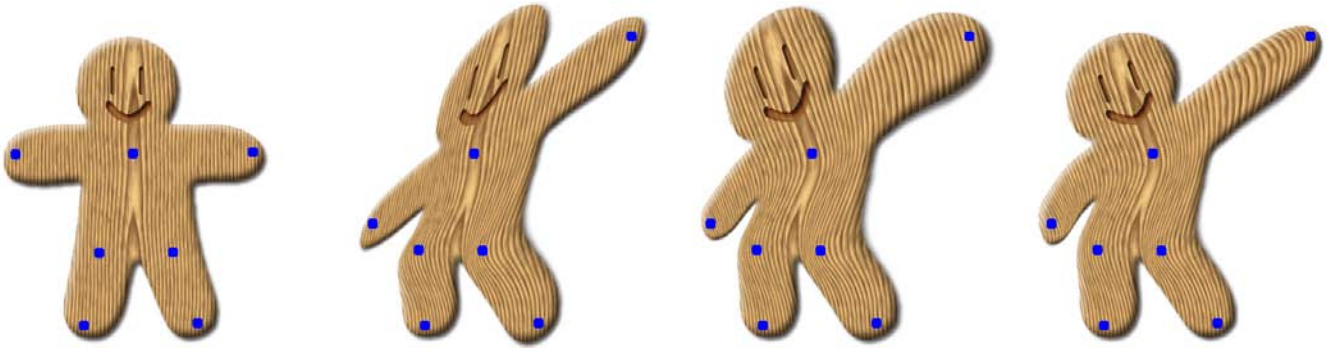
Image Deformation Using Moving Least Squares

Scott Schaefer* Texas A&M University
Travis McPhail† Rice University
Joe Warren‡ Rice University



(a) (b) (c) (d)

Figure 1: Deformation using Moving Least Squares. Original image with control points shown in blue (a). Moving Least Squares deformations using affine transformations (b), similarity transformations (c) and rigid transformations (d).

Abstract

We provide an image deformation method based on Moving Least Squares using various classes of linear functions including affine, similarity and rigid transformations. These deformations are realistic and give the user the impression of manipulating real-world objects. We also allow the user to specify the deformations using either sets of points or line segments, the later useful for controlling curves and profiles present in the image. For each of these techniques, we provide simple closed-form solutions that yield fast deformations, which can be performed in real-time.

CR Categories: I.3.5 [Computer Graphics]: Computational Geometry and Object Modeling—Boundary representations; Curve, surface, solid, and object representations; Geometric algorithms, languages, and systems

Keywords: Deformations, moving least squares, rigid transformations

1 Introduction

Image deformation has a number of uses from animation, to morphing [Smythe 1990] and medical imaging [Ju et al. 2003]. To perform these deformations the user selects some set of handles to control the deformation. These handles may take the form of points [Bookstein 1989], lines [Beier and Neely 1992], or even polygon grids [MacCracken and Joy 1996]. As the user modifies the position and orientation of these handles, the image should deform in an intuitive fashion.

We view this deformation as a function f that maps points in the undeformed image to the deformed image. Applying the function f to each point v in the undeformed image creates the deformed image. Now consider an image with a set of handles p that the user moves to new positions q. For f to be useful for deformations it must satisfy the following properties:

- *Interpolation*: The handles p should map directly to q under deformation. (i.e; $f(p_i) = q_i$).
- *Smoothness*: f should produce smooth deformations
- *Identity*: If the deformed handles q are the same as the p, then f should be the identity function. (i.e; $q_i = p_i \Rightarrow f(v) = v$).

These properties are very similar to those used in scattered data interpolation. The first two properties simply state that the function f interpolates the scattered data values and is smooth. The last property is sometimes referred to as linear precision in the approximation field. It states that if data is sampled from a linear function, then the interpolant reproduces that linear function. Given these similarities, it comes as no surprise that many deformation methods borrow techniques from scattered data interpolation.

Previous Work

Previous work on image deformation has focused on specifying deformations using different types of handles. Grid-based techniques such as free-form deformations [Sederberg and Parry 1986; Lee et al. 1995] parameterize the image using bivariate cubic splines to create C^2 deformations. Typically these methods require aligning grid lines corresponding to the control points of the spline with features of the image, which can be cumbersome for the user.

Beier et al. [Beier and Neely 1992] improve upon these grid-based techniques and allow the user to specify the deformation using sets of lines. This method is based on Shepard's interpolant [Shepard 1968] and creates smooth deformations. However, the authors note that their method produces complicated warps that

*email: sschaefe@rice.edu
†email:tjice@rice.edu
‡email:jwarren@rice.edu

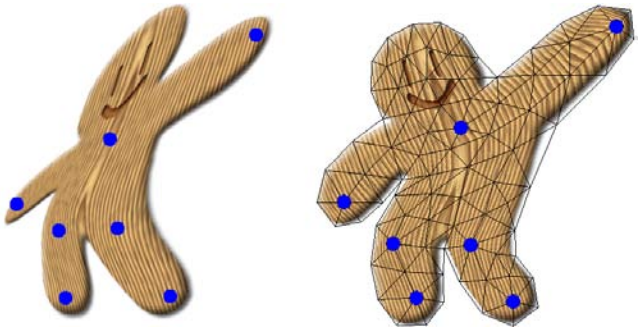


Figure 2: Deformation of the test shape from figure 1 using thin-plate splines (left). The deformation is smooth but lacks realism. On the right we use the method by Igarashi et al. shown with triangulation (right). The lack of smoothness is clearly visible in the wood grain.

can sometimes suffer from "ghosts", undesirable folding in the deformation. Koba et al. [Kobayashi and Ootsubo 2003] later generalized this technique to surface deformations.

Very few deformation methods investigate the type of transformations that are desirable for performing deformation. One notable exception is worked based on thin-plate splines [Bookstein 1989] that attempts to minimize the amount of bending in the deformation. Bookstein presents a deformation algorithm using the simplest deformation handle, a point, that uses radial basis functions with thin-plate splines. Figure 2 (left) shows an example of the deformation created with thin-plate splines for our example in figure 1. The deformation appears very similar to the affine-method in figure 1. In both cases, the test shape undergoes local non-uniform scaling and shearing, which is undesirable in many applications.

