Pacific Graphics 2009
S. Lee, D. Lischinski, and Y. Yu
(Guest Editors)

Volume 28 (2009), Number 7

Optimizing Structure Preserving Embedded Deformation for Resizing Images and Vector Art

Qi-xing Huang [1], Radomir Mech[2], and Nathan Carr[2]

[1] Stanford University, [2] Adobe Systems Incorporated

Abstract

Smart deformation and warping tools play an important part in modern day geometric modeling systems. They allow existing content to be stretched or scaled while preserving visually salient information. To date, these techniques have primarily focused on preserving local shape details, not taking into account important global structures such as symmetry and line features. In this work we present a novel framework that can be used to preserve the global structure in images and vector art. Such structures include symmetries and the spatial relations in shapes and line features in an image. Central to our method is a new formulation of preserving structure as an optimization problem. We use novel optimization strategies to achieve the interactive performance required by modern day modeling applications. We demonstrate the effectiveness of our framework by performing structure preservation deformation of images and complex vector art at interactive rates.

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

1. Introduction

Realistic deformation of digital objects requires minimizing visually perceptible distortions. The existing work in this area has focused on preserving high frequency signals such as local shape features [Sor06, GSCO06, ESA07, WGCO07, AS07, SSP07, ZHM08, WLT08, WTSL08, KSSCO08] or motion structures such as skeletons [HSL*06, SZT*07].

Digital shapes not only contain visually prominent local features, but often possess global structures such as symmetries, repeated contents and line features. As studied in visual physiology [Arn74], the human visual system is very sensitive in detecting these global shape structures. In many cases the global form of an object is intricately tied to its underlying functionality as demonstrated by the fan example in figure 2. For these reasons, analyzing and maintaining a shape's global structure is of critical importance for generating both pleasing and realistic shape deformations.

In this paper, we consider preserving two types of global structures in images and vector art: line features and symmetries. We consider the following features: line segments and vanishing points that frequently occur in natural images and reflectional, rotational and translational symmetries which are popular in 2d vector art.

To accomplish the editing process, we introduce a generalized embedded deformation framework [SSP07]. We start with detecting global structures and computing a *rigidity map* from the input scene. This embedded deformation framework is modified to incorporate structure preserving constraints. Using this framework, the resizing process can then be formulated as a constrained optimization problem. The careful design of this optimization problem yields an efficient solution. Using our method we achieve interactive rates allowing the artist to freely manipulate the artwork undergoing the deformation process.

When computing the rigidity map, we decompose visually salient features into global structures and local features. Unlike the previous works that try to preserve the shapes of all the salient features, we preserve the shapes of local features while allowing global structures to deform. This strategy exhibits more flexibility in resizing images since in many cases the input images are filled with many locally salient features that can overly restrict the deformation process. The visual distortion in our framework is minimized by preserving the **configuration of global structures** in the artwork.

(a) Input (b) Ours (c) Homo. Scaling (d) [WGCO07] (e) Direct SC (f) Indirect SC (g) [WTSL08]

Figure 1: *Resizing an image in which the line features and the people in the middle are prominent features. Our method tries to preserve these line features and shape of the people together which exhibits much less visual distortion compared with existing techniques. (a) Upper: The input image. Bottom: the line features and the deformation map. (b) Our method. (c) Homogeneous scaling. (d) Content-driven Retargeting [WGCO07]. (e) Seam Carving [AS07]. (f) Indirect Seam Carving [AS07] with the people and the lane being marked. (g) Optimized Scale-and-Stretch [WTSL08].*

The effectiveness of our framework is demonstrated by the interactive editing of various types of vector art and images that contain global structures. Experiments show that our method generates visually more pleasing results than previously published work.

1.1. Related Work

MLS Deformation. Schaefer et al. introduced a very effective image deformation tool based on minimizing a MLS system. They showed that one can manipulate image using only a few control points [SMW06]. However, structure preservation is not considered by their method.

Context-Aware Editing. Context aware editing has been addressed in a grid-based embedded deformation [GSCO06, ESA07, WTSL08, WLT08, KSSCO08]. The idea is to let the feature objects deform rigidly or similarly and use an affine deformation to smoothly interpolate unimportant regions. These techniques are able to preserve the shape of features or axis aligned reflection symmetries. However, they are unable to preserve global structures in the general setting since structure preservation is not imposed explicitly.

Avidan and Shamir proposed Seam Carving [AS07, RSA08, SA09] for image resizing. They adaptively remove or add vertical and horizontal seams to generate the output image. Since seam carving manipulates images at the pixel level, it is a challenging task to incorporate symmetry preservation into this process because global structures such as symmetries and line features are resolution independent.

Recently, the inverse patch transform [CBAF08] and bidirectional similarity [SCSI08] have been used for image editing. In both methods, the patch similarities between the input and the target image are maximized. In particular, these two methods are able to achieve some interesting editing results such as removing rows and columns of grid patterns, however, there is no guarantee that they can preserve line features and symmetries since these constraints are not incorporated. Moreover, they are not suitable for interactive editing due to their running times.

Symmetric Deformation. Mitra et al. [MGP07] introduced a symmetrization method to make approximately symmetric shapes more symmetric. A contemporary work is also described in [PGR07], which focused on symmetric remeshing of 3d shapes. Conceptually, it is possible to apply these symmetrization techniques to symmetry preserving editing. For example, one can apply standard shape deformation techniques to deform the shape followed by these symmetrization methods to make the shape more symmetric. However, we implemented this strategy and found that it converges very slowly, particularly when the symmetric deformation is large.

Structure Recovery. Realistic structure preserving editing relies on robust structure recovery methods. Our technique builds on a long series of works in pattern recognition. We refer the reader to [MGP06, PSG*06, CHL*07, YM09] for recent advances in symmetry detection and to [EMED08] for state-of-art line feature detection algorithms.

The remainder of this paper is organized as follows. In Sec. 2, we present the overview of our method. In Sec. 3, we describe how to detect important global structures from the input image/vector art. Then in Sec. 4, we describe how to preserve these detected structures in the embedded deformation framework and in Sec. 5, we show how to efficiently solve the induced constrained optimization problem. We demonstrate the effectiveness our method in Sec. 6 through resizing various images and vector art. Finally, we conclude this paper and introduce some future directions in Sec. 7.

