [AS07]. In image-sensitive shape deformation (henceforth, simply "shape deformation" - see Fig. 1 right), the goal is to deform a shape within an image in a natural way – i.e., by taking into account the image details – subject to user input in the form of control points that the user has moved. (This is the application sketched in the first paragraph.) This problem may also be cast within our framework, and the results perform well against the methods of [SMW06] and [IMH05].

1.1. Prior Work

Shape deformation is an important subfield of computer graphics, and its 3D version is useful for modeling and posing 3D shapes, particularly in animation systems. The interested reader is referred to [SA07] for an extensive survey of 3D methods. Methods for 2D shape manipulation, in particular for applications of image deformation and manipulation, have been developed in parallel. An early example is the work of [AG99], who show how traditional image processing operations, such as sharpening, may be obtained by a warp of the image domain, without explicitly modifying the image content per se. This contrasts with the classical "signal processing" approaches, where the signal undergoes any number of modifications. More recent examples include work on 2D shape deformation [IMH05, SMW06, GSCO06, ESA07, FH07, WXXC08, WBCG09], as well as image resizing [AS07, RSA08, WGCO07, WTSL08]. We will relate the more relevant prior art to ours in the body of the paper, particularly in the sections dealing with experimental results.

1.2. Summary of Contributions

1. **Flexibility and Generality of the Deformation Framework:**
   a. **Image-Based Terms:** The framework allows for the incorporation of external considerations, such as the saliency of the image, within the deformation computation.
   b. **Generalization of Prior Approaches:** The framework generalizes previous approaches, including [SA07, LZX*08]. In particular, the deformation energy admits minimization via a **local-global** optimization technique.
2. **Numerical Effectiveness of the Deformation Computation:**
   a. **Local Step:** The image-based effects are incorporated into the local step, rather than the global step. This has the following consequences:
   b. **Controllable Deformations:** The deformation locally depends much more finely on the local image terms when these terms enter the computation through the local step. If the terms enter through the global step, their effect is much more diffuse.
   c. **Foldover Avoidance:** The computation does not produce foldovers in the relevant meshes. This is important, as avoidance of foldovers is difficult for many algorithms.
3. **Successful Applications of the Framework**
   a. **Image resizing:** Results improve upon the state of the art, [WTSL08] and [AS07].
   b. **Image-Sensitive Object Deformation:** Results perform well against the methods of [SMW06] and [IMH05].

2. The Deformation Energy and its Minimization

2.1. The Deformation Energy

We begin by considering an energy function defined on deformations of the Euclidean space $\mathbb{R}^d$. A *deformation* of $\mathbb{R}^d$ is simply a C^1 function $f : \mathbb{R}^d \to \mathbb{R}^d$. To define the deformation energy $E[f]$, we require a collection of *sets of allowed transformations* which act on $\mathbb{R}^d$, $\{G_x\}$, one set for each $x \in \mathbb{R}^d$. The set G_x contains the actions on the space which we do *not* consider distorting at the point x. For example, we might wish to compute a deformation which is close to locally isometric, in which case G_x would be the set of rigid transformations. However, G_x can also incorporate external or image-based considerations, which will be the approach we take here. For example, at a point x where the image is homogeneous, G_x might be a very large class of transformations, thus imposing very little restriction on the deformation at that point; whereas if the image at x is highly detailed, we might wish for G_x to be closer to the set of rigid transformations.

The energy of a deformation f is defined as

$$E[f] = \int_{x\in\mathbb{R}^d} \left\{ \min_{g(x)\in G_x} \left\| J_f(x) - g(x) \right\|_F^2 \right\} dx \qquad (1)$$

where $J_f(x)$ is the Jacobian of f at x, and $\|\cdot\|_F$ is the Frobenius norm of a matrix. Note the slight abuse of notation: $g(x)$ is a matrix representation of the linear transformation $g(x)$, rather than the transformation itself. (Note also that the above functional is well-defined only for Euclidean space deformations; although it is possible to derive an energy for more general deformations on manifolds, such a derivation is beyond the scope of this paper.)

The term $\|J_f(x) - g(x)\|_F^2$ captures the idea that locally, the energy measures the deviation of the transformation f from the action of an element of G_x. Thus, for example, if G_x is the set of rotations $SO(n)$, then E measures the deviation of the deformation f from a (locally) isometric deformation. On the other hand, if $d = 2$ and G_x is the set of similarity transformations, then E measures how close the deformation f is to being conformal. In our applications, G_x will contain image-based information, thus also depend strongly on x, in the manner alluded to above.

In summary, then, the deformation energy $E[f]$ given in (1) is quite general. In fact, it is not hard to show that it is the generalization of a number of current approaches, such as the "As-Rigid-As-Possible" (ARAP) technique of [SA07] (a simple version of this for 2D was first proposed in [IMH05]), where $G_x = SO(2)$ for all x; and of the 3D mesh parameterization technique of [LZX*08] for 3D triangle meshes, where G_x is either the set of rigid or similarity transformations.

2.2. Minimization of the Energy: the Local-Global Algorithm

Our goal is to minimize the deformation energy given some user-defined constraints expressing what the user wishes the deformation to achieve. Typically, the user wishes to map some "control points" in the space $\mathbb{R}^d$ to new locations, i.e., for some fixed x_i, $i = 1, \dots, k$,

$$f(x_i) = x_i^0$$

In this case, we would like to solve:

$$\min_f E[f] \quad \text{subject to} \quad f(x_i) = x_i^0, i = 1, \dots, k \tag{2}$$

Sorkine and Alexa [SA07] proposed an alternating least-squares algorithm, the so-called "local-global" algorithm, for solving an instance (the "As-Rigid-As-Possible" deformation) of this problem. In their algorithm, which operates on a triangular discretization of a manifold 3D surface, the algorithm alternates between a "local" stage, where a small optimization problem is solved per triangle, and a "global" stage, where a large linear system is solved for the entire set of triangles. Variants of this algorithm were also proposed by Igarashi et al. [IMH05], who proposed to apply just one iteration, and Wang et al. [WXXC08], who used a simple heuristic global step. We now show that an extension of this technique is possible in the more general setting.

Denote (overloading the E symbol of (1)):

$$E[f, g] = \int \|J_f(x) - g(x)\|_F^2 \, dx \tag{3}$$

and the constraints as

$$f(\cdot) \in \Gamma$$

Then the optimization problem (2) can be rewritten as

$$\min_{f,g} E[f, g] \quad \text{subject to} \quad f(\cdot) \in \Gamma \tag{4}$$

where in the end, we are actually only interested in f. Our algorithm for solving this optimization is given by the iterative scheme in Figure 2. Note that the initialization stage is application specific. This will be discussed in Sections 3.4 and 4.4.

```
Initialize  Initialize $f = f_0$.
Local Step  Given $f_{t-1}$, minimize $E$ with respect to $g$:
        $g_t = \arg\min_g E[f_{t-1}, g]$
Global Step  Given $g_t$, minimize $E$ with respect to $f$:
        $f_t = \arg\min_f E[f, g_t] \quad \text{subject to} \quad f(\cdot) \in \Gamma$
Convergence  If convergence has been reached (up to some
    tolerance), end. Otherwise, go to the Local Step.
```

Figure 2: *The alternating least squares "local-global" algorithm.*

Note that there are additional constraints on f and g in (4), thus also on f_t and g_t in the iterations: f should be C^1, and, locally, g should behave at x like some $g(x) \in G_x$.

While seemingly obvious, the correctness of this algorithm requires proof (see Appendix for proofs of all theorems in this paper).

Theorem 1 The algorithm in Figure 2 converges to a local minimum of E. ■

Having established our framework for computing deformations, we now move on to two applications of this framework, image resizing and image-sensitive shape deformation. In both cases, the key st andep will be proper definition of the sets of allowed transformations, $\{G_x\}$. Given the definitions, we can derive explicit versions of the local and global steps of the algorithm for each case. On the way, we will comment on the fact that the image-based information is incorporated in the local step, as well as what that entails: more controllable deformations without foldovers.