Our paper builds primarily on a recent paper by Igarashi et al. [Igarashi et al. 2005] that proposes a point-based image deformation technique for cartoon-like images in which the resulting deformations are as "rigid-as-possible". Such deformation have the property that amount of local scaling and shearing is minimized. (The concept of rigid-as-possible transformations was itself first introduced in Alexa [Alexa et al. 2000].)

To produce rigid-as-possible deformations, Igarashi et al. triangulate the input image and solve a linear system of equations whose size is equal to the number of vertices in the triangulation. In contrast, our method creates deformations by solving a small linear system (2×2) at each point in a uniform grid (see Section 4 for details). Since, we solve much smaller systems of equations, we can create very fast deformations of grids consisting of tens of thousands of vertices in real-time whereas Igarashi et al. report that their methods slows at 300 vertices on a 1 GHz machine. Due to the relatively small number of vertices, the deformations produced by Igarashi et al. may contain noticeable discontinuities as shown in figure 2. Figure 7 shows an equivalent deformation with our technique, which appears smooth.

Contributions

In this paper, we propose an image deformation method based on linear Moving Least Squares. To construct deformations that minimize the amount of local scaling and shear, we restrict the classes of transformations used in Moving Least Squares to similarity and rigid-body transformations. By using MLS, we avoid the need to triangulate the input image (as done in Igarashi et al.) and produce deformations that are globally smooth.)

Next, we derive closed-form formulas for both similarity and rigid MLS deformations. These formula are simple, easy to implement and provide real-time deformations. This derivation relies on a surprising and little-known relationship between similarity transformations and rigid transformations that minimize a common least squares problem. As opposed to Igarashi et al., our formulas do not require the use of a general linear solver.

As a natural extension of our point-based method, we extend our MLS deformation method from sets of points to sets of line segments and again provide closed-form expressions for the resulting deformation method.

2 Moving Least Squares Deformation

Here we consider building image deformations based on collections of points with which the user controls the deformation. Let p be a set of control points and q the deformed positions of the control points p. We construct a deformation function f satisfying the three properties outlined in the introduction using Moving Least Squares [Levin 1998]. Given a point v in the image, we solve for the best affine transformation $l_v(x)$ that minimizes

$$\sum_i w_i \left| l_v(p_i) - q_i \right|^2 \tag{1}$$

where p_i and q_i are row vectors and the weights w_i have the form

$$w_i = \frac{1}{|p_i - v|^{2\alpha}}.$$

Because the weights w_i in this least squares problem are dependent on the point of evaluation v, we call this a *Moving Least Squares* minimization. Therefore, we obtain a different transformation $l_v(x)$ for each v.

Now we define our deformation function f to be $f(v) = l_v(v)$. Observe that as v approaches p_i, w_i approaches infinity and the function f interpolates, (i.e; $f(p_i) = q_i$). Furthermore, if $q_i = p_i$, then each $l_v(x) = x$ for all x and, therefore, f is the identity transformation $f(v) = v$. Finally, this deformation function f has the property that it is smooth everywhere (except at the control points p_i when $\alpha \leq 1$).

Now since $l_v(x)$ is an affine transformation, $l_v(x)$ consists of two parts: a linear transformation matrix M and a translation T.

$$l_v(x) = xM + T \tag{2}$$

We can actually remove the translation T from this minimization problem further simplifying these equations. Equation 1 is quadratic in T. Since the minimizer is where the derivatives with respect to each of the free variables in $l_v(x)$ are zero, we can solve directly for T in terms of the matrix M. Taking the partial derivatives with respect to the free variables in T produces a linear system of equations. Solving for T yields that

$$T = q_* - p_* M$$

where p_* and q_* are weighted centroids.

$$\begin{aligned} p_* &= \frac{\sum_i w_i p_i}{\sum_i w_i} \\ q_* &= \frac{\sum_i w_i q_i}{\sum_i w_i} \end{aligned}$$

With this observation we can substitute T into equation 2 and rewrite $l_v(x)$ in terms of the linear matrix M.

$$l_v(x) = (x - p_*)M + q_* \tag{3}$$

Based on this insight, the least squares problem of equation 1 can be rewritten as

$$\sum_i w_i \left| \hat{p}_i M - \hat{q}_i \right|^2 \tag{4}$$

where $\hat{p}_i = p_i - p_*$ and $\hat{q}_i = q_i - q_*$. Notice that Moving Least Squares is very general in that the matrix M does not have to be a fully affine transformation. In fact, this framework allows us to investigate different classes of transformation matrices M. In particular, we are interested in the case where M is a rigid transformation. However, we first examine the case where M is an affine transformation as the derivation is the simplest. Next we construct deformations with similarity transformations and show how these solutions can be used to find closed-form solutions to Moving Least Square deformations with rigid transformations.

2.1 Affine Deformations

Finding an affine deformation that minimizes equation 4 is straightforward using the classic normal equations solution.

$$M = \left(\sum_i \hat{p}_i^T w_i \hat{p}_i \right)^{-1} \sum_j \hat{p}_j^T \hat{q}_j.$$

Though this solution requires the inversion of a matrix, the matrix is a constant size (2×2) and is fast to invert. With this closed-form solution for M we can write a simple expression for the deformation function $f_a(v)$.

$$f_a(v) = (v - p_*) \left(\sum_i \hat{p}_i^T w_i \hat{p}_i \right)^{-1} \sum_j \hat{p}_j^T \hat{q}_j + q_*. \qquad (5)$$

Applying this deformation function to each point in the image creates a new, deformed image.

While the user creates these deformations by manipulating the points q, the points p are fixed. Since the p do not change during deformation, much of equation 5 can be precomputed yielding very fast deformations. In particular, we can rewrite equation 5 in the form

$$f_a(v) = \sum_j A_j \hat{q}_j + q_*.$$

where A_j is a single scalar given by

$$A_j = (v - p_*) \left(\sum_i \hat{p}_i^T w_i \hat{p}_i \right)^{-1} \hat{p}_j^T.$$

Notice that, given a point v, everything in A_j can be precomputed yielding a simple, weighted sum. Table 1 provides timing results for the examples in this paper, which shows that these deformations may be performed over 500 times per second in our examples.