2. Overview

As shown in Fig. 2, our method proceeds in three stages. In the first stage, we analyze the input scene in order to detect its global structures and we compute the corresponding structure-aware deformation map. Then in the second stage, we feed the detected global structures and the structure-aware deformation map into our generalized embedded deformation framework to construct the corresponding constrained optimization problem. Finally, given the user input,

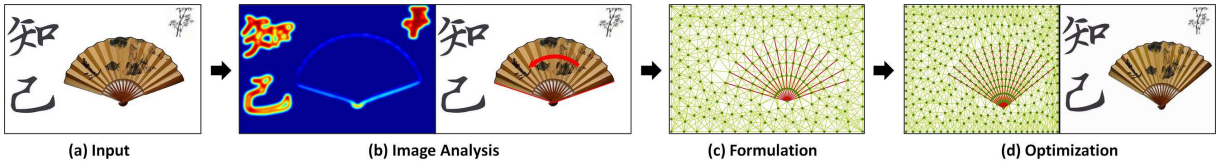


Figure 2: *The pipeline of our method. (a-b) Given the input image, we first perform image analysis to compute the underlying global structures and the rigidity map. (b)In the second step, we feed the structures and the rigidity map in our generalized embedded deformation framework and formulate image resizing as an constrained optimization problem. (c) In the last step, we use a novel coarse-to-fine optimization strategy to solve this optimization problem in real-time.*

we optimize the embedded deformations to resize images by solving this constrained optimization problem. We now describe each step in more detail. Throughout the rest of this paper, we will regard vector art as images unless otherwise specified.

Image Analysis. We detect four type of global structures in images: vanishing points and their associated line segments, reflection symmetries, translational symmetries and rotational symmetries. For vanishing points and their associated line segments, we employ the Hough transform method. To detect symmetries, we modify the methods introduced in [MGP06, PMW*08] to handle images in an efficient way.

Besides detecting structures, we also compute a *rigidity map* to guide the possible deformation applied to the input image. The rigidity map is derived from the visual saliency map [WTSL08] with regions of structures being decayed. This way because we preserve the global structures. On the other hand, if there are no global structures, the rigidity map is similar to the visual saliency map.

Formulation. To preserve the global structures of an input image, we need to have a good deformation model such that the structure preserving constraints can be incorporated. We use a generalized embedded deformation framework [SSP07] for this purpose.

In embedded deformation, the deformation of the input scene is given by a deformation graph. We preserve the detected global structures by placing constraints on the spatial relations between the graph vertex positions. The transformation associated with each vertex is determined by the rigidity map. More precisely, for each vertex in the deformation graph, we introduce a rotation matrix and an affine transformation. The transformations are blended linearly using the weight specified by the rigidity map. By doing this, we avoid sudden changes between rigid transformations and affine transformations in the spatial domain.

Optimization. The optimization problem we are trying to solve is a non-linear constrained optimization problem. As our goal is to resize images at interactive rates, direct and exact solvers are not applicable in this case since we have non-linear terms in both the objective function and the constraints. For this reason, we use an approximate solver that induces realistic deformations.

Our solver decomposes the variables into groups based on their types. We optimize each type of variables subsequently. Using this strategy, the constraints are always linear in the variables which allows for efficient optimization. In addition, we show how our solver is designed to incorporate the power of both Gauss-Newton method [SSP07] and an alternating subset method [SA07], which are frequently used in the context of shape editing.

3. Image Analysis

In this section we describe our steps for detecting global structures in images and constructing the rigidity map that constrains the deformation process. We note that detecting global structures in images is an ongoing and active area of research. We describe here the steps that were taken for the examples in our paper, however, our underlying deformation method does not preclude the use of more advanced image analysis techniques.

3.1. Structure Detection

Line features. We use the standard Hough transform to compute vanishing points and their associated line segments. We start with computing edge features from image gradients using hysteresis thresholding [Can86]. Then we polish the edge features by line segment fitting. More precisely, we start at an end point and trace along an edge until the line fitting error is bigger than a user supplied threshold (typically 2 pixels). We repeat this process until all the edges have been traced. This procedure outputs a collection of line segments. Line segments that exceed twenty pixels in length are fed into the Hough transform for detecting vanishing points. Typically there are at most 3 vanishing points in an image. We pick the first three detected vanishing points. To avoid including false positives, we prune any detected vanishing points which have only 2 associated line segments. We also connect line segments that have the same line equations.

Symmetries. We modify the methods introduced in [MGP06, PMW*08] to detect symmetries. The existence of each type of symmetry is tested independently. For fixed type of symmetries, we proceed in three steps to detect its existence. In the first step, we vote for candidate symmetries based on correspondences between the SIFT features [Low04]. In the second step, we filter the candidate

symmetries by clustering in the transformation space. Finally, we use the standard image registration technique to verify filtered symmetries. Each detected symmetry is specified by its type, its symmetric transformation, and the support region in which this symmetry occurs.

To vote for candidate symmetries, a naive way is to directly use the correspondences between SIFT features. However, we found that this approach typically generate a huge set of samples in the transformation space which slows down the clustering step significantly.

Our approach is to use correspondences between pairs of SIFT features. More precisely, we form a pair between each SIFT feature and each of its neighboring SIFT features. We discard pairs that are more than 30 pixels apart. We also discard those features smaller than 15 pixels to ensure that each pair has a robust orientation. We compute matches between line segments. Two pairs are matched if their start and end SIFT features match with each other and the deviation in their lengths is less than 5 pixels. We typically choose a conservative SIFT feature matching threshold in order to include sufficient evidences of the underlying symmetries.

The candidate symmetries are selected as follows. If we are detecting rotational symmetries, the rotation angle and the rotation center is the angle between the directions of matched pairs and their intersection points, respectively. For translational symmetries, we further require that the angle between the directions is less than 30 degrees. The translation vector is simply the vector between their middle points. Similarly, for reflection symmetries, we compute a reflection axis based on their middle points. We choose this reflection axis if the angle between this reflection axis and the bisector of the two line segments is less than 30 degrees.