3. Image Resizing

The problem of image resizing may be posed as follows. Given an image, the user is allowed to move the bottom right corner of the image from its original position to a desired location, while the top left corner is fixed. This essentially rescales the x- and y-axes, possibly changing the image aspect ratio. This type of operation is already common in photographic printing on different paper sizes, and in the conversion of video content from one format to another (e.g. standard 4:3 to HD 16:9). It will become more common as improvements in printing technologies encourage customized printing, which in turn will require the embedding of given image content into different template sizes, on demand. The simplest uniform resizing will stretch all parts of the image equally. Standard algorithms for converting 4:3 format to 16:9 format stretch the image non-uniformly: the stretch is minimal in the center of the image, where it is more noticeable by the viewer, and gradually increases towards the periphery. However, it may be desirable to resize different parts of the image differently, depending on the image content. For example, where there is interesting detail, we may not wish to deform the image too much; namely, we will desire a close to *isotropic* scaling. However, where there is not much in the way of interesting detail, we will not mind stretching. Indeed, the stretching in these regions may be even greater than that implied by the original user constraint.

In the following four sections, we define a natural deformation energy, describe the deformation algorithm using a discretization based on a quadrangular mesh, discuss implementation issues, and show results.

3.1. The Deformation Energy

3.1.1. Measure of Image Detail

We wish to adapt the energy of Eq. (1) to the problem of image resizing. To capture our preference for less distortion where there is more image detail, we use the following measure of image detail:

$$\lambda(x) = K_\lambda \|\nabla I(x)\|,$$

i.e., the magnitude of the image gradient, where we choose K_λ so that $0 \leq \lambda(x) \leq 1$. Previous papers [WTSL08] have used more complex measures, such as the saliency measure of [IKN98], but we have found that this simple measure works just as well. Extensions could easily include more complicated notions of saliency or level of detail.

3.1.2. A Natural, but Problematic, Formulation

Given this notion of image detail, $\lambda(x)$, it is quite natural to incorporate it as a weight in the integrand of the deformation energy, i.e., redefine E as

$$E[f,g] = \int \lambda(x) \left\|J_f(x) - g(x)\right\|_F^2 dx$$

Solving Problem (4) is now a simple weighted linear least-squares problem, whose solution may be efficiently obtained by solving a linear system of normal equations. We do not, however, follow this route. The reason is that incorporating the image detail in such a way will imply that its only effect is in the global step, as we shall see in Sections 3.2 and 3.3. This, in turn, leads to very little local control based on varying $\lambda(x)$; instead, the effects of the image detail are averaged out, due to the Poisson nature of the global step. Additionally, nothing prevents the solution from containing unacceptable "foldover" artifacts, where the mapping is not injective.

3.1.3. Our Formulation

Rather than following the above line of logic, we incorporate $\lambda(x)$ in G_x, the set of allowed transformations. This means that the optimization will now be more complex, i.e. the full iterative "local-global" algorithm of Fig. 2, because now a local step is required to accommodate it.

The relevant space is the plane, $\mathbb{R}^2$. Assume that the user chooses to stretch the x-axis more than the y-axis; the reverse situation can be treated analogously, by switching the roles of the variables. Suppose that the x-axis is stretched by a factor of s_x, and the y-axis by a factor of s_y; then if we define $r = s_x/s_y$, our assumption is equivalent to the condition that $r \geq 1$. Now, the sets of allowed transformations G_x consist of axis-aligned affine transformations which can be anything from isotropic scaling up to γr more scaling in the x-direction than in the y-direction, where γ is a small number larger than 1. Typically, γ will be chosen in the range of $1.5 - 2.5$.

More specifically, for a given real number $\rho \geq 1$, we define the set of axis-aligned affine transformations $G_{axis}(\rho)$ by

$$G_{axis}(\rho) = \left\{ g = \begin{bmatrix} a & 0 \\ 0 & b \end{bmatrix} : a,b \geq 0,\ 1 \leq \frac{a}{b} \leq \rho \right\}$$

Given $\lambda(x)$, we wish to allow ρ to vary with λ. In particular, we let $\rho = \psi(\lambda)$, where ψ satisfies

$$\psi(0) = \gamma r, \quad \psi(\infty) = 1, \quad \psi'(\cdot) < 0$$

This captures the idea that where there is no interesting detail ($\lambda = 0$), we allow a stretch of γr, that is, effectively γ times more than uniform resizing would prescribe. By contrast, if there is a significant detail ($\lambda \to \infty$), we try to eliminate stretching altogether. For example, we may use

$$\psi(\lambda) = \frac{\beta\lambda + \gamma r}{\beta\lambda + 1}$$

where β is a free parameter which controls the rate of decrease of ψ. Finally, we define the sets of allowed transformations by

$$G_x = G_{axis}(\psi(\lambda(x)))$$

3.2. The Local Step

Although the formulation of the energy in Eq. (1) is continuous, we now adapt it to the relevant discrete structure for the problem of image resizing: the quadrangle mesh, which is initially a grid of axis-aligned squares. For this, we relax our definition of f to $f \in C^0$ (but C^1 within each quad). If we assume that the deformation f is affine on a given quad, then the Jacobian is constant, and it is straightforward to show that

$$\int_{x \in q} \|J_f(x) - g(x)\|_F^2 dx \propto \sum_{i \in q} \|(f(x_{i+1}) - f(x_i)) - g_q(x_{i+1} - x_i)\|^2$$

where q refers to the quad in question, i is a vertex in that quad, $i+1$ is the next vertex over in the (say) clockwise direction, and g_q is the deformation of the quad q, taken from $G_{axis}(\psi_q)$. Therefore, the energy E can be rewritten as

$$E = \sum_q \sum_{i \in q} \|(f_{i+1} - f_i) - g_q(x_{i+1} - x_i)\|^2 \tag{5}$$

where $\psi_q = \psi(\lambda_q)$ where λ_q is the salience of q, and we have replaced $f(x_i)$ with the shorthand f_i. Note that this holds under the assumption that f is affine on a given quad. In what follows, we drop this assumption, and let each of the vertices of a given quad be set independently.

The local step is therefore

$$\min_{g_q \in G_{axis}(\psi_q)} \sum_{i \in q} \|(f_{i+1} - f_i) - g_q(x_{i+1} - x_i)\|^2$$

Note that we have dropped the t index, implying that f and x should be taken from the previous iteration.

This can be rewritten as

$$\min_{a_q, b_q} \left\{ \sum_{i \in q} \left[(f^1_{i+1} - f^1_i) - a_q(x^1_{i+1} - x^1_i) \right]^2 + \left[(f^2_{i+1} - f^2_i) - b_q(x^2_{i+1} - x^2_i) \right]^2 \right\}$$

$$\text{subject to:} \quad 1 \le a_q / b_q \le \psi_q, \quad a_q, b_q \ge 0$$

where x^j_i indicates the j^{th} coordinate of x_i. This is a quadratic optimization problem with linear inequality constraints.

Note that one of the constraints in the above formulation is $a_q, b_q \ge 0$, that is, that the resizing is non-negative in both directions. This seemingly innocuous constraint is what strongly mitigates the problem of foldovers. Several techniques, such as [WTSL08], try to prevent foldovers in a heuristic fashion. By contrast, we explicitly constrain the local step to produce a legal (i.e. not folded over) transformation. It is still possible that a foldover could enter through the global step; however, such foldovers are never observed in practice.

Returning to the optimization problem, note that technically this problem is a quadratic program. However, due to the small number of optimizing variables (two), we are able to solve the problem explicitly, as follows. First, note that without the constraints, the solution is

$$\hat{a}_q = E^1_q / D^1_q, \quad \hat{b}_q = E^2_q / D^2_q$$

where

$$D^j_q = \sum_{i \in q} (x^j_{i+1} - x^j_i)^2, \quad E^j_q = \sum_{i \in q} (x^j_{i+1} - x^j_i)(f^j_{i+1} - f^j_i)$$

We can now use the solution to the unconstrained system to obtain the constrained solution to the local step in a "local-global" iteration.