Figure 1 (b) illustrates this affine Moving Least Squares deformation applied to our test image. Unfortunately, the deformation does not appear very desirable due to the stretching in the arms and torso. These artifacts are created because affine transformations include deformations such as non-uniform scaling and shear. To eliminate these undesirable deformations we need to consider restricting the linear transformation $l_v(x)$. In particular, we modify the class of deformations $l_v(x)$ produces by restricting the transformation matrix M from being fully linear to similarity and rigid-body transformations.

2.2 Similarity Deformations

While affine transformations include effects such as non-uniform scaling and shear, many objects in reality do not undergo even these simple transformations. Similarity transformations are a special subset of affine transformations that only include translation, rotation and uniform scaling.

To alter our deformation technique to only use similarity transformations, we constrain the matrix M to have the property that $M^T M = \lambda^2 I$ for some scalar λ. If M is a block matrix of the form

$$M = \begin{pmatrix} M_1 & M_2 \end{pmatrix}$$

where M_1, M_2 are column vectors of length 2, then restricting M to be a similarity transform requires that $M_1^T M_1 = M_2^T M_2 = \lambda^2$ and $M_1^T M_2 = 0$. This constraint implies that $M_2 = M_1^\perp$ where $\perp$ is an operator on $2D$ vectors such that $(x,y)^\perp = (-y,x)$. Though restricted, the minimization problem from equation 4 is still quadratic in M_1 and can be rephrased as finding the column vector M_1 that minimizes

$$\sum_i w_i \left| \begin{pmatrix} \hat{p}_i \\ -\hat{p}_i^\perp \end{pmatrix} M_1 - \hat{q}_i^T \right|^2.$$

This quadratic function has a unique minimizer, which yields the optimal transformation matrix M

$$M = \frac{1}{\mu_s} \sum_i w_i \begin{pmatrix} \hat{p}_i \\ \hat{p}_i^\perp \end{pmatrix} \begin{pmatrix} \hat{q}_i^T & \hat{q}_i^{\perp T} \end{pmatrix} \qquad (6)$$

where

$$\mu_s = \sum_i w_i \hat{p}_i \hat{p}_i^T.$$

Similar to the affine deformations, the user manipulates the q to produce the deformation while the p remain fixed. Using this observation we write the deformation function $f_s(v)$ in a form that allows us to precompute as much information as possible. $f_s(v)$ is then

$$f_s(v) = \sum_i \hat{q}_i \left(\frac{1}{\mu_s} A_i \right) + q_*$$

where μ_s and A_i depend only on the p_i, v and can be precomputed and A_i is

$$A_i = w_i \begin{pmatrix} \hat{p}_i \\ -\hat{p}_i^\perp \end{pmatrix} \begin{pmatrix} v - p_* \\ -(v - p_*)^\perp \end{pmatrix}^T. \qquad (7)$$

As expected, similarity MLS deformations preserves angles in the original image better than affine MLS deformations. (Transformations that strictly preserve angle are called conformal transformations and have been studied extensively in [Gu and Yau 2003].) While approximate (or exact) angle preservation is a desirable property in many cases, allowing local scaling can often lead to undesirable deformations. Figure 1 (c) shows an example of applying the similarity Moving Least Squares deformation to our test image. The result is a much more realistic looking deformation than (b). However, this deformation scales the size of the upper arm as it is stretched. To remove this scaling, we consider building deformations using only rigid transformations.

2.3 Rigid Deformations

Recently, several works [Alexa et al. 2000; Igarashi et al. 2005] have shown that, for realistic shapes, deformations should be as rigid as possible; that is, the space of deformations should not even include uniform scaling. Traditionally researchers in deformation have been reluctant to approach this problem directly due to the non-linear constraint that $M^T M = I$. However, we note that closed-form solutions to this problem are known from the Iterated Closest Point community [Horn 1987]. Horn shows that the optimal rigid transformation can be found in terms of eigenvalues and eigenvectors of a covariance matrix involving the points p_i and q_i. We show that these rigid deformations are related to the similarity deformations from section 2.2 via the following theorem.

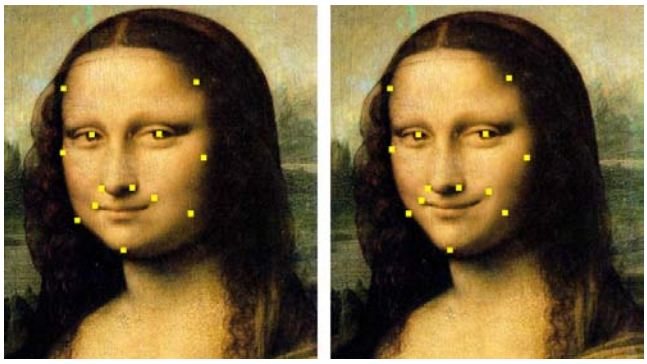


Figure 3: Original image (left) and its deformation using the rigid MLS method (right). After deformation, the face is thinner and she is smiling.

Theorem 2.1 *Let C be the matrix that minimizes the following similarity functional*

$$\min_{M^T M=\lambda^2 I} \sum_i w_i \left|\hat{p}_i M - \hat{q}_i\right|^2.$$

If C is written in the form λR where R is a rotation matrix and λ is a scalar, the rotation matrix R minimizes the rigid functional

$$\min_{M^T M=I} \sum_i w_i \left|\hat{p}_i M - \hat{q}_i\right|^2.$$

Proof: See Appendix A.