Clustering in the transformation space is also performed differently among different types of symmetries. For reflection symmetries, we apply the mean-shift algorithm introduced in [MGP06]. The center of rotation is also determined in the same way. For rotational and translational symmetries, we apply the generalized Hough transform technique introduced in [PMW*08].

3.2. Rigidity Map

Fig. 3.2 illustrates the process of computing the rigidity map $\mathbf{D}: \mathcal{R}^2 \rightarrow [0,1]$ which is a mapping between the image domain and a value in $[0,1]$. The rigidity map D is derived from two other maps: the visual saliency map [WTSL08] $\mathbf{F}: \mathcal{R}^2 \rightarrow [0,1]$ and the structure map $\mathbf{S}: \mathcal{R}^2 \rightarrow [0,1]$. The visual saliency map $\mathbf{F}$ encodes visual saliency information of the input image. We compute $\mathbf{F}$ as the convolution between the well-known image gain map $\mathbf{I}$ [FLW02] and the disk filter $B_r(\mathbf{x}) = \frac{3}{4\pi r^3}$, $\|\mathbf{x}\| < r$ and $B_r(\mathbf{x}) = 0$ otherwise:

$$\mathbf{F}(\mathbf{x}) = \mathbf{I}(\mathbf{x}) \otimes B_r(\mathbf{x}). \quad (1)$$

We set $r = 5$ pixels in this paper.

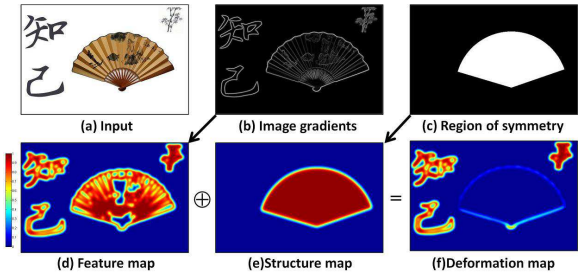


Figure 3: *The rigidity map is computed from a feature map with the signals of structured regions being decayed by the structure map.*

Similarly, the structure map is defined as

$$\mathbf{S}(\mathbf{x}) = \mathbf{S}'(\mathbf{x}) \otimes B_r(\mathbf{x}). \quad (2)$$

Where $\mathbf{S}'(\mathbf{x})$ is a binary map that takes value 1 if $\mathbf{x}$ is within distance r to a detected structure or its support region except that for reflection symmetry. The structure map basically describes where and to what extent we can relax the rigidity preservation. We don't take reflection symmetries into account because there are still rich features in their support regions.

Given the feature map $\mathbf{F}$ and the structure map $\mathbf{S}$, we finally define the rigidity map as

$$\mathbf{D}(\mathbf{x}) = \mathbf{F}(\mathbf{x})(1 - \mathbf{S}(\mathbf{x})). \quad (3)$$

We also normalize the rigidity map such that its maximum value is 1 and its minimum value is 0.

Fig. 3.2 illustrates the behavior of the rigidity map $\mathbf{D}$. If we only used the feature map to guide the deformation as most context aware resizing techniques do, we would preserve both the shapes of the fan and the characters. As we will see later though, this could be problematic if we stretch the image drastically. In the rigidity map, the shape of the fan is not preserved. However, by using the rigidity map and preserving the rotation symmetry, we achieve the effect of folding/unfolding. By doing so, visual distortion is prevented even when the input image is stretched drastically.

4. Formulation

We now describe how to formulate the structure preserving resizing problem using our embedded deformation framework. We start with reviewing the embedded deformation framework [SSP07] in Sec. 4.1. Then in Sec. 4.2 and in Sec. 4.3, we show how to extend the original framework to preserve structures and how to incorporate the rigidity map.

4.1. Embedded Deformation

In the embedded deformation, the deformation of the input image is controlled by a deformation graph $\mathcal{G}$. The deformation graph consists of a set of vertices v_i where $1 \leq i \leq N$

and a set of edges which connect vertices with their k-nearest neighbors (k=9 in our case). We will use $\mathcal{N}_i$ to denote the indices of the neighboring vertices of v_i. Note that in the original framework, vertices are uniformly distributed. In the next section, we will show how to sample the vertices to match the detected structures.

Each vertex v_i has two positions $\mathbf{p}_i$ and $\mathbf{q}_i$. $\mathbf{p}_i$ is its fixed position at which it is sampled and $\mathbf{q}_i$ is its active position after the deformation graph has been deformed.

Each vertex v_i is associated with an affine transformation matrix R_i. R_i describes how the scene deforms in neighborhood of v_i. The consistency between the neighborhood of v_i and the this transformation matrix R_i is measured as

$$e_i^c = \sum_{j\in\mathcal{N}_i} \|R_i(\mathbf{p}_i - \mathbf{p}_j) - (\mathbf{q}_i - \mathbf{q}_j)\|^2. \tag{4}$$

R_i also induces a transformation $\mathcal{T}_i(\cdot) : \mathcal{R}^2 \to \mathcal{R}^2$ in the neighborhood of v_i as

$$\mathcal{T}_i(\mathbf{x}) = R_i(\mathbf{x} - \mathbf{p}_i) + \mathbf{q}_i. \tag{5}$$

Finally, the deformed position $\mathbf{x}'$ of each point $\mathbf{x}$ in the input scene is parameterized by $\mathbf{q}_i$ and R_i using partition of unity [SSP07] as

$$\mathbf{x}' = \sum_{i=1}^{M} w_i(\mathbf{x})\mathcal{T}_i(\mathbf{x}) / \sum_{i=1}^{M} w_i(\mathbf{x}). \tag{6}$$

The weights $w_{\mathcal{C}_i}(\mathbf{x})$ are given by

$$w_i(\mathbf{x}) = \begin{cases} (1.0 - \|\mathbf{x} - \mathbf{p}_i\|/r_i)^2 & \|\mathbf{x} - \mathbf{p}_i\| < r_i \\ 0 & \text{otherwise} \end{cases} \tag{7}$$

Where r_i is the maximum distance between $\mathbf{p}_i$ and its neighbors.

The user-input is formulated as constraints on the vertex positions. We impose two types of constraints: soft constraints and hard constraints. Soft constraints are formulated as minimizing the sum of squared distances between the deformed handle vertex positions and their target positions:

$$E_h = w_h \sum_{i\in\mathcal{H}} \|\mathbf{q}_i - \mathbf{h}_i\|^2. \tag{8}$$

Where $\mathcal{H}$ collects the indices of the handle vertices.$\mathbf{h}_i$ is the target position of $\mathbf{q}_i$.