Theorem 2 The local step is as follows. Given the solution to the unconstrained problem $(\hat{a}_q, \hat{b}_q)$, define

$$\tilde{a}_q = \frac{E^1_q + E^2_q}{D^1_q + D^2_q}, \quad \tilde{b}_q = \tilde{a}_q$$

and

$$\breve{a}_q = \frac{\psi^2_q E^1_q + \psi_q E^2_q}{\psi^2_q D^1_q + D^2_q + \lambda_q(\psi^2_q - 1)}, \quad \breve{b}_q = \frac{\breve{a}_q}{\psi_q}$$

Also let $\Gamma(a,b) = D^1_q a - 2E^1_q a + D^2_q b - 2E^2_q b$. Then the solution (a^*_q, b^*_q) to the constrained problem is as follows:

1. If $\hat{b}_q \le \hat{a}_q \le \psi_q \hat{b}_q$ and $\hat{a}_q, \hat{b}_q \ge 0$, then $(a^*_q, b^*_q) = (\hat{a}_q, \hat{b}_q)$.
2. Otherwise:
   a. If $\tilde{a}_q, \tilde{b}_q > 0$ and either (i) $\breve{a}_q < 0$ or $\breve{b}_q < 0$; or (ii) $\Gamma(\tilde{a}_q, \tilde{b}_q) \le \Gamma(\breve{a}_q, \breve{b}_q)$, then $(a^*_q, b^*_q) = (\tilde{a}_q, \tilde{b}_q)$.
   b. If $\breve{a}_q, \breve{b}_q > 0$ and and either (i) $\tilde{a}_q < 0$ or $\tilde{b}_q < 0$; or $\Gamma(\breve{a}_q, \breve{b}_q) < \Gamma(\tilde{a}_q, \tilde{b}_q)$, then $(a^*_q, b^*_q) = (\breve{a}_q, \breve{b}_q)$.
   c. Otherwise, $(a^*_q, b^*_q) = (0, 0)$.

■

3.3. The Global Step

The global step is quite straightforward. Taking the derivative of (5) with respect to f_i gives the normal equations:

$$\sum_{j \in \mathcal{N}(i)} \left[(f_i - f_j) - g_{q(i,j)}(x_i - x_j) \right] = 0$$

where $\mathcal{N}(i)$ is the set of vertex neighbours of i in the quad mesh, and $q(i,j)$ is the quad containing the half-edge (i,j). This is a Poisson-type system of linear equations in the f_i.

Note, of course, that this system of equations must be solved subject to the constraints, which in this case fix the upper left and lower right vertices, as well as one coordinate of each of the vertices along the boundaries of the image (x-coordinate of the left and right boundaries, y-coordinate of the top and bottom boundaries). In this case, the equations corresponding to the coordinates in question are replaced by the constraints. Note also that the Poisson system is sparse, and any change in g or the user-specified constraints changes only the right-hand side of the equation. Thus the system is amenable to efficient sparse linear solvers, and the fixed system matrix may be prefactored, so that simple and very efficient back-substitution may be used in subsequent solves.

3.4. Implementation and Results

We begin by noting the technique we use to initialize the local-global algorithm. In this case, there is a natural choice, uniform resizing, which we use in all of the experiments.

Our first experiment justifies the incorporation of the image information in the local step, which led to a more complicated iterative algorithm (compared to a simple linear system when the image information is incorporated in the global step). Figure 3 shows the result when an image is resized by a factor of 2.4 along the horizontal axis. Using the image information in the global solution results in foldovers, which after rendering lead to a highly cropped image. With our more sophisticated algorithm, such foldovers do not occur.

We compared our deformation-based image resizing method with the two latest advances in this field: the "seam-carving" (SC) method of [AS07] (recently improved by [RSA08]) which removes the less significant top-down and left-right seams from the image, and the "optimized scale-and-stretch" (OSS) method of [WTSL08] which operates under principles similar to ours. The latter, however, uses quite different energy terms and minimization algorithm, which may generate significantly different results. A comparison of the algorithms on a number of images appears in Figure 4. The OSS results were obtained using software kindly provided by the authors, and the SC results from the

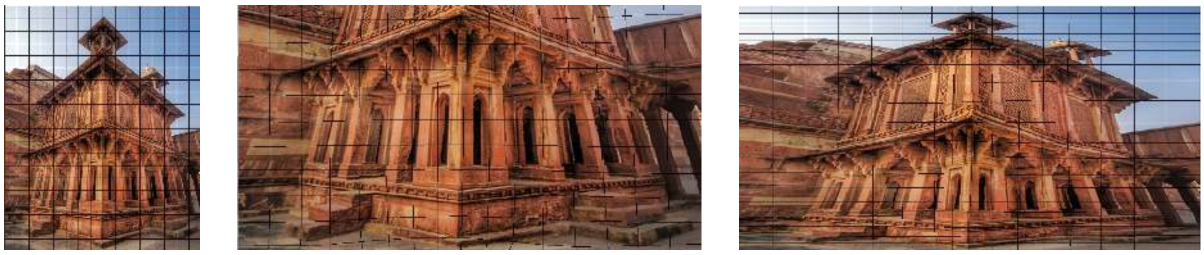

Figure 3: *Incorporation of image information when resizing. Left: Original image. Middle: Image resized using global incorporation, which leads to foldovers (rendered as crops). Right: Image resized using local incorporation.*

current implementation in Adobe Photoshop CS4 Extended (version 11.0.1).

There are a number of important differences between the SC family of algorithms and those based on discrete resizing. First and foremost, SC may be viewed as a "0/1" algorithm. When "shrinking" an image, it either completely removes pixels from the image or completely leaves them in the image. This extreme removal policy implies some information loss. When "expanding" an image, it similarly creates new pixels, introducing some redundancy. In contrast, the family of algorithms based on a more continuous type of optimization, such as ours and OSS, never removes pixels completely or creates completely new ones; all pixels in the output are some filtered version of pixels in the input.

A second significant difference between the two families of algorithms is the somewhat unnatural behavior of SC in scenarios where the expected results seem obvious. The most striking difference is in the case where the user specifies uniform scaling in both dimensions. The natural result to expect would indeed be uniform scaling (as this introduces no distortion at all), and this is exactly what OSS and our method do. SC, in contrast, produces quite different results. This manifests also in the following scenario: Changing the aspect ratio of an image, e.g. by shortening its horizontal dimension, in two different ways: 1) The "direct" way - shortening its horizontal dimension. 2) The "indirect" way - lengthening its vertical dimension and then uniformly downscaling the result. While equivalent in OSS and our methods, SC produces dramatically different results for the two variants. In our comparisons, we have run both the direct and indirect versions of SC, although we show results of only the more natural direct SC, which seems to typically produce better results.

Note that the original image for each example in Figure 4 has been uniformly rescaled so that it will fit in the table of images; however, this step is *not* a part of any preprocessing done by any of the algorithms, and is merely for display purposes. The first two rows of Figure 4 show the results of all the algorithms on two images used by [WTSL08], when significantly shrinking along the horizontal dimension. (We were able to reproduce exactly the OSS results shown by [WTSL08] using the software supplied by the authors.) The shortcomings of SC are obvious, for example the breaking of straight line features in the "car" image and the breaking of the left side of the building and loss of some of the legs of the girl wearing black pants in the "girls" image.

Our results are somewhat similar to OSS in the top two rows of Figure 4. However, as the next four rows show, there are images for which we generate significantly better results. The input image of the third row, depicting a shoreline with sky, is taken from [AS07]; when significantly shrunk along the horizontal dimension, OSS disproportionately increases the sky region. In the "dolphin" example of the fourth row (input image also taken from [AS07]), the image is stretched along the horizontal dimension to double its original size; in this case, OSS shrinks the top portion of the image, including some of the dolphin, in an extreme manner, effectively truncating it. (This may also be due to a foldover problem). The fifth row shows the "camera" image, in which OSS introduces a rounding out of the desk underneath the camera, as well as leaving the camera too small compared to the rest of the image. In this case, SC also leads to a distortion of the contour of the desk, as well as an inexplicable shortening of the lens barrel of the camera. In the "Garfield" example of the sixth row, in which the input image has been shrunk along the horizontal dimension to half of its original size, OSS prefers to strictly preserve the aspect ratio of the cartoons, shrinking them almost uniformly to half their original size, creating a large amount of white space in the result, somewhat defeating the purpose of the application.