This theorem is valid in arbitrary dimension, however, it is very easy to apply in $2D$. Using this theorem, we find that the rigid transformation is exactly the same as equation 6 except that we use a different constant μ_r in the solution so that $M^T M = I$ given by

$$\mu_r = \sqrt{\left(\sum_i w_i \hat{q}_i \hat{p}_i^T\right)^2 + \left(\sum_i w_i \hat{q}_i \hat{p}_i^{\perp T}\right)^2}.$$

Unlike the similarity deformation $f_s(v)$, we cannot precompute as much information for the rigid deformation function $f_r(v)$. However, the deformation process can still be made very efficient. Let

$$\vec{f}_r(v) = \sum_i \hat{q}_i A_i$$

where A_i is defined in equation 7, which may be precomputed. This vector $\vec{f}_r(v)$ is a rotated and scaled version of the vector $v - p_*$. To compute $f_r(v)$ we normalize $\vec{f}_r$, scale by the length of $v - p_*$ (which also can be precomputed), and translate by q_*.

$$f_r(v) = |v - p_*| \frac{\vec{f}_r(v)}{|\vec{f}_r(v)|} + q_*. \tag{8}$$

This method is slower than the similarity deformation due to the normalization; however, these deformations are still very fast as shown in table 1.

Figure 1 (d) shows this rigid deform applied to the test image in (a). As opposed to the other methods, this deformation is quite realistic and almost feels as if the user is manipulating a real object. Figures 3 and 4 show additional examples of this rigid deformation method. In the figure with the Mona Lisa, we deform the image to create a thinner facial profile and make her smile. In the figure with the horse, we stretch the horses legs and neck to create a giraffe. Due to the use of rigid transformations, the deformation maintains rigidity and scale locally so that the body and head of the horse retain their relative shape.

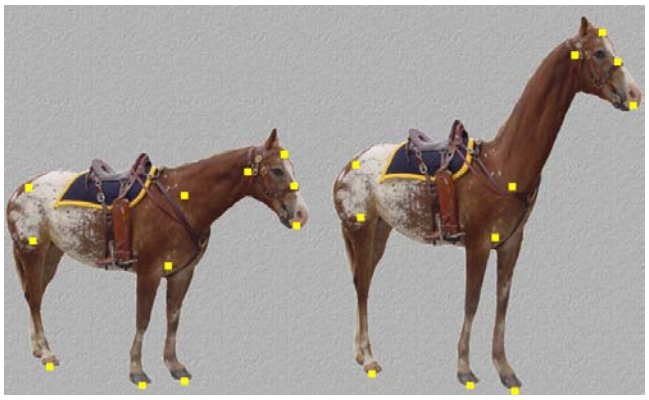


Figure 4: Original image (left) and its deformation using the rigid MLS method (right).

3 Deformation with Line Segments

So far we have considered creating deformations with Moving Least Squares using only sets of points to control the deformation. In applications where precise control over curves such as profiles in the image is needed, points may be insufficient for specifying these deformations. One solution that allows the user to control curves precisely is to convert these curves to dense sets of points and apply a point-based deformation [Wolberg 1998]. The disadvantage of this approach is that the computation time of the deformation is proportional to the number of control points used and creating large numbers of control points adversely affects performance.

Alternatively, we desire a generalization of these Moving Least Squares deformations from section 2 to arbitrary curves in the plane. First, assume $p_i(t)$ is the i^{th} control curve and $q_i(t)$ is the deformed curve corresponding to $p_i(t)$. We generalize the quadratic function in equation 1 by integrating over each control curve $p_i(t)$ where we assume $t \in [0,1]$.

$$\sum_i \int_0^1 w_i(t) \left|p_i(t) M + T - q_i(t)\right|^2 \tag{9}$$

where $w_i(t)$ is

$$w_i(t) = \frac{|p_i'(t)|}{|p_i(t) - v|^{2\alpha}}$$

and $p_t'(t)$ is the derivative of $p_i(t)$. (This factor of $|p'(t)|$ makes the integrals independent of the parameterization of the curve $p_i(t)$.) Now notice that, despite the integral, equation 9 is still quadratic in T and can be solved for in terms of the matrix M.

$$T = q_* - p_* M$$

where p_* and q_* are again weighted centroids.

$$\begin{aligned} p_* &= \frac{\sum_i \int_0^1 w_i(t) p_i(t) dt}{\sum_i \int_0^1 w_i(t) dt} \\ q_* &= \frac{\sum_i \int_0^1 w_i(t) q_i(t) dt}{\sum_i \int_0^1 w_i(t) dt} \end{aligned} \tag{10}$$

Therefore, we rewrite equation 9 only in terms of M as

$$\sum_i \int_0^1 w_i(t) \left|\hat{p}_i(t) M - \hat{q}_i(t)\right|^2 \tag{11}$$

where

$$\begin{aligned} \hat{p}_i(t) &= p_i(t) - p_* \\ \hat{q}_i(t) &= q_i(t) - q_*. \end{aligned}$$

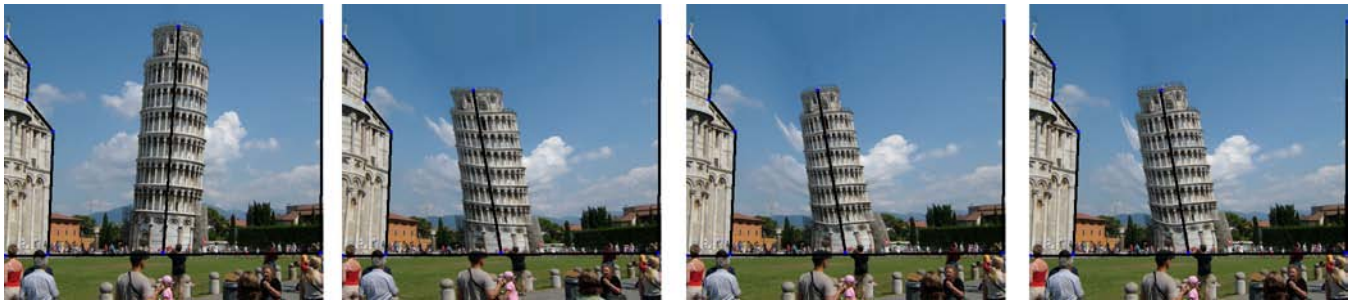


Figure 5: Deformation of the Leaning Tower of Pisa. From left to right: original image, Affine MLS, Similarity MLS and Rigid MLS deformations.