Each hard constraint specifies the x or y coordinate of a vertex. For simplicity, we write all the hard constraints in a matrix form as

$$F\mathbf{q} = \mathbf{c}. \tag{9}$$

Here vector $\mathbf{q}$ collects all the deformed positions $\mathbf{q}_i$. Hard constraints are used to fix the boundary of the target image.

We also want to minimize the sum of consistency measures at each vertex in order to make the deformation to be smooth:

$$E_c = \sum_{i=1}^{N} \sum_{j\in\mathcal{N}_i} \|R_i(\mathbf{p}_i - \mathbf{p}_j) - (\mathbf{q}_i - \mathbf{q}_j)\|^2. \tag{10}$$

Combining Equ. 8, Equ. 9 and Equ. 10, we compute the optimal deformed vertex positions by solving the following constrained optimization problem

$$\min_{\mathbf{q},\{R_i\}} \sum_{i=1}^{M} \sum_{j\in\mathcal{N}_i} \|R_i(\mathbf{p}_i - \mathbf{p}_j) - (\mathbf{q}_i - \mathbf{q}_j)\|^2 + w_h \sum_{i\in\mathcal{H}} \|\mathbf{q}_i - \mathbf{h}_i\|^2 \tag{11}$$

subject to $F\mathbf{q} = \mathbf{c}$.

For optimization, the transformations R_i are treated as latent variables and are optimized with $\mathbf{q}_i$. Once $\mathbf{q}_i$ and R_i have been computed, we deform the input scene using Equ. 6.

Although in the embedded deformation framework, there is no guarantee that edges are not flipped. We did not find any edge flip effect in all the examples we have tested.

4.2. Structure Preservation

We formulate structure preservation as constraints on the deformed vertex positions $\mathbf{q}_i$. This leads to two questions: How to initialize the vertices and how to constrain them during deformation. In the following section, we describe how we initialize these samples. Then we introduce the constraints on the vertices to preserve structures. The key idea is the way we parameterize the vertices using structure parameters. Finally, we present an unified constraint expression to ease the discussion in Sec. 5.

Initialization. We introduce two types of vertices. The first type is used to sample the detected structures. The placement of samples differs from structure to structure and it will be discussed below in detail. The second type of vertices is used to uniformly fill the rest of the input domain. The sampling process is controlled by a sampling density δ.

Translational Symmetry. Suppose a translational symmetry s appears on a $m \times n$ grid. Without losing generality, we assume this symmetry is along the x axis of the grid. Since a translational symmetry is given by a translational vector, we generate a grid of vertices $\mathbf{q}_{ij}$ of size $m \times n$ and a constraint:

$$\mathbf{q}_{i+1,j} - \mathbf{q}_{ij} = \mathbf{t}. \tag{12}$$

Where $\mathbf{t}$ is the translational vector which can be changed during manipulation.

Rotational Symmetry. We consider fan-like rotational symmetries in this paper. As shown in Fig. 4, we sample the rotational symmetry using m lines of vertices $\mathbf{q}_{ij}$ where $1 \le i \le m, \quad 1 \le j \le n$. As a rotational symmetry is parameterized by the angle of the first line, the rotation angle and the rotation radius, we constrain the vertices to preserve this symmetry as

$$\mathbf{q}_{ij} = \mathbf{v} + \mathbf{n}(\theta + (i-1)\phi)t_j. \tag{13}$$

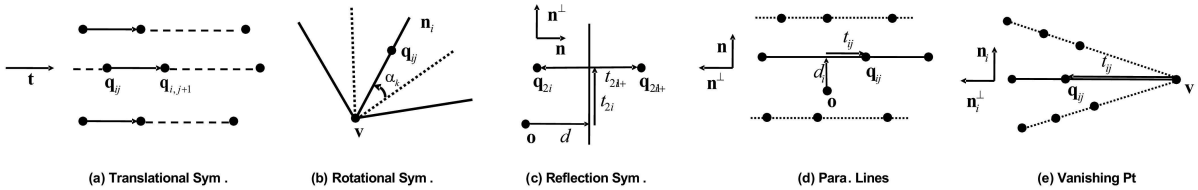


Figure 4: *Structure preserving constraints of each type of global structure.*

Where θ specifies the angle of the first line. ϕ is the rotation angle and t_j is the rotation radius of each trajectory. The normal vector is expressed as $\mathbf{n} = (\cos\theta, \sin\theta)^T$.

If the rotational center resides in the deformation domain, one can also treat the rotation center as an additional vertex $\mathbf{q}_1$ and rewrite the constraints as

$$\mathbf{q}_{ij} - \mathbf{q}_1 = \mathbf{n}(\theta + (i-1)\phi)t_j. \tag{14}$$

Reflection Symmetry. We sample a reflection symmetry s using n pairs of reflection symmetric vertices $(\mathbf{q}_{2i}, \mathbf{q}_{2i+1})$ where $1 \le i \le n$. The reflection axis is parameterized by its normal direction $\mathbf{n}$ and the projected distance d from the origin. Given the coordinates frame centered at the origin $\mathbf{v}$ with $\mathbf{n}$ and $\mathbf{n}^\perp$ being the axes, we can parameterize them as

$$\begin{aligned} \mathbf{q}_{2i} &= \mathbf{n}\cdot(d - t_{2i+1}) + \mathbf{n}^\perp t_{2i} \\ \mathbf{q}_{2i+1} &= \mathbf{n}\cdot(d + t_{2i+1}) + \mathbf{n}^\perp t_{2i}. \end{aligned} \tag{15}$$

Vanishing Point. A vanishing point structure s contains m sets of co-linear vertices $\{\mathbf{q}_{ij}\}$ and $1 \le i \le m, 1 \le j \le j_i$ whose corresponding lines L_i share at a single point $\mathbf{v}$. We can parameterize these vertices as

$$\mathbf{q}_{ij} = \mathbf{v} + \mathbf{n}_i t_{ij}. \tag{16}$$

Where $\mathbf{n}_i$ is the normalized direction of the i-th line.