In our implementation we partition each image into a grid of identical squares of between 15-20 pixels in both dimensions. The iterative algorithm terminated when all grid points did not move by more than half a pixel between iterations. To gauge the speed of the algorithm, we ran – in addition to the experiments below – 50 further resizings. We found that the average number of iterations until convergence was 7.4, with a maximum of 12.

4. 2D Shape Deformation

In this application, we are interested in the following problem, typically within an image. A segment of an image has been selected, either manually or using a semi-automatic tool such as GrabCut [RKB04]. The user wishes to deform this segment (usually for animation keyframing) by manually moving a small number of "control points" within the segment – typically points on the boundary, though not necessarily. We wish to compute a "good" deformation of the segment by minimizing an appropriate deformation energy. As in the image resizing scenario, the energy should also take into account the fact that we would prefer to introduce

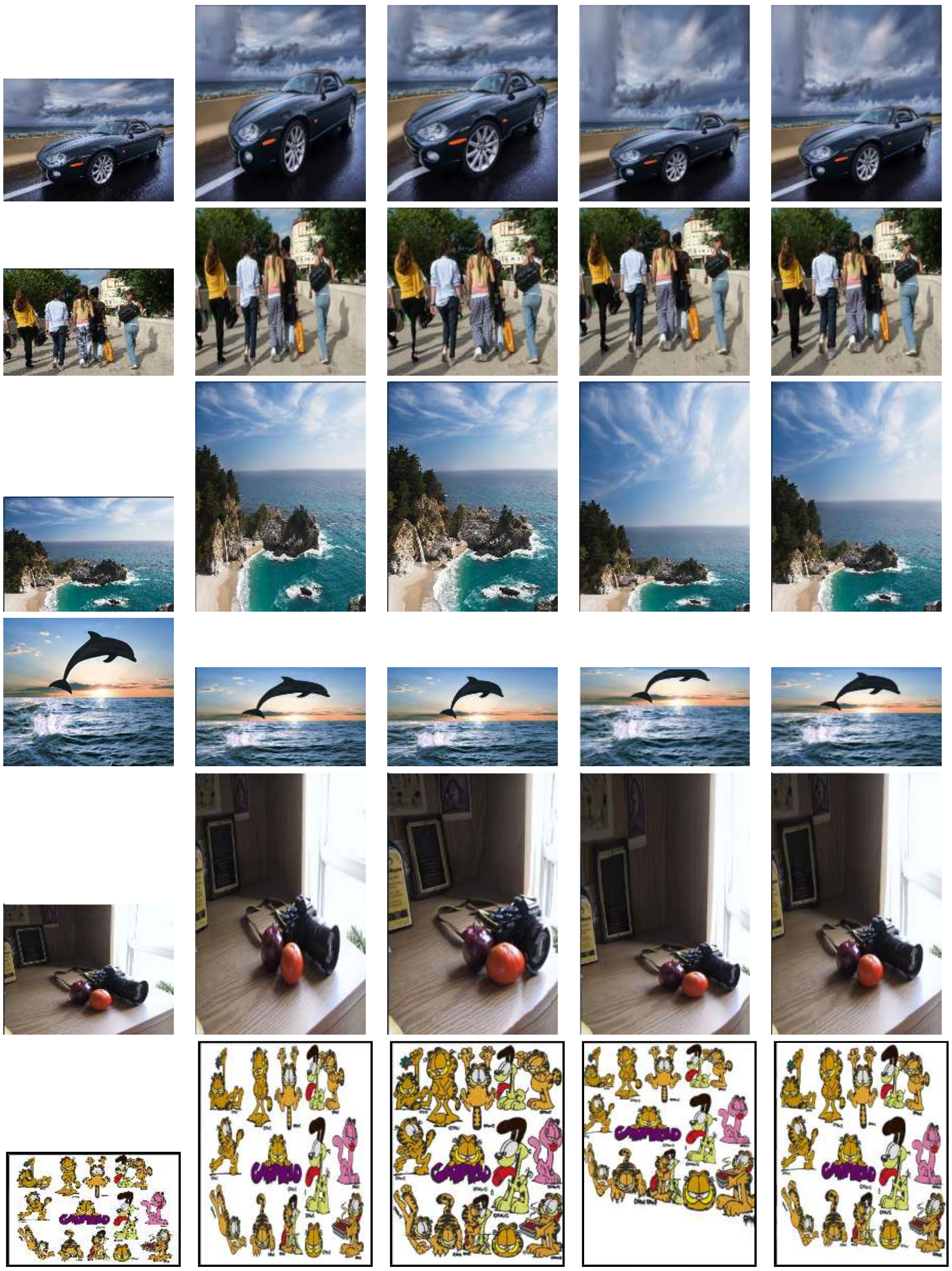

Figure 4: *Comparison of image resizing algorithms.*

less distortion where the image segment has more detail. In relatively flat regions of the segment, we are willing to tolerate more deformation, as this will typically go unnoticed.

We begin by defining a deformation energy which is a special case of (1). We then derive the local and global steps for this energy, the latter using a discretization based on a triangle mesh. Finally, we discuss implementation issues and show results.

4.1. The Deformation Energy

As in the case of image resizing, we measure the level of image detail by the image gradient, $\lambda(x) = K_\lambda \|\nabla I(x)\|$. In order to define the set of allowed transformations, $\{G_x\}$, consider the following. Given a triangular region of the image segment, a good deformation will just rotate and translate the triangle. This rigid transformation will introduce no distortion into that region of the image, since all proportions are preserved. Some distortion will be introduced if the triangle undergoes a similarity transformation, as then the triangle changes its relative size (but not aspect ratio). The most distortion is introduced when the triangle undergoes an affine transformation to another arbitrary triangle. This is the ranking of linear transformations that our energy function will capture.

Specifically, we choose two thresholds, λ_s and λ_r, with $0 < \lambda_s < \lambda_r < 1$. In the region $\lambda(x) \in [0, \lambda_s]$, the set of allowed transformations G_x moves smoothly from any possible affine transformation ($\lambda(x) = 0$) to the set of similarity transformations ($\lambda(x) = \lambda_s$). In the region $\lambda(x) \in [\lambda_s, \lambda_r]$, the set of allowed transformations G_x moves smoothly from the set of similarity transformations ($\lambda(x) = \lambda_s$) to the set of rigid transformations ($\lambda(x) = \lambda_r$). And finally, if $\lambda(x) \geq \lambda_r$, the set of allowed transformations G_x is the set of rigid transformations.

We capture these ideas formally as follows. If $\lambda(x) \in [0, \lambda_s]$, then let $\alpha(x) = \frac{\lambda(x)}{\lambda_s}$, so that $0 \leq \alpha(x) \leq 1$. We define

$$G_{as}(\alpha) = \left\{ g = USV^T : U, V \in SO(2);\ S = \begin{bmatrix} \sigma_1 & 0 \\ 0 & \sigma_2 \end{bmatrix}, \right.$$
$$\left. \ell(\alpha) \leq \frac{\sigma_1}{\sigma_2} \leq u(\alpha),\ \sigma_i \geq 0 \right\}$$

where the bounds $\ell(\cdot)$ and $u(\cdot)$ must satisfy:

$$\begin{array}{lll} \ell(0) = 0, & \ell(1) = 1, & \ell'(\cdot) > 0 \\ u(0) = \infty, & u(1) = 1, & u'(\cdot) < 0 \end{array}$$

Let us elaborate a bit on the structure of $G_{as}(\alpha)$. The transformation elements are represented in their SVD format, and we constrain the *ratio* of singular values σ_1/σ_2 to lie between $\ell(\alpha)$ and $u(\alpha)$. When $\alpha = 0$, the ratio can take on any value, and so $G_{as}(0)$ is simply the set of 2×2 matrices, representing all possible affine transformations. When $\alpha = 1$, the ratio must be exactly 1, thus the singular values must be identical; hence, $G_{as}(1)$ is the set of all similarity transformations. For α's in between 0 and 1, we interpolate smoothly between these two cases. In practice, we choose $\ell(\alpha) = \alpha^\eta$ and $u(\alpha) = \alpha^{-\eta}$ for a positive η, usually taking $\eta = 1$.