Until now, $p_i(t)$ and $q_i(t)$ have been arbitrary curves. However, the integrals in equation 11 may be difficult to evaluate for arbitrary functions. Instead, we restrict these functions to be line segments and derive closed-form solutions for the deformations in terms of the end-points of these segments. Similar to section 2, we first consider affine transformations due to its relatively simple derivation and then move to similarity transformations, which we use to create closed-form solutions to the equivalent problem using rigid-body transformations.

3.1 Affine Lines

Since $\hat{p}_i(t)$, $\hat{q}_i(t)$ are line segments, we can represent these curves as matrix products

$$\hat{p}_i(t) = \begin{pmatrix} 1-t & t \end{pmatrix} \begin{pmatrix} \hat{a}_i \\ \hat{b}_i \end{pmatrix}$$
$$\hat{q}_i(t) = \begin{pmatrix} 1-t & t \end{pmatrix} \begin{pmatrix} \hat{c}_i \\ \hat{d}_i \end{pmatrix}$$

where $\hat{a}_i$, $\hat{b}_i$ are the end-points of $\hat{p}_i(t)$ and $\hat{c}_i$, $\hat{d}_i$ are the end-points of $\hat{q}_i(t)$. Equation 11 is then written as

$$\sum_i \int_0^1 \left| \begin{pmatrix} 1-t & t \end{pmatrix} \left(\begin{pmatrix} \hat{a}_i \\ \hat{b}_i \end{pmatrix} M - \begin{pmatrix} \hat{c}_i \\ \hat{d}_i \end{pmatrix} \right) \right|^2 \qquad (12)$$

whose minimizer is

$$M = \left(\sum_i \begin{pmatrix} \hat{a}_i \\ \hat{b}_i \end{pmatrix}^T W_i \begin{pmatrix} \hat{a}_i \\ \hat{b}_i \end{pmatrix} \right)^{-1} \sum_j \begin{pmatrix} \hat{a}_j \\ \hat{b}_j \end{pmatrix}^T W_j \begin{pmatrix} \hat{c}_j \\ \hat{d}_j \end{pmatrix}$$

where W_i is a weight matrix given by

$$W_i = \begin{pmatrix} \delta i^{00} & \delta_i^{01} \\ \delta_i^{01} & \delta_i^{11} \end{pmatrix}$$

and the δ_i are integrals of the weight function $w_i(t)$ multiplied by the different quadratic polynomials.

$$\begin{aligned} \delta_i^{00} &= \int_0^1 w_i(t)(1-t)^2 dt \\ \delta_i^{01} &= \int_0^1 w_i(t)(1-t)t\,dt \\ \delta_i^{11} &= \int_0^1 w_i(t)t^2 dt \end{aligned}$$

These integrals have closed-form solutions for various values of α. In appendix B we provide a closed-form solution for $\alpha = 2$ though other solutions can be computed with the aid of a symbolic integration package. Note that these integrals can also be used to evaluate p_* and q_* from equation 10.

$$\begin{aligned} p_* &= \frac{\sum_i a_i(\delta_i^{00}+\delta_i^{01}) + b_i(\delta_i^{01}+\delta_i^{11})}{\sum_i \delta_i^{00} + 2\delta_i^{01} + \delta_i^{11}} \\ q_* &= \frac{\sum_i c_i(\delta_i^{00}+\delta_i^{01}) + d_i(\delta_i^{01}+\delta_i^{11})}{\sum_i \delta_i^{00} + 2\delta_i^{01} + \delta_i^{11}} \end{aligned}$$

As before, we write the deformation function $f_a(v)$ as

$$f_a(v) = \sum_j A_j \begin{pmatrix} \hat{c}_j \\ \hat{d}_j \end{pmatrix} + q_*$$

where A_j is a 1×2 matrix of the form

$$A_j = (v - p_*) \left(\sum_i \begin{pmatrix} \hat{a}_i \\ \hat{b}_i \end{pmatrix}^T W_i \begin{pmatrix} \hat{a}_i \\ \hat{b}_i \end{pmatrix} \right)^{-1} \begin{pmatrix} \hat{a}_j \\ \hat{b}_j \end{pmatrix}^T W_j.$$

During the deformation, the end-points a_i and b_i of the line segment $p_i(t)$ are fixed while the user manipulates the end-points c_i and d_i of the line segments $q_i(t)$. Since A_j is independent of c_i and d_i, A_j can be precomputed.

Figure 5 shows an example deformation performed with line segments where we modify the Leaning Tower of Pisa to lean the opposite direction and shrink the tower. The Affine MLS deformation shears the tower to the side instead of being rotated and does not appear to be realistic. To remove this shear effect, we restrict the matrix in equation 11 to be a similarity or rigid-body transformation.

3.2 Similarity Lines

Restricting equation 12 to similarity transforms requires that $M^T M = \lambda^2 I$ for some scalar λ. As noted in section 2.2, M can be parameterized using a single column vector M_1 yielding

$$\sum_i \int_0^1 \left| \begin{pmatrix} 1-t & 0 & t & 0 \\ 0 & 1-t & 0 & t \end{pmatrix} \left(\begin{pmatrix} \hat{a}_i \\ -\hat{a}_i^\perp \\ \hat{b}_i \\ -\hat{b}_i^\perp \end{pmatrix} M_1 - \begin{pmatrix} \hat{c}_i^T \\ \hat{d}_i^T \end{pmatrix} \right) \right|^2$$

This error function is quadratic in M_1. To find the minimizer, we differentiate with respect to the free variables in M_1 and solve the linear system of equations to obtain the matrix M.