Parallel Lines. Parallel lines are considered special vanishing point structures where their vanishing points at infinity. In our paper, we use a separate constraint set for parallel lines to prevent numerical instabilities. In the detected vanishing point structures, the ones with norm of vanishing points bigger than 10^4 pixels are treated as parallel lines.

We sample a parallel line structure s using m lines of vertices $\{\mathbf{q}_{ij}\}$ where $1 \le j \le j_i$ and j_i depends on the length of the corresponding line. Once again we add constrains on these vertices by parameterized them using latent variables. The line segments are parameterized by a shared normal direction and a distance d_i to the origin for each line segment. Once the line segments have been specified, each vertex is parameterized using a additional scalar t_{i_j} as

$$\mathbf{q}_{ij} = \mathbf{n}d_i - \mathbf{n}^\perp t_{ij}. \tag{17}$$

Unified Representation. To ease the discussion below, we present a unified representation for structure preserving constraints. Based on the structures of Equ. 12, Equ. 13, Equ. 15, Equ. 16 and Equ. 17, we write down the constraint for each structure s as

$$D_s\mathbf{q} = P_s(\Phi_s)\mathbf{t}_s. \tag{18}$$

Where $\mathbf{t}_s$ encodes all the linear structure parameters such as d_i and t_{ij}, and Θ_s collects all the non-linear structure parameters such as normal directions. We use Θ to denote all the non-linear parameters for simplification.

Due to different purposes, we decompose the set of all the structures $\mathcal{S} = \mathcal{E} \cup \mathcal{A}$ into two subsets where $\mathcal{E}$ contains the structures to be preserved exactly, e.g, by satisfying constraint specified in Equ. 18. $\mathcal{E}$ usually includes line features and rotational symmetries. $\mathcal{A}$ contains structures to be preserved only approximately. Reflection symmetry is typically included in $\mathcal{A}$. Moreover, $\mathcal{A}$ is also used to design efficient solver in Sec. 5.

For each structure $s \in \mathcal{A}$, we penalize the deviation of Equ. 18 in L2-norm. The total structure energy term is given by:

$$E_s = \sum_{s\in\mathcal{A}} w_s \|D_s\mathbf{q} - P_s(\Theta_s)\mathbf{t}_s\|_2^2. \tag{19}$$

Where w_s represents the importance of structure s. In our framework, the default value of $w_s = 0.5$.

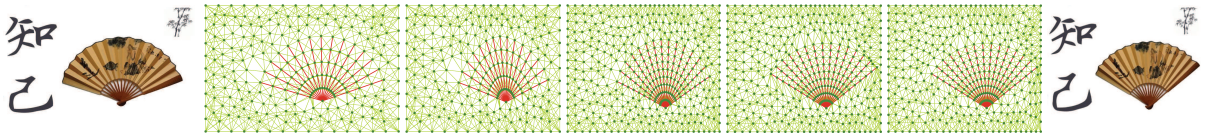
4.3. Feature Preservation

Similar to most of context-aware deformation techniques [ESA07, WGCO07, WTSL08, KSSCO08], we preserve visual salient feature regions by constraining the transformations of the vertices in corresponding regions to be rigid. A straightforward way is to let R_i be a rigid transform if the value of $\mathbf{p}_i$ in the rigidity map $\mathbf{D}(\mathbf{p}_i)$ is above a threshold. However, this strategy would make the transformations change abruptly between feature regions and flat regions.

We address this issue by introducing a rotation R_{i1} and an affine transformation R_{i2} to each vertex v_i. We also make them to be consistent in the neighborhood of v_i. The local consistency measure e_i^c is refined as

$$e_i^c = \sum_{k=1}^{2} \lambda_{ik} \sum_{j\in\mathcal{N}_i} \|R_{ik}(\mathbf{p}_i - \mathbf{p}_j) - (\mathbf{q}_i - \mathbf{q}_j)\|^2. \tag{20}$$

where $\lambda_{i1} = \mathbf{D}(\mathbf{p}_i)$ and $\lambda_{i2} = 1 - \mathbf{D}(\mathbf{p}_i)$. Intuitively, if $\mathbf{D}(\mathbf{p}_i)$ is close to 1, minimizing e_i^c forces the neighborhood of v_i to



(a) (b) (c) (d) (e) (f) (g)

Figure 5: *Different stages of our optimization strategy. (a) The input image with a rotational symmetry.;(b) The coarse deformation graph. (c) Optimize Θ in stage I; (d) Subdivide the deformation graph and optimize $\mathbf{t}$ in Stage II;(e) Enforce exact preservation before stage III (f) Relax vertex positions in stage III. (g) Result.*

deform rigidly. On the other hand, if $\mathbf{D}(\mathbf{p}_i)$ is close to 0, the neighborhood of v_i can be stretched.

Under this new definition, the cumulative consistency measure E_c is modified as

$$E_c = \sum_{k=1}^{2}\sum_{i=1}^{N} \lambda_{ik} \|R_{ik}(\mathbf{p}_i - \mathbf{p}_j) - (\mathbf{q}_i - \mathbf{q}_j)\|^2. \tag{21}$$

The matrix R_i used to deform the input scene can be taken as either R_{i1} and R_{i2}. In our experiments, we found that a better strategy is to let

$$R_i = \lambda_{i1} R_{i1} + \lambda_{i2} R_{i2}. \tag{22}$$

Intuitively, we want R_i to be close to a rotation when $\mathbf{D}(\mathbf{p}_i)$ is big.

For optimization, we parameterize:

$$R_{i1} = \begin{pmatrix} \phi_{i1}^1 & \phi_{i1}^2 \\ \phi_{i1}^3 & \phi_{i1}^4 \end{pmatrix}, R_{i2} = \begin{pmatrix} \cos\phi_{i2}^1 & -\sin\phi_{i2}^1 \\ \sin\phi_{i2}^1 & \cos\phi_{i2}^1 \end{pmatrix} \tag{23}$$

We use Φ_{ik} to denote all the transformation parameters of R_{ik} and use Φ to denote all the transformation parameters.