We deal with the case $\lambda(x) \in (\lambda_s, 1]$ in a similar fashion. Now, we define:

$$\beta(x) = \begin{cases} \frac{\lambda(x) - \lambda_s}{\lambda_r - \lambda_s} & \text{if } \lambda_s < \lambda(x) \leq \lambda_r \\ 1 & \text{if } \lambda(x) > \lambda_r \end{cases}$$

and

$$G_{sr}(\beta) = \{ g = \sigma R : R \in SO(2); \ell(\beta) \leq \sigma \leq u(\beta) \}$$

where $\ell(\cdot)$ and $u(\cdot)$ are as above. For the case $\beta = 0$, we allow σ to take on any (positive) value, spanning the set of similarity transformations; hence, as desired, $G_{as}(1) = G_{sr}(0)$. For the case $\beta = 1$, σ is constrained to be exactly 1, spanning only the set of rigid transformations. As β moves from 0 to 1, the set of transformations moves smoothly between these two extremes.

Finally, to summarize our construction:

$$G_x = \begin{cases} G_{as}(\alpha(x)) & \text{if } 0 \leq \lambda(x) \leq \lambda_s \\ G_{sr}(\beta(x)) & \text{if } \lambda_s < \lambda(x) \leq 1 \end{cases}$$

Typically, we will choose $\lambda_s = 0.33$ and $\lambda_r = 0.75$.

4.2. The Local Step

From (3), the local step at x computes the function $g(x)$:

$$\arg \min_{g(x) \in G_x} \left\| J_f(x) - g(x) \right\|_F^2$$

Depending on the value of $\lambda(x)$, we can substitute either $G_{as}(\alpha(x))$ or $G_{sr}(\beta(x))$ for G_x. We will address each of these cases in turn.

If $\lambda(x) \leq \lambda_s$, we would like to solve

$$\min_{g \in G_{as}(\alpha(x))} \|J - g\|_F^2 \tag{6}$$

where we have written the Jacobian simply as J. Due to the constraints on σ_1 and σ_2, this is a relatively involved problem; we simply state the solution here, with the proof deferred to the Appendix. Let the singular value decomposition of J be $J = USV^T$, with the singular values of J denoted σ_1, σ_2:

Theorem 3 Given σ_1, σ_2, and $\alpha(x)$. Let us abbreviate $\ell = \ell(\alpha(x))$, $u = u(\alpha(x))$, and define $\hat{\sigma}_1, \hat{\sigma}_2$ as follows:

$$(\hat{\sigma}_1, \hat{\sigma}_2) = \begin{cases} \left(\frac{u^2\sigma_1 + u\sigma_2}{u^2+1}, \frac{u\sigma_1 + \sigma_2}{u^2+1} \right) & \text{if } \sigma_2 < \frac{\sigma_1}{u} \\ (\sigma_1, \sigma_2) & \text{if } \frac{\sigma_1}{u} \leq \sigma_2 \leq \frac{\sigma_1}{\ell} \\ \left(\frac{\ell^2\sigma_1 + \ell\sigma_2}{\ell^2+1}, \frac{\ell\sigma_1 + \sigma_2}{\ell^2+1} \right) & \text{if } \sigma_2 > \frac{\sigma_1}{\ell} \end{cases}$$

Let $\hat{S}$ be the diagonal matrix with $\hat{\sigma}_1$ and $\hat{\sigma}_2$ on its diagonal. Then $g = U\hat{S}V^T$ is the solution to the minimization problem (6). ■

If $\lambda(x) > \lambda_s$, we would like to solve

$$\min_{g \in G_{sr}(\beta(x))} \|J - g\|_F^2 \tag{7}$$

This situation is somewhat less complicated, though still non-trivial:

Theorem 4 Given σ_1, σ_2, and $\beta(x)$. Let us abbreviate $\ell = \ell(\beta(x))$, $u = u(\beta(x))$. Let $\bar{\sigma} = (\sigma_1 + \sigma_2)/2$ and define $\hat{\sigma}$ as follows:

$$\hat{\sigma} = \begin{cases} \ell & \text{if } \bar{\sigma} < \ell \\ \bar{\sigma} & \text{if } \ell \leq \bar{\sigma} \leq u \\ u & \text{if } \bar{\sigma} > u \end{cases}$$

Let $\hat{S} = \hat{\sigma} I$ where I is the identity matrix. Then $g = U\hat{S}V^T$ is the solution to the minimization problem (7). ■

Let us again return to the issue of foldovers. By expliciting constraining the transformations so that their singular values are positive (see the definitions of $G_{as}(\cdot)$ and $G_{sr}(\cdot)$), we ensure that the output of the local step is a legal (i.e. not folded over) transformation. As in the case of image resizing, this strongly mitigates the problem of foldovers.

4.3. The Global Step

In this application, we discretize by triangulating the 2D shape, given as a simple polygon. The triangulation may include as many interior points as necessary. We therefore assume that the deformation f is piecewise-linear in the natural way, so that the Jacobian J_f, and hence the allowed transformation g, are constant over the triangles, indexed by τ. With the assumption of piecewise-linearity of f and the allowed transformations taken as given (coming from the local step), the global step then reduces to

$$\min_f \sum_\tau A(\tau) \left\| J_f(\tau) - g(\tau) \right\|_F^2$$

where $A(\tau)$ is the area of the triangle τ. Using a standard result of [PP93], we can rewrite the global step as

$$\min_f \sum_\tau \sum_{i=1}^{3} \mu_\tau^{i,i+1} \| (f(x_{v(\tau,i+1)}) - f(x_{v(\tau,i)})) \\ - g(\tau)(x_{v(\tau,i+1)} - x_{v(\tau,i)}) \|^2 \qquad (8)$$

where τ refers to a triangle in the mesh; i is an index from 1 to 3, where $v(\tau, i)$ is the i^{th} vertex of triangle τ (and the term $i+1$ is taken in a cyclic sense, so that $3+1=1$); and $x_{v(\tau,i)}$ are the coordinates of vertex $v(\tau,i)$ in the original mesh. (Note, however, that not all of the vertices so defined are unique, as most belong to more than one triangle τ.) Finally, the term $\mu_\tau^{i,i+1}$ is the well known cotangent weight, i.e. the cotangent of the angle opposite edge joining vertices $v(\tau,i)$ and $v(\tau,i+1)$ within the triangle τ.

It is clear that the minimization in (8) is quadratic in the f's, and therefore the solution to the minimization problem will be obtained from a linear set of normal equations. In fact, Liu et al. [LZX*08] show that the above sum may be written as a sum over half-edges, and that the resulting linear system can be written as n 2D equations, of which the following is the i^{th}:

$$\sum_{j \in \mathcal{N}(i)} \left[\cot(\theta_{ij}) + \cot(\theta_{ji})\right] (f_i - f_j) \\ = \sum_{j \in \mathcal{N}(i)} \left[\cot(\theta_{ij}) g(\tau(i,j)) + \cot(\theta_{ji}) g(\tau(j,i))\right] (x_i - x_j)$$

where there are n vertices, $\mathcal{N}(i)$ are the vertices which are neighbours of i; $\tau(i,j)$ is the triangle containing half-edge (i,j); θ_{ij} is the angle opposite half-edge (i,j) in triangle $\tau(i,j)$; and f_i is shorthand for $f(x_i)$. Solution of this linear system for the unknowns f_i is the global step - the Poisson equation.