$$M = \frac{1}{\mu_s} \sum_j \begin{pmatrix} \hat{a}_j \\ -\hat{a}_j^\perp \\ \hat{b}_j \\ -\hat{b}_j^\perp \end{pmatrix}^T W_j \begin{pmatrix} \hat{c}_j^T & \hat{c}_j^{\perp T} \\ \hat{d}_j^T & \hat{d}_j^{\perp T} \end{pmatrix} \qquad (13)$$

where W_j is a weight matrix

$$W_j = \begin{pmatrix} \delta_j^{00} & 0 & \delta_j^{01} & 0 \\ 0 & \delta_j^{00} & 0 & \delta_j^{01} \\ \delta_j^{01} & 0 & \delta_j^{11} & 0 \\ 0 & \delta_j^{01} & 0 & \delta_j^{11} \end{pmatrix}$$

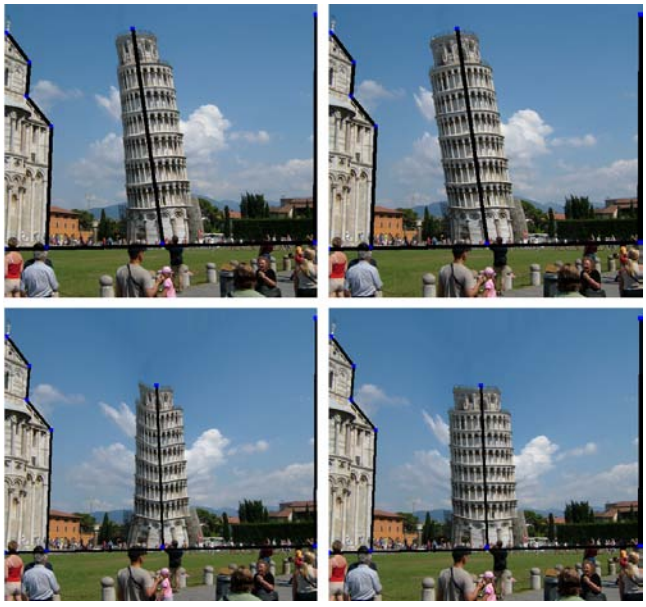


Figure 6: Comparison of the line deformation method of Beier et al. (left) with the Rigid MLS deformation (right).

and μ_s is again a scaling constant, which has the form

$$\mu_s = \sum_i \hat{a}_i \hat{a}_i^T \delta_i^{00} + 2\hat{a}_i \hat{b}_i^T \delta_i^{01} + \hat{b}_i \hat{b}_i^T \delta_i^{11}.$$

This deformation function has a very similar structure to the point-based similarity deformation. Using this matrix we write $f_s(v)$ explicitly as

$$f_s(v) = \sum_j \begin{pmatrix} \hat{c}_j & \hat{d}_j \end{pmatrix} \left(\frac{1}{\mu_s} A_j\right) + q_*$$

where A_j is a 4×2 matrix.

$$A_j = W_j \begin{pmatrix} \hat{a}_j \\ -\hat{a}_j^\perp \\ \hat{b}_j \\ -\hat{b}_j^\perp \end{pmatrix} \begin{pmatrix} v - p_* \\ -(v-p_*)^\perp \end{pmatrix}^T \qquad (14)$$

Figure 5 shows the tower deformed using this similarity-based method. In contrast to the affine method, the tower actually appears to be rotated, not sheared, to the left resulting in a more realistic deformation. Similarity transformations contain uniform scaling, which is apparent from the way in which the tower shrinks with the line segment. Rigid transformations remove this uniform scaling.

3.3 Rigid Lines

Using the solution from section 3.2 and Theorem 2.1, we immediately have a closed form solution for rigid-body transformations. The transformation matrix is, therefore, the same as equation 13 except we choose a different scaling constant μ_r so that $M^T M = I$.

$$\mu_r = \left| \sum_j \begin{pmatrix} \hat{a}_j^T & -\hat{a}_j^{\perp T} & \hat{b}_j^T & -\hat{b}_j^{\perp T} \end{pmatrix} W_j \begin{pmatrix} \hat{c}_j^T \\ \hat{d}_j^T \end{pmatrix} \right|$$

This deformation is non-linear, but we can compute it in a simple fashion using equation 8. This equation uses the rotated vector

Method	Figure 1 (7 points)	Figure 4 (11 points)	Figure 5 (7 lines)
Affine MLS	1.5 ms	2.2 ms	1.5 ms
Similarity MLS	2.3 ms	3.4 ms	1.6 ms
Rigid MLS	2.6 ms	3.8 ms	3.3 ms
[Bookstein 1989]	2 ms	2.7 ms	N/A
[Beier and Neely 1992]	N/A	N/A	1.6ms

Table 1: Deformation times for the various methods.

$\vec{f}_r(v)$, scales the vector so that its length is $|v - p_*|$ and translates by q_*. For this deformation using line segments, the rotated vector is given by

$$\vec{f}_r(v) = \sum_j \begin{pmatrix} \hat{c}_j & \hat{d}_j \end{pmatrix} A_j$$

where A_j is from equation 14.

Figure 5 (right) shows a deformation of the tower using this rigid method. In this deformation, the tower is rotated but does not shrink as the similarity deformation does. Instead the effect is almost the same as non-uniform scaling along the direction of the line segment.

Figure 6 also shows a comparison of the rigid deformation technique (right) with the line deformation method of Beier et al. [Beier and Neely 1992] (left). The warps created with Beier et al.'s method fold and pull in unrealistic ways whereas the rigid method does not suffer from these same defects.

4 Implementation

To implement these deformations, we precompute as much information as possible for the deformation functions $f(v)$. When we apply the deformation to an image, we typically do not apply $f(v)$ to every pixel in the image. Instead we approximate the image with a grid and apply the deformation function to each vertex in the grid. We then fill the resulting quads using bilinear interpolation (see figure 7).

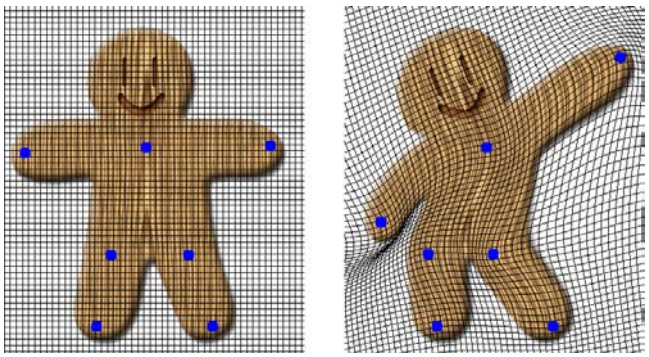


Figure 7: Deforming an image with a uniform grid (50×50). Original image (left) and rigid MLS deformation (right) using bilinear interpolation in each quad.