Combining Equ. 8, Equ. 19, Equ. 21, Equ. 9 and Equ. 18, we now describe the optimization problem that we are trying to solve as follows:

$$\min_{\mathbf{q},\Phi,\mathbf{t},\Theta} E_h + E_c + E_h \tag{24}$$

subject to $F\mathbf{q} = \mathbf{c}$ and $D_s\mathbf{q} = P_s(\Theta_s)\mathbf{t}_s, \forall s \in \mathcal{E}$.

To further simplify the discussion below, we use a more general expression which is described as follows:

$$\begin{aligned} &\min_{\mathbf{x}} \|\mathbf{f}(\mathbf{x})\|^2 \\ &\text{subject to} \quad A\mathbf{q} = B(\Theta)\mathbf{t}. \end{aligned} \tag{25}$$

Here the vector $\mathbf{x} = (\mathbf{q}, \Phi, \mathbf{t}, \Theta)$ collects all the variables. $\mathbf{f}^T\mathbf{f}$ collects all the terms in E_h, E_c and E_s. $A\mathbf{q} = B(\Theta)\mathbf{t}$ encapsulates all the equality constraints.

5. Optimization

Image resizing requires us to find an efficient solver for the optimization problem specified in Equ. 25. A useful property of image resizing or general shape editing is that one can always treat the previous frame as the initial guess of the current frame since the deformation is continuous. In other words, applying local optimization is sufficient to achieve good results. However, for this particular problem we are trying to solve, optimizing it using any direct solver would be very hard since the unknowns comprise variables of different scales and types and there are non-linear parameters in the constraints. Our strategy is to decompose this hard problem into three easier problems and solve them subsequently.

Fig 5 shows the work flow of our optimization technique. Our method proceeds in three stages. At stage I, we estimate the non-linear parameters Θ by treating the rest of the variables as latent variables. As Θ typically consists of directions of reflection axis which are resolution independent, we find that it is sufficient to optimize them at a coarse level (See Fig. 5(c)). Moreover, we let all the structures be preserved approximately such that we only have hard handle constraints which are linear in the variables. We employ the Gauss-Newton method [SSP07], which runs very efficiently for small scale problems. At stage II, we fix the non-linear structure parameters Θ and optimize the linear structure parameters $\mathbf{t}$ by treating $\mathbf{q}$ and Φ as latent variables. Note that at stage II, all the structures are still preserved approximately. Finally, at stage III, we enforce the exact preservation and re-optimize $\mathbf{q}$ and Φ. In both stage II and stage III, we use variants of the alternating optimization method [SA07] on a denser graph.

Stage I: Optimize Θ. At this stage, we let $\mathcal{A} = \mathcal{S}$. In this case, the constraints simply become $F\mathbf{q} = \mathbf{c}$. Each Gauss-Newton iteration improves the current values of the free variables as $\mathbf{x}_{k+1} = \mathbf{x}_k + d\mathbf{x}$ by solving the following optimization problem

$$\begin{aligned} &\min_{d\mathbf{x}} \|\mathbf{f}(\mathbf{x}_k) + \nabla\mathbf{f}(\mathbf{x}_k)(d\mathbf{x})\|^2 \\ &\text{subject to} \quad F \cdot (d\mathbf{q}) = 0. \end{aligned} \tag{26}$$

Where $\nabla\mathbf{f}(\mathbf{x}_k)$ is the Jacobian of $\mathbf{f}(\mathbf{x})$ at $\mathbf{x}_k$.

We employ the SuperLU package [DEG*99] to solve the linear system derived from Equ. 26. Moreover, as the purpose of this step is to estimate the non-linear parameters, typically 3-4 Gauss-Newton iteration is sufficient for this purpose.

We typically use 100-150 vertices for the coarse deformation graph which leads to 400-800 variables. The perfor-

mance of our solver on a machine with 4 1.65GHZ processors is 1ms-2ms per iteration and is 4ms - 8ms in total.

Stage II and III: Optimize t,q and Φ. For simplicity, we introduce stage II and stage III together. When the non-linear structure parameters Θ are fixed, we use the alternating subset method to optimize **q**, **t** and Φ. More precisely, we alternate between optimizing local transformations $\{R_{ik}(\Phi_{ik})\}$ and linear structure parameters **t** with deformed vertex positions **q** being fixed, and optimizing the deformed vertex positions **q** with $\{R_{ik}(\Phi_{ik})\}$ and **t** being fixed.

As can be seen from Equ. 19 and Equ. 21, local transformations R_{ik} and linear structure parameters are decoupled when **q** is fixed. Thus, we can optimize them independently. Optimizing each local transformation R_j^k can be done in a closed way, we refer the reader to [SMW06] for more details. The linear structure parameter $\mathbf{t}_s$ is optimized at:

$$\mathbf{t}_s^* = \arg\min_{\mathbf{t}_s} \|D_s\mathbf{q} - P_s\mathbf{t}_s\|^2 = (P_s^T P_s)^{-1}(P_s^T D_s).$$

When **t**, Θ and Φ are fixed, $f(\mathbf{x})$ is linear in **q**, and so are the constraints. Thus, we can compute the optimal vertex positions $\mathbf{q}^* = \mathbf{q}^k + d\mathbf{q}$ by solving the following linearly constrained quadratic programming problem:

$$\begin{aligned} &\min_{\mathbf{q}} \|\mathbf{f}(\mathbf{x}^k) + \nabla_{\mathbf{q}}\mathbf{f}(\mathbf{x}^k)d\mathbf{q}|^2 \\ &\text{subject to} \quad Ad\mathbf{q} = 0. \end{aligned} \tag{27}$$

We use SuperLU package [DEG*99] to solve the linear system derived from Equ. 27. As can be seen from Equ. 25, the matrix $\nabla_{\mathbf{q}}\mathbf{f}^T\nabla_{\mathbf{q}}\mathbf{f}$ is a constant matrix. Since A is also a constant matrix, it follows that we can pre-factorize the right-hand side of the linear system and use back-substitution in order to solve this linear system. This strategy significantly reduces the cost per iteration.

To detect convergence, we monitor whether $\|\nabla_{\mathbf{q}}F\| < \varepsilon = 1e-2$. We also set a maximum number of iterations for the alternating methods. Typically, 10-20 iterations are sufficient.

At stage II and stage III, we typically use 300-450 vertices for the deformation graph which leads to 1000-1500 variables. Because we can pre-factorize the system, the performance of our solver on the same machine is around 0.5 ms per iteration and is 10ms - 20ms in total.