Note, of course, that this system of equations must be solved subject to the user-specified constraints, which in this case take the form $f_{i_k} = f_{i_k}^0$ for $k = 1, \ldots, m$. In this case, the equations corresponding to the indices i_k are replaced by the constraints.

As in the image resizing application, the Poisson system matrix is sparse and does not change between iterations, or when the user input changes. Thus the matrix may be prefactored and back-substitution used in subsequent solves.

4.4. Implementation and Results

The local-global algorithm brings us to a local minimum of the deformation energy. In order to ensure that the local minimum is a reasonable one, i.e. is sufficiently close to the global minimum, it is important to properly initialize the algorithm. We use the well known Least Squares Conformal Map (LSCM) technique of Levy et al. [LPRM02] for this purpose. This has already been shown to be equivalent to minimizing a deformation energy which allows similarity transformations [LZX*08], and is a simple sparse linear system.

Our application loads an image and the user manually draws a polygonal contour around the region of the shape to be deformed (automatic segmentation may also be used). The interior of the contour is triangulated at some user-specified triangle density. For each triangle the application estimates its importance by averaging the gradients of the pixels in its interior. Interactively specifying positional constraints at a few triangulation vertices by moving them in the image plane, the shape is deformed by applying the optimization algorithm to minimize the deformation energy.

We ran our deformation algorithm with and without saliency information, and compared the results to other algorithms which aim to preserve isometries in the deformation. The only exception is the simple "As-Similar-As-Possible" LSCM algorithm, which preserves similarities, but is easy to compute (using a sparse linear solver), thus convenient to use as an initial guess for other iterative algorithms. The more elaborate algorithms are those of [IMH05] (ARAP), which is essentially equivalent to just one iteration of our energy-based deformation (EBD) algorithm (initialized using the LSCM result) and the rigid version of the MLS deformation algorithm of [SMW06], which does not operate on a triangulation. Rather, it generates a continuous mapping of $\mathbb{R}^2$ to $\mathbb{R}^2$ based on just the control points.

Figure 5 shows the results on a number of sample images. The control points are shown as colored dots and are identical in all deformations of the same image. For all images, the LSCM result will typically contain artificial scaling in many image regions, due to the conformal nature of the mapping. The MLS result, not respecting the boundary of the image region, and taking into account only Euclidean distances in

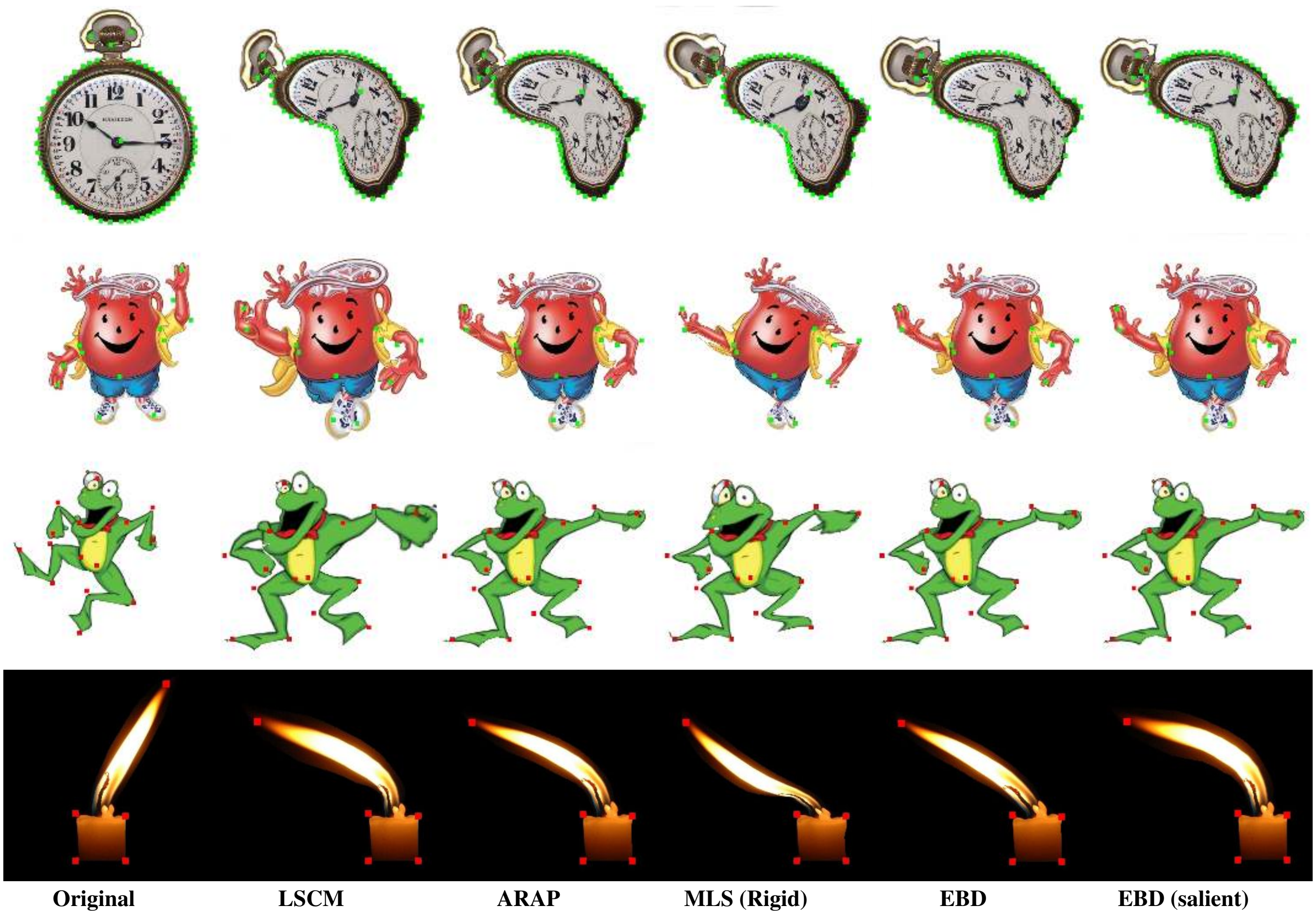

Figure 5: *Comparison of image deformation algorithms.*

the plane, is clearly inferior. The Clock sequence depicts an attempt to "bend" an image of a clock a-la Dali. Since a large number of control points are used, all the results are somewhat similar. However, our EBD result has the advantage of no foldovers in the image (in the concave region), and a better deformation of the winding head. The Koolaid^{TM} and Frog sequences show cartoon characters which are modified to form a keyframe for an animation sequence. Our EBD result is clearly more pleasing than ARAP, as it better preserves isometries in the image (note the hand and hair of Koolaid and the right eye of the frog). The saliency improves the result somewhat. In the Candle sequence, our EBD result with saliency better preserves the isometry of the region next to the candle wick, resulting in a more natural bend of the flame.

With regard to performance, the iterative algorithm terminated when all grid points did not move by more than half a pixel between iterations. To gauge the speed of the algorithm, we ran an experiment similar to that run for resizing (see Section 3.4). We found that the average number of iterations until convergence was 10, when initialized with LSCM. In an interaction application, where the user "drags" the control points continuously in the plane, and the deformation is constantly updated to reflect the changing positions of the constraints, the iterative solver may be initialized with these positions from the previous event in time. This reduces the number of iterations required to 3-4.

5. Conclusions

We have provided a very general framework for content-aware image deformation by energy minimization. The energy function incorporates the key elements of what is considered a good deformation. This energy has been shown to generalize a number of energies used in the literature so far, and proven to be useful in a number of important applications, generating results which are generally superior to competing state-of-the-art algorithms. The alternating least-squares "local-global" algorithm for minimizing such an energy is extremely simple to implement, and quite efficient in practice.

The framework is general enough to work in any dimension, thus the next natural step would be to apply it to 3D shape deformation.