In practice, this approximation technique produces deformations indistinguishable from the more expensive process of applying the deformation to every pixel in the image. For all of the examples in this paper, the images were approximately 500×500 pixels. To compute the deformations, we used grids on the order of 100×100 vertices. If desired, more accurate deformations may be achieved with denser grids and the deformation time is linear in the number of vertices of these grids.

Table 1 shows the amount of time taken to deform each of the images using various methods on a 3 GHz Intel machine. Each deformation uses a grid of size 100×100. The rigid transformations take the longest due to the square root in the deformation function, but are still quite fast.

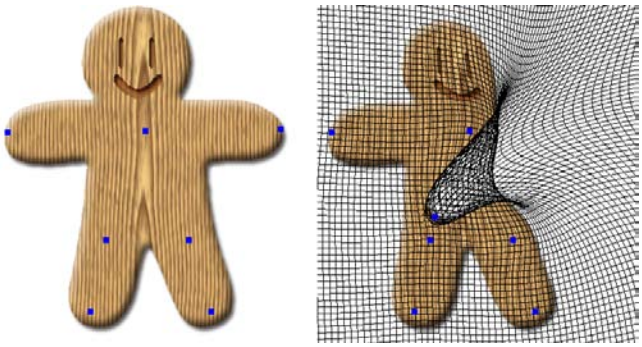


Figure 8: Foldback caused during deformations.

5 Conclusions and Future Work

We have provided a method for creating smooth deformations of images using either points or lines as handles to control the deformation. Using Moving Least Squares we created deformations using affine, similarity and rigid transformations while providing closed-form expressions for each of these techniques. Though the least squares minimization with rigid transformations led to a non-linear minimization, we showed how these solutions could be computed directly from the closed-form deformation using similarity transformations thereby bypassing the non-linear minimization.

In terms of limitations, our method may suffer from fold-backs like most other space warping approaches. These situations occur when the sign of the Jacobian of f changes. For many deformations, these fold backs may not be noticeable though extreme deformations will certainly cause such fold-backs to happen (see figure 8). For some deformations, fold-backs are acceptable since these 2D images are meant to represent 3D objects. Igarashi et al. take advantage of the explicit topology of the image and provide a simple method for rendering these deformations. Our lack of topology makes this technique difficult though topological information may be added to our method.

In other applications, fold-backs are not desirable and must be eliminated. There is a generic approach available for fixing these fold-backs provided by Tiddeman et al. [Tiddeman et al. 2001]. Given a warp, Tiddeman et al. create a subsequent warp such that the product of the two warps results in a non-negative Jacobian. Since we provide simple equations for our deformations, we intend to explore the possibility of constructing closed-formed formulas for the Jacobian for use with Tiddeman et al.'s method.

Our warping technique also deforms the entire plane that the image lies in without regard to the topology of the shape in the image. This lack of topology is both a benefit and a limitation. One of the advantages of our approach is the lack of such topology, which creates a simple warping function. Other techniques such as Igarashi et al. [Igarashi et al. 2005] construct triangulations that outline the boundary of the shape and build deformations dependent on the specified topology. This topological information can create better deformations by separating parts of the images such as the legs of the horse in figure 4 that are geometrically close together. Notice that our method is general enough to accommodate different distance metrics dependent on the topology of the shape rather than the simple, Euclidean distance used as our weight factor. We intend to explore this issue in future work.

Finally, in the future we would like to explore generalizing these deformation methods to 3*D* to deform surfaces. Such a generalization has potential applications in the motion capture field where animation data can take the form of points in space for each frame of animation. However, the similarity transformation in section 2.2 no longer leads to a quadratic minimization, but an eigenvector problem and we are looking into methods to efficiently compute the solution to this minimization.

References

ALEXA, M., COHEN-OR, D., AND LEVIN, D. 2000. As-rigid-as-possible shape interpolation. In *Proceedings of ACM SIGGRAPH 2000*, ACM Press/Addison-Wesley Publishing Co., New York, NY, USA, 157–164.

BEIER, T., AND NEELY, S. 1992. Feature-based image metamorphosis. In *SIGGRAPH '92: Proceedings of the 19th annual conference on Computer graphics and interactive techniques*, ACM Press, New York, NY, USA, 35–42.

BOOKSTEIN, F. L. 1989. Principal warps: Thin-plate splines and the decomposition of deformations. *IEEE Trans. Pattern Anal. Mach. Intell. 11*, 6, 567–585.

GU, X., AND YAU, S.-T. 2003. Global conformal surface parameterization. In *SGP '03: Proceedings of the 2003 Eurographics/ACM SIGGRAPH symposium on Geometry processing*, Eurographics Association, Aire-la-Ville, Switzerland, Switzerland, 127–137.

HORN, B. 1987. Closed-form solution of absolute orientation using unit quaternions. *Journal of the Optical Society of America A 4*, 4 (April), 629–642.

IGARASHI, T., MOSCOVICH, T., AND HUGHES, J. F. 2005. As-rigid-as-possible shape manipulation. *ACM Trans. Graph. 24*, 3, 1134–1141.

JU, T., WARREN, J., EICHELE, G., THALLER, C., CHIU, W., AND CARSON, J. 2003. A geometric database for gene expression data. In *SGP '03: Proceedings of the 2003 Eurographics/ACM SIGGRAPH symposium on Geometry processing*, 166–176.

KOBAYASHI, K. G., AND OOTSUBO, K. 2003. t-ffd: free-form deformation by using triangular mesh. In *SM '03: Proceedings of the eighth ACM symposium on Solid modeling and applications*, ACM Press, 226–234.