6. Results

We have implemented our framework for various examples of vector art and images and tested it on a machine with 4 1.65GHZ processors. For the all the example, we achieved 10-30 frames per second.

Resizing Our method is good at resizing images and vector art which contains line features and symmetries. As shown in Fig. 1, our technique is able to preserve both the line features and the people in the input image. The Seam Carving method [AS07] tends to fail in this example (See Fig. 1(e)) due to the dominant diagonal line features such as the lane markers that crosses the image. Even if we constrain Seam Carving to try to avoid distorting both people and lane markers, distortions of straight lines still occur 1(f). Shape preserving based method s [WGCO07, WTSL08] also fail in this example because line features that are not aligned with the image axis are not preserved. Finally, homogeneous scaling does preserve the line features but it distorts the people in the image. In contrast, our method preserves lines by allowing them to smoothly rotate while undergoing deformation. Fig.7 shows additional results of resizing urban scenes.

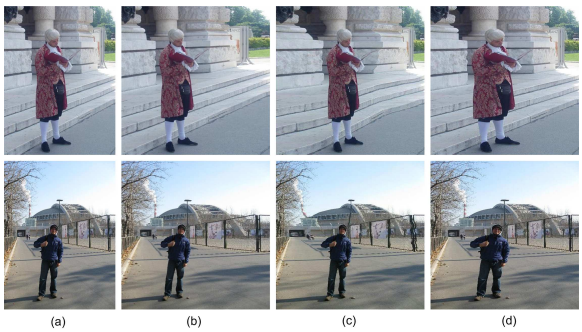


Figure 7: *Resizing images which contains rich line features and people. Compared with alternative methods, our method can preserve both the shapes of the people and the line features (a) Input. (b) Our result. (c) Seam Carving [AS07] with people being marked. (d) Homogeneous scaling.*

Similar behavior is also exhibited in the fan example where our method is able to fold/unfold the fan but the other methods either distort the image or in other cases only scale each individual object(see Fig. 8). Note that for this example, seam carving + uniform scaling [RSA09] could achieve better result if the goal is only to preserve the shape of each individual object. Fig. 9 shows a similar example where we are able to fold the wings of a white peacock.

Figure 8: *Comparison on resizing the Chinese art shown in Fig. 5(a). (a) Ours. (b) [AS07]. (c) [WTSL08].*

Fig. 6 shows a more complicated example where the input image contains a reflection symmetry and three vanishing points. Since we preserve these global structures in our framework, we achieve much better result than previous works.

(a) Input (b) Ours (c) Homogeneous Scaling (d) CDR [WGCO07] (e) Seam Carving [AS07]

Figure 6: *Resizing an image that has one reflection symmetry and three vanishing points. Our method yields much better result when than previous works. (a) Input. (b) Ours. (c) Homo. Scaling. (d) Content-driven [WGCO07]. (e) Seam Carving [AS07].*

(a) (b) (c)

Figure 9: *Results of resizing an image which contains a peacock. (a) Input. (b) Result I. (c) Results II.*

(a) Input (b) Single Layer (c) Two Layers

Figure 10: *Resizing a two-layered vector art. (a) Input. (b) Using one layer. (c) Using two layers where the two translational symmetries in the background are preserved.*

Our framework can be easily extended to vector art with multiple layers. In this case, we can apply different deformations to preserve the structures of different layers. As shown in 10, this strategy can avoid the case where energy terms from different layers adversely compete with each other.

(a) (b) (c)

Figure 11: *Warping vector art that has a global reflection symmetry. Preserving symmetry yields better result than without preserving symmetry. (a) Input. (b) Without preserving symmetry. (c) Preserving symmetry.*

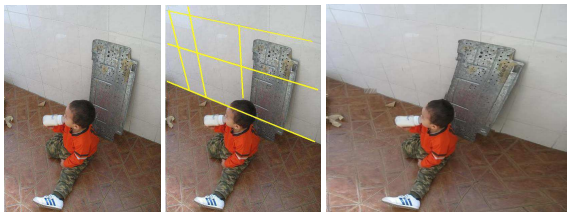
Warping. Our technique can also to be used to warp a given scene. In this case, the user manipulates the scene by placing and moving control handles that we provide. Figure 11 shows a vector art example where we have a prominent reflection symmetry. In this example, we try to preserve this reflection symmetry approximately during warping (see Fig. 11(c)). Compared with Fig. 11(b) where the symmetry is not preserved, we can see that our technique yields a better result. Fig. 12 shows another example where we preserve the partial rotational symmetry exactly. Again we can see that preserving the symmetry yields a more pleasing result.

(a) (b) (c)

Figure 12: *Warping a vector art that has a partial rotation symmetry. (a) Input. (b) Without preserving symmetry. (c) Preserving symmetry.*

Discussion. In our technique we are trying to understand the scene to some extent before applying the deformation. Given the input scene, we decompose it into regions where the shape of features should be preserved and regions where we allow certain deformation but with the global structure being preserved. As we have shown in our examples, this strategy gives more flexibility in deforming the scene than preserving everything indicated by the feature map.

On the other hand, structure preservation relies on robust structure detection methods. Fig. 13 shows an example where we failed to detect the line features on the ground. In this example, visual distortion is presented near these features when pulling the image. In the future, we want to find more reliable structure detection methods or to incorporate human interaction into this process.



(a) (b) (c)

Figure 13: *A partial failure case of our method where line features on the wall and ground are not detected. (a) Input. (b) The detected the line structures. (c) Resizing result. One can see that the wall and the ground are distorted.*

7. Conclusion

In this paper, we introduced a method for preserving global structures while resizing images and vector art. Structure preserving deformation is formulated as an optimization problem where we seek to achieve multiple goals of maximizing both the structural similarity and local similarity. A novel optimization framework is used to achieve real-time performance. Finally, the effectiveness of our framework is demonstrated on both natural images as well as vector art.

There are ample opportunities for future research. In general, detecting global structures in images and vector art content is still an area requiring additional exploration. Furthermore, our method does not explore structure preserving 3D object deformations. We note that structure preserving deformation is a highly over-constrained process. Further analysis could be done to compare the result of the deformation with user expectations. This may lead to a better weighting of constraints.

