

# Image Metamorphosis By Affine Transformations

Tim Myers and Peter Spiegel

December 16, 2005

## Abstract

Among the many ways to manipulate an image is a technique known as *morphing*. Image morphing is a special effect that transforms or *morphs* one image into another. In this paper we show how this can be done using affine transformations.

## Introduction

Traditionally morphing would involve cross-fading the images. That is, the first image would be displayed and then would begin to fade out while the second image is fading in simultaneously. The end result would be that of the second image. Since the early 1990's, however, this technique has become almost obsolete. Many new techniques for image morphing can produce more realistic looking results than the old cross-fading technique did. Affine transformations is an excellent example of one of the new techniques.

[Home Page](#)[Title Page](#)[Page 1 of 32](#)[Go Back](#)[Full Screen](#)[Close](#)[Quit](#)

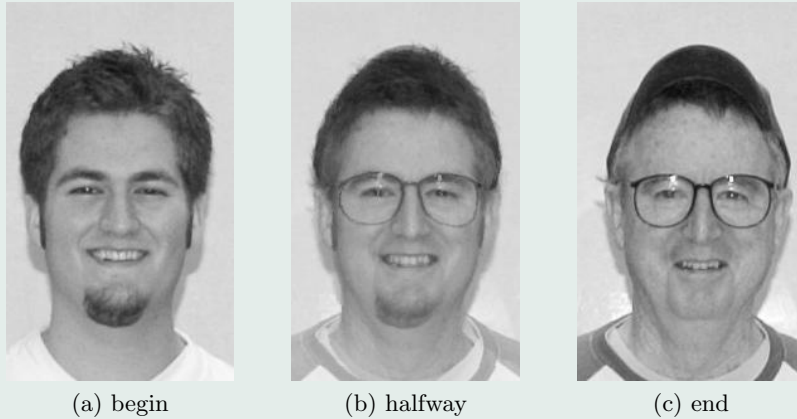


Figure 1: Morph by Affine Transformations

## Objective

Using affine transformations, the *begin image* and *end image* would be triangulated similarly and each triangle from the begin image would be transformed into the corresponding triangle in the end image through matrix multiplication. The color density would then be mapped to its new value. After the image is successfully warped from the begin image to the end image the steps are reversed



and a weighted average of each frame is taken to get a more realistic result for each transition frame of the image. The result is a smooth transition that can be shown frame by frame. This approach provides an advantage over many other morphing routines in that each frame of the morph produces a realistic looking image.

## Triangulation

The images are divided up by selecting vertex points in each image. In our program the vertex points are free to be chosen by the user with a simple mouse click. The program assumes the four corners of each image are chosen and the number of vertex points can be easily modified to accommodate a large number of triangles. We begin by sending the vertex points in the begin image into Matlab's built-in command *delaunay* to obtain a triangulation of the image. We then use the same triangulation on the end image to form similar triangles. When the whole picture is broken up into triangles each triangle can be dealt with individually. To keep things as simple as possible, we will explain the process for one triangle.

Let us begin by describing a simple warp of a triangle in  $\mathbb{R}^2$ . Let  $v_1, v_2$ , and  $v_3$  be the vertices of a given triangle. We impose the restriction that the vertex points be noncollinear. We will refer to this triangle as the ***begin-triangle*** (Figure 2a). Let us now establish a second triangle with vertices  $w_1, w_2$ , and  $w_3$  which we will call the ***end-triangle*** (Figure 2b). We need to map each vertex point in the begin-triangle to its new location in the end-triangle, where  $v_1 \mapsto w_1, v_2 \mapsto w_2$ , and  $v_3 \mapsto w_3$ . This can be done easily using linear transformations.

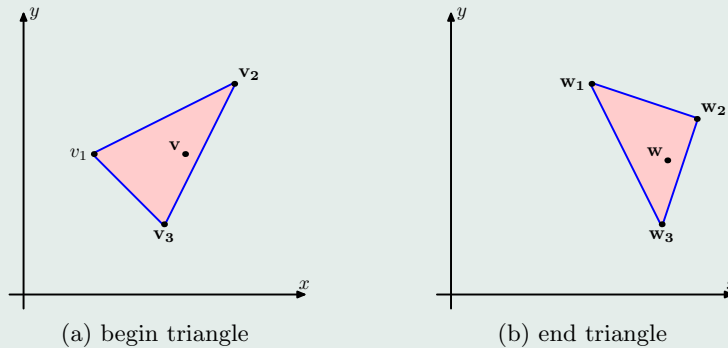


Figure 2: Vertex Point Displacement

However, because we need to obtain several frames between the begin and end images we chose to map the vertex points along a straight line path. This method clearly defines the location of each point at each frame. We need to find the location at certain times  $t$  between 0 and 1, where 0 is the position in the begin image and 1 is the position in the end image. For each vertex point  $v_i \mapsto w_i$  where  $\{i = 1, 2, 3\}$  the location can be found by

$$u_i(t) = (1 - t)v_i + tw_i.$$

Once the vertex positions of a given triangle are known, for each individual

[Home Page](#)[Title Page](#)[◀◀](#) [▶▶](#)[◀](#) [▶](#)[Page 4 of 32](#)[Go Back](#)[Full Screen](#)[Close](#)[Quit](#)

frame of the image, we can transform the shape of the triangle using a linear transformation.

## Linear Transformations

The product of a 2-by-2 matrix and a vector in  $\mathbb{R}^2$  can yield very desirable results for transforming triangles. This 2-by-2 matrix we will refer to as the *standard matrix*. If the standard matrix is

$$A = \begin{bmatrix} a & b \\ c & d \end{bmatrix},$$

then

$$T\left(\begin{bmatrix} x \\ y \end{bmatrix}\right) = \begin{bmatrix} a & b \\ c & d \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} ax + by \\ cx + dy \end{bmatrix}.$$

There is a number of different effects these linear operations can produce, namely, reflections, rotations, compressions, expansions and shears. When discussing the different types of operations we will observe the effect of each on the unit-square (Figure 3).

Note that the standard matrix multiplies every vector in the square. Therefore it changes the shape of the square. Reflections and rotations are two of the effects a standard matrix can have on a vector (Figure 4). The matrix

$$A = \begin{bmatrix} -1 & 0 \\ 0 & 1 \end{bmatrix}$$

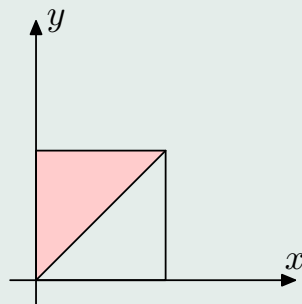


Figure 3: Unit Square

[Home Page](#)[Title Page](#)[◀](#) [▶](#)[◀](#) [▶](#)

Page 6 of 32

[Go Back](#)[Full Screen](#)[Close](#)[Quit](#)

creates a reflection about the  $y$ -axis (Figure 