Acknowledgments

Thanks to the many Flickr.com users whose images were used in this paper. The Garfield character is a creation of Jim Davis and Twentieth Century Fox. The KoolAid character is owned by General Foods. The work of C. Gotsman was partially supported by Israel-Niedersachsen (Volkswagen Foundation) grant #ZN2046, the Israel Ministry of Science and the Fund for the Promotion of Research at the Technion.

References

[AG99] ARAD N., GOTSMAN C.: Enhancement by image-dependent warping. *IEEE Trans. Im. Proc. 8* (1999), 1063–1074.

[AS07] AVIDAN S., SHAMIR A.: Seam carving for content-aware image resizing. *ACM Transactions on Graphics (Proc. SIGGRAPH) 26*, 3 (2007), 10.

[ESA07] EITZ M., SORKINE O., ALEXA M.: Sketch based image deformation. In *Proc. Vision, Modeling and Visualization (VMV)* (2007), pp. 135–142.

[FH07] FANG H., HART J.: Detail preserving shape deformation in image editing. In *Proc. SIGGRAPH* (2007), ACM.

[GSCO06] GAL R., SORKINE O., COHEN-OR D.: Feature-aware texturing. In *Proc. Symp. Rendering* (2006).

[IKN98] ITTI L., KOCH C., NIEBUR E.: A model of saliency-based visual attention for rapid scene analysis. *IEEE Trans. Patt Analysis and Mach. Intel.* (1998), 1254–1259.

[IMH05] IGARASHI T., MOSCOVICH T., HUGHES J.: As-rigid-as-possible shape manipulation. *Proc. SIGGRAPH 2005 24*, 3 (2005), 1134–1141.

[LPRM02] LEVY B., PETITJEAN S., RAY N., MAILLOT J.: Least squares conformal maps for automatic texture atlas generation. *ACM Transactions on Graphics 21*, 3 (2002), 362–371.

[LZX*08] LIU L., ZHANG L., XU Y., GOTSMAN C., GORTLER S.: A local/global approach to mesh parameterization. In *Proc. Eurographics Symposium on Geometry Processing* (2008).

[PP93] PINKALL U., POLTHIER K.: Computing discrete minimal surfaces and their conjugates. *Exp. Math. 2*, 1 (1993), 15–36.

[RKB04] ROTHER C., KOLMOGOROV V., BLAKE A.: " GrabCut": interactive foreground extraction using iterated graph cuts. *ACM Transactions on Graphics 23*, 3 (2004), 309–314.

[RSA08] RUBINSTEIN M., SHAMIR A., AVIDAN S.: Improved seam carving for video retargeting. *ACM Trans. Graph. 27*, 3 (2008), 1–9.

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

[SMW06] SCHAEFER S., MCPHAIL T., WARREN J.: Image deformation using moving least squares. *Proc. SIGGRAPH 25*, 3 (2006), 533–540.

[WBCG09] WEBER O., BEN-CHEN M., GOTSMAN C.: Complex barycentric coordinates with applications to image deformation. *Computer Graphics Forum 28*, 2 (2009).

[WGCO07] WOLF L., GUTTMANN M., COHEN-OR D.: Non-homogeneous content-driven video-retargeting. In *Proc. ICCV* (2007), pp. 1–6.

[WTSL08] WANG Y.-S., TAI C.-L., SORKINE O., LEE T.-Y.: Optimized scale-and-stretch for image resizing. *ACM Transactions on Graphics (Proc. ACM SIGGRAPH Asia) 27*, 5 (2008).

[WXXC08] WANG Y., XU K., XIONG Y., CHENG Z.-Q.: 2D shape deformation based on rigid square matching. *Computer Animation and Virtual Worlds 19*, (3-4) (2008), 411–420.

Appendix

Proof of Theorem 1

Define the sequence $\{E_s\}$ by $E_{2t-1} = E[f_{t-1}, g_t]$ and $E_{2t} = E[f_t, g_t]$. Then by definition of the local step, $E[f_{t-1}, g_t] \leq E[f_{t-1}, g]$ for all g, so that in particular, $E[f_{t-1}, g_t] \leq E[f_{t-1}, g_{t-1}]$, so that $E_{2t-1} \leq E_{2t-2}$ for any t. Similarly, by definition of the global step, $E[f_t, g_t] \leq E[f, g_t]$ for all f, so that in particular, $E[f_t, g_t] \leq E[f_{t-1}, g_t]$, so that $E_{2t} \leq E_{2t-1}$ for any t. In summary, then, $E_{s+1} \leq E_s$ for all s. Note also that the functional $E[\cdot,\cdot]$ is bounded below by 0. Then by the Monotone Convergence Theorem of analysis, the sequence $\{E_s\}$ converges to a finite limit E^*, to which there is a corresponding pair of arguments (f^*, g^*). (Note: it is possible to construct a case in which $\{E_s\}$ converges to a finite limit E^*, but the arguments (f^*, g^*) do not themselves converge. However, in this case, we will be equally happy to take any such (f^*, g^*), as they all yield the same value E^* of the deformation energy. (In fact, this case is never observed in practice.))

To see that this limit is local optimum of $E[\cdot,\cdot]$, proceed by contradiction, i.e. suppose that (f^*, g^*) is not a local optimum. Then there exists an f such that $E[f, g^*] < E[f^*, g^*]$ (or a g such that $E[f^*, g] < E[f^*, g^*]$; the cases are symmetric). But then $f^* \neq \arg\min_f E[f, g^*]$, which is a contradiction. (And again, this argument is easily extended to the case in which the arguments themselves do not converge.) ■

Proof of Theorem 2

Using the notation of Section 3.2 and dropping the q subscripts for convenience, the optimization problem is easily shown to be equivalent to

$$\min_{a,b} D^1 a^2 - 2E^1 a + D^2 b^2 - 2E^2 b = \Gamma(a,b)$$

subject to the constraints $b \leq a \leq \psi b$, and $a, b \geq 0$. This is a standard non-linear programming problem, for which the usual techniques (e.g. Karush-Kuhn-Tucker conditions) can be shown to generate the solution given in the theorem. In particular, $(\tilde{a}, \tilde{b})$ is the point on the line $a = b$ which minimizes $\Gamma(a,b)$, while $(\breve{a}, \breve{b})$ is the point on the line $a = \psi b$ which minimizes $\Gamma(a,b)$. The results follow. ■

Proof of Theorem 3

We begin with a lemma.

Lemma: Let A and $\hat{A}$ be 2×2 matrices with singular value decompositions $A = USV^T$ and $\hat{A} = \hat{U}\hat{S}\hat{V}^T$. Suppose that $\hat{S}$ is fixed, but we can choose $\hat{U}$ and $\hat{V}$. Then for *any* S and $\hat{S}$, $\|A - \hat{A}\|_F^2$ is minimized when $\hat{U} = U$ and $\hat{V} = V$.

Proof: $\|A - \hat{A}\|_F^2 = tr((A - \hat{A})^T (A - \hat{A})) = tr(A^T A) + tr(\hat{A}^T \hat{A}) - 2tr(A^T \hat{A})$. Now, $tr(A^T A)$ does not depend on $\hat{U}$ and $\hat{V}$; furthermore, $tr(\hat{A}^T \hat{A}) = \hat{\sigma}_1^2 + \hat{\sigma}_2^2$, which also does not depend on either $\hat{U}$ or $\hat{V}$. Thus, it is sufficient to show that $tr(A^T \hat{A})$ is maximized when $\hat{U} = U$ and $\hat{V} = V$.