Acknowledgement: We would like to thank the anonymous reviewers for their helpful comments. Qi-xing Huang was supported by Stanford Graduate Fellowship.

References

[Arn74] ARNHEIM R.: *Art and Visual Perception: A Psycology of the Creative Eye*. University of California Press, 1974.

[AS07] AVIDAN S., SHAMIR A.: Seam carving for content-aware image resizing. *ACM Trans. Graph. 26*, 3 (2007), 10–18.

[Can86] CANNY J.: A computational approach to edge detection. *IEEE Trans. Pattern Anal. Mach. Intell. 8*, 6 (1986), 679–698.

[CBAF08] CHO T. S., BUTMAN M., AVIDAN S., FREEMAN W. T.: The patch transform and its applications to image editing. In *CVPR '08* (2008), pp. 1 – 8.

[CHL*07] CHEN P.-C., HAYS J. H., LEE S., PARK M., LIU Y.: *A Quantitative Evaluation of Symmetry Detection Algorithms*. Tech. Rep. CMU-RI-TR-07-36, 2007.

[DEG*99] DEMMEL J. W., EISENSTAT S. C., GILBERT J. R., LI X. S., LIU J. W. H.: A supernodal approach to sparse partial pivoting. *SIAM J. Matrix Analysis and Applications 20*, 3 (1999), 720–755.

[EMED08] EL MEJDANI S., EGLI R., DUBEAU F.: Old and new straight-line detectors: Description and comparison. *'PR 41*, 6 (June 2008), 1845–1866.

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

[FLW02] FATTAL R., LISCHINSKI D., WERMAN M.: Gradient domain high dynamic range compression. In *SIGGRAPH '02* (2002), pp. 249–256.

[GSCO06] GAL R., SORKINE O., COHEN-OR D.: Feature-aware texturing. In *Proceedings of Eurographics Symposium on Rendering* (2006), pp. 297–303.

[HSL*06] HUANG J., SHI X., LIU X., ZHOU K., WEI L.-Y., TENG S.-H., BAO H., GUO B., SHUM H.-Y.: Subspace gradient domain mesh deformation. *ACM Trans. Graph. 25*, 3 (2006), 1126–1134.

[KSSCO08] KRAEVOY V., SHEFFER A., SHAMIR A., COHEN-OR D.: Non-homogeneous resizing of complex models. *ACM Trans. Graph. 27*, 5 (2008), 1–9.

[Low04] LOWE D. G.: Distinctive image features from scale-invariant keypoints. *Int. J. Comput. Vision 60*, 2 (2004), 91–110.

[MGP06] MITRA N. J., GUIBAS L., PAULY M.: Partial and approximate symmetry detection for 3d geometry. *TOG. 25*, 3 (2006), 560–568.

[MGP07] MITRA N. J., GUIBAS L. J., PAULY M.: Symmetrization. In *SIGGRAPH '07: ACM SIGGRAPH 2007 papers* (2007), pp. 63–70.

[PGR07] PODOLAK J., GOLOVINSKIY A., RUSINKIEWICZ S.: Symmetry-enhanced remeshing of surfaces. In *SGP '07* (July 2007).

[PMW*08] PAULY M., MITRA N. J., WALLNER J., POTTMANN H., GUIBAS L.: Discovering structural regularity in 3D geometry. *ACM Transactions on Graphics 27*, 3 (2008), 1–11.

[PSG*06] PODOLAK J., SHILANE P., GOLOVINSKIY A., RUSINKIEWICZ S., FUNKHOUSER T.: A planar-reflective symmetry transform for 3D shapes. *TOG. 25*, 3 (July 2006).

[RSA08] RUBINSTEIN M., SHAMIR A., AVIDAN S.: Improved seam carving for video retargeting. In *SIGGRAPH '08: ACM SIGGRAPH 2008 papers* (2008), pp. 1–9.

[RSA09] RUBINSTEIN M., SHAMIR A., AVIDAN S.: Multi-operator media retargeting. *ACM Transactions on Graphics (Proceedings SIGGRAPH 2009) 28*, 3 (2009), to appear.

[SA07] SORKINE O., ALEXA M.: As-rigid-as-possible surface modeling. In *SGP '07* (2007), pp. 109–116.

[SA09] SHAMIR A., AVIDAN S.: Seam carving for media retargeting. *Commun. ACM 52*, 1 (2009), 77–85.

[SCSI08] SIMAKOV D., CASPI Y., SHECHTMAN E., IRANI M.: Summarizing visual data using bidirectional similarity. In *CVPR '08* (2008), pp. 1 – 8.

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

[Sor06] SORKINE O.: Differential representations for mesh processing. *Computer Graphics Forum 25*, 4 (December 2006), 789–807.

[SSP07] SUMNER R. W., SCHMID J., PAULY M.: Embedded deformation for shape manipulation. In *SIGGRAPH '07* (New York, NY, USA, 2007), ACM, p. 80.

[SZT*07] SHI X., ZHOU K., TONG Y., DESBRUN M., BAO H., GUO B.: Mesh puppetry: cascading optimization of mesh deformation with inverse kinematics. In *SIGGRAPH '07* (New York, NY, USA, 2007), ACM, p. 81.

[WGCO07] WOLF L., GUTTMANN M., COHEN OR D.: Non-homogeneous content-driven video-retargeting. In *ICCV07* (2007), pp. 1–6.

[WLT08] WANG Y.-S., LEE T.-Y., TAI C.-L.: Focus+context visualization with distortion minimization. *IEEE Transactions on Visualization and Computer Graphics 14*, 6 (2008), 1731–1738.

[WTSL08] WANG Y.-S., TAI C.-L., SORKINE O., LEE T.-Y.: Optimized scale-and-stretch for image resizing. *ACM Trans. Graph. 27*, 5 (2008), 1–8.

[YM09] YEH Y.-T., MECH R.: Detecting symmetries and curvilinear arrangements in vector art. *Computer Graphics Forum 28*, 2 (2009), 707–716.

[ZHM08] ZHANG Y.-F., HU S.-M., MARTIN R. R.: Shrinkability maps for content-aware video resizing. *Comput. Graph. Forum 27*, 7 (2008), 1797–1804.