LEE, S.-Y., CHWA, K.-Y., AND SHIN, S. Y. 1995. Image metamorphosis using snakes and free-form deformations. In *SIGGRAPH '95: Proceedings of the 22nd annual conference on Computer graphics and interactive techniques*, ACM Press, New York, NY, USA, 439–448.

LEVIN, D. 1998. The approximation power of moving least-squares. *Mathematics of Computation 67*, 224, 1517–1531.

MACCRACKEN, R., AND JOY, K. I. 1996. Free-form deformations with lattices of arbitrary topology. In *Proceedings of ACM SIGGRAPH 1996*, ACM Press, 181–188.

SEDERBERG, T. W., AND PARRY, S. R. 1986. Free-form deformation of solid geometric models. In *Proceedings of ACM SIGGRAPH 1986*, ACM Press, 151–160.

SHEPARD, D. 1968. A two-dimensional interpolation function for irregularly-spaced data. In *Proceedings of the 1968 23rd ACM national conference*, ACM Press, 517–524.

SMYTHE, D. 1990. A two-pass mesh warping algorithm for object transformation and image interpolation. Tech. Rep. 1030, ILM Computer Graphics Department, Lucasfilm, San Rafael, Calif.

TIDDEMAN, B., DUFFY, N., AND RABEY, G. 2001. A general method for overlap control in image warping. *Computers and Graphics 25*, 1, 59–66.

WOLBERG, G. 1998. Image morphing: a survey. *The Visual Computer 14*, 8/9, 360–372.

A Appendix

Here we provide a proof of Theorem 2.1.

Theorem 2.1 *Let C be the matrix that minimizes the following similarity functional*

$$\min_{M^TM=\lambda^2 I} \sum_i w_i \left|\hat{p}_i M - \hat{q}_i\right|^2.$$

If C is written in the form λR where R is a rotation matrix and λ is a scalar, the rotation matrix R minimizes the rigid functional

$$\min_{M^TM=I} \sum_i w_i \left|\hat{p}_i M - \hat{q}_i\right|^2.$$

Proof: First, we expand both of the above error functions into their quadratic forms yielding

$$\begin{array}{c} \min_{R^TR=I,\lambda} \sum_i w_i \left(\lambda^2 \hat{p}_i \hat{p}_i^T - 2\lambda \hat{p}_i R \hat{q}_i^T + \hat{q}_i \hat{q}_i^T\right) \\ \min_{R^TR=I} \sum_i w_i \left(\hat{p}_i \hat{p}_i^T - 2\hat{p}_i M \hat{q}_i^T + \hat{q}_i \hat{q}_i^T\right) \end{array}$$

These minimization problems are very similar. We find the matrices that minimize these error functions by differentiating the functions with respect to the free variables θ_j in R.

$$\begin{array}{c} \sum_i w_i \left(-2\lambda \hat{p}_i \frac{\partial R}{\partial \theta_j} \hat{q}_i^T\right) = 0 \\ \sum_i w_i \left(-2\hat{p}_i \frac{\partial R}{\partial \theta_j} \hat{q}_i^T\right) = 0 \end{array}$$

Now, unless $\lambda = 0$, which implies a degenerate transformation, these equations are equal. Since $C = \lambda R$, this implies that $\pm R$ minimizes the quadratic function using rigid transformations. The negative solution corresponds to a maximum while the positive solution is the minimum. QED

B Appendix

In section 3 we derive closed-form solutions for Moving Least Squares deformations using line segments. In order to complete the derivation, we need closed-form solutions for integrals of three quadratic polynomials times the weight function $w_i(t)$ over the line segments. Let a_i, b_i be the endpoints of the line segment described by $p_i(t)$ and let

$$\begin{array}{rcl} \Delta_i & = & (a_i - v)^{\perp}(a_i - b_i)^T \\ \theta_i & = & \tan^{-1}\left(\frac{(b_i-v)(b_i-a_i)^T}{(b_i-v)^{\perp}(b_i-a_i)^T}\right) - \tan^{-1}\left(\frac{(a_i-v)(a_i-b_i)^T}{(a_i-v)^{\perp}(a_i-b_i)^T}\right) \\ \beta_i^{00} & = & (a_i - v)(a_i - v)^T \\ \beta_i^{01} & = & (a_i - v)(v - b_i)^T \\ \beta_i^{11} & = & (v - b_i)(v - b_i)^T. \end{array}$$

The integrals then have the closed-form solution

$$\begin{array}{rcl} \int_0^1 w_i(t)(1-t)^2 dt & = & \frac{|a_i-b_i|}{2\Delta_i^2}\left(\frac{\beta_i^{01}}{\beta_i^{00}} - \frac{\beta_i^{11}\theta_i}{\Delta_i}\right) \\ \int_0^1 w_i(t)t(1-t) dt & = & \frac{|a_i-b_i|}{2\Delta_i^2}\left(1 - \frac{\beta_i^{01}\theta_i}{\Delta_i}\right) \\ \int_0^1 w_i(t)t^2 dt & = & \frac{|a_i-b_i|}{2\Delta_i^2}\left(\frac{\beta_i^{01}}{\beta_i^{11}} - \frac{\beta_i^{00}\theta_i}{\Delta_i}\right). \end{array}$$

When v is on the line segment defined by a_i and b_i, these integrals do not need to be evaluated because the function $f(v)$ interpolates the line segments. However, if v is on the extension of one of these line segments, $\Delta_i = 0$ and these integrals reduce to

$$\begin{array}{rcl} \int_0^1 w_i(t)(1-t)^2 dt & = & \frac{|a_i-b_i|^5}{3((v-b_i)(b_i-a_i)^T)((a_i-v)(b_i-a_i)^T)^3} \\ \int_0^1 w_i(t)t(1-t) dt & = & \frac{-|a_i-b_i|^5}{6((v-b_i)(b_i-a_i)^T)^2((a_i-v)(b_i-a_i)^T)^2} \\ \int_0^1 w_i(t)t^2 dt & = & \frac{|a_i-b_i|^5}{3((v-b_i)(b_i-a_i)^T)^3((a_i-v)(b_i-a_i)^T)}. \end{array}$$