Now, using the standard identity $tr(XY) = tr(YX)$, we obtain $tr(A^T \hat{A}) = tr(VSU^T \hat{U}\hat{S}\hat{V}^T) = tr(U^T \hat{U}\hat{S}\hat{V}^T VS) =$

$tr(\tilde{U}\hat{S}\tilde{V}S)$, where $\tilde{U} = U^T\hat{U}$ and $\tilde{V} = V^T\hat{V}$ are both orthogonal matrices. It is a matter of simple algebra to show that

$$\begin{aligned} tr(\tilde{U}\hat{S}\tilde{V}S) &= \hat{\sigma}_1\tilde{u}_{11}\sigma_1\tilde{v}_{11} + \hat{\sigma}_2\tilde{u}_{12}\sigma_1\tilde{v}_{21} + \hat{\sigma}_1\tilde{u}_{21}\sigma_2\tilde{v}_{12} \\ &\quad + \hat{\sigma}_2\tilde{u}_{22}\sigma_2\tilde{v}_{22} \\ &= (\hat{\sigma}_1\sigma_1 + \hat{\sigma}_2\sigma_2)\tilde{u}_{11}\tilde{v}_{11} - (\hat{\sigma}_2\sigma_1 + \hat{\sigma}_1\sigma_2)\tilde{u}_{12}\tilde{v}_{12} \\ &\equiv A_1\tilde{u}_{11}\tilde{v}_{11} - A_2\tilde{u}_{12}\tilde{v}_{12} \end{aligned}$$

where we have used the fact that $\tilde{u}_{11} = \tilde{u}_{22}$ and $\tilde{u}_{12} = -\tilde{u}_{21}$, since $\tilde{U}$ is an orthogonal matrix, and likewise for the $\tilde{v}$'s. Due to the orthogonality, we have two further constraints, namely $\tilde{u}_{11}^2 + \tilde{u}_{12}^2 = 1$, and and likewise for the $\tilde{v}$'s. Writing a Lagrangian and taking derivatives with respect to $\tilde{u}_{11}$ gives $A_1\tilde{v}_{11} - \lambda_u\tilde{u}_{11} = 0$, where λ_u is the Lagrange multiplier on the constraint $\tilde{u}_{11}^2 + \tilde{u}_{12}^2 = 1$. Taking the derivative with respect to $\tilde{v}_{11}$ gives $A_u\tilde{u}_{11} - \lambda_v\tilde{v}_{11} = 0$. Combining these two equations yields

$$\left(1 - \frac{\lambda_u\lambda_v}{A_1^2}\right)\tilde{v}_{11} = 0$$

Thus, either $\lambda_u\lambda_v = A_1^2$ or $\tilde{v}_{11} = 0$. Taking derivatives of the Lagrangian with respect to $\tilde{u}_{12}$ and $\tilde{v}_{12}$ similarly yields that either $\lambda_u\lambda_v = A_2^2$ or $\tilde{v}_{12} = 0$. Now, it is not possible that $\lambda_u\lambda_v = A_1^2$ and $\lambda_u\lambda_v = A_2^2$ simultaneously, since in general $A_1 \neq A_2$. Further, it is not possible that $\tilde{v}_{11} = 0$ and $\tilde{v}_{12} = 0$ simultaneously, since $\tilde{v}_{11}^2 + \tilde{v}_{12}^2 = 1$. Thus, either $(\tilde{v}_{11}, \tilde{v}_{12}) = (1, 0)$ or $(\tilde{v}_{11}, \tilde{v}_{12}) = (0, 1)$. Using similar logic, we can show from the derivatives of the Lagrangian that when $\tilde{v}_{11} = 0$, then $\tilde{u}_{11} = 0$, and the same for $\tilde{v}_{12}$ and $\tilde{u}_{12}$. Re-examining the relevant expression,

$$\begin{aligned} tr(\tilde{U}\hat{S}\tilde{V}S) &= A_1\tilde{u}_{11}\tilde{v}_{11} - A_2\tilde{u}_{12}\tilde{v}_{12} \\ &= A_1\tilde{u}_{11}^2 - A_2\tilde{u}_{12}^2 \end{aligned}$$

and noticing that since $\sigma_1 \geq \sigma_2$ and $\hat{\sigma}_1 \geq \hat{\sigma}_2$, we must have that $A_1 \geq A_2$, so that the proper choice is $\tilde{u}_1 1 = 1$. The rest of the relations lead immediately to $\tilde{U} = \tilde{V} = I$, which implies $\hat{U} = U$ and $\hat{V} = V$. ■

Now, we are ready to prove the theorem:

Proof of Theorem 3: Let $J = USV^T$. Looking at the structure of $G_{as}(\alpha)$ and using the lemma, we know that independent of the choice of singular values for g, the optimal g will be of the form $g = U\hat{S}V^T$ for some $\hat{S}$. Given this, we have that

$$\begin{aligned} \|J - g\|_F^2 &= \|U(S - \hat{S})V^T\|_F^2 \\ &= tr(V(S - \hat{S})U^T U(S - \hat{S})V^T) \\ &= tr(V^T V(S - \hat{S})U^T U(S - \hat{S})) \\ &= tr((S - \hat{S})^2) \\ &= (\hat{\sigma_1} - \sigma_1)^2 + (\hat{\sigma_2} - \sigma_2)^2 \end{aligned}$$

Now, the goal is to minimize this function subject to the constraints. If $\frac{\sigma_1}{u} \leq \sigma_2 \leq \frac{\sigma_1}{\ell}$, then σ_1 and σ_2 satisfy the constraints of $G_{as}(\alpha)$, so the optimal choices are just $\hat{\sigma}_i = \sigma_i$ for $i = 1, 2$. If on, the other hand, $\sigma_2 < \frac{\sigma_1}{u}$, then we must choose the $\hat{\sigma}_i$ to lie within the constrained region, yet be as close as possible to σ_1, thereby minimizing $(\hat{\sigma_1} - \sigma_1)^2 + (\hat{\sigma_2} - \sigma_2)^2$. The solution is the point on the line $\hat{\sigma}_2 = \frac{\hat{\sigma}_1}{u}$ which is closest to (σ_1, σ_2), which is precisely the result stated in the theorem. The third case follows analogously. ■

Proof of Theorem 4

Our goal is to solve

$$\min_{g \in G_{sr}(\beta(x))} \|J - g\|_F^2$$

Begin by noting that

$$\begin{aligned} \|J - g\|_F^2 &= tr((J - g)^T(J - g)) \\ &= tr(J^T J) + tr(g^t g) - 2tr(J^T g) \\ &= tr(J^T J) + 2\sigma^2 - 2\sigma tr(J^T R) \end{aligned}$$

where in the third line, we have used the fact that if $g \in G_{sr}(\beta(x))$, then $g = \sigma R$; and that $tr(R^T R) = tr(I) = 2$. Given the fact that J is fixed, our minimization problem is thus equivalent to

$$\min_{\sigma, R} 2\sigma^2 - 2\sigma tr(J^T R) \tag{9}$$

subject to $\ell(\beta) \leq \sigma u(\beta)$ and $R \in SO(2)$.

Now, let the singular value decomposition of J be $J = USV^T$. Then

$$tr(J^T R) = tr(VSU^T R) = tr(U^T RVS) = tr(\hat{R}S)$$

where $\hat{R} = U^T RV$ is also a rotation matrix. This implies that $\hat{R}_{ij} \leq 1$, so that

$$tr(\hat{R}S) = \sum_{i=1}^{2} \hat{R}_{ii}\sigma_i \leq \sum_{i=1}^{2} \sigma_i$$

with equality when $\hat{R}_{ii} = 1$, i.e. when $\hat{R} = I$. The condition $\hat{R} = I$ implies that $U^T RV = I$, which in turn implies that $R = UV^T$. To summarize, $tr(J^T R)$ is maximized when $R = VU^T$; and its maximum value is $\sigma_1 + \sigma_2$.

Having chosen R optimally, we may turn to the problem of choosing σ optimally. The optimization in (9) can be rewritten as

$$\min_{\ell(\beta) \leq \sigma \leq u(\beta)} 2\sigma^2 - 2\sigma(\sigma_1 + \sigma_2)$$

If σ is unconstrained, then setting the derivative to 0 gives the condition $\sigma = (\sigma_1 + \sigma_2)/2 \equiv \bar{\sigma}$. The constrained solution follows in a straightforward manner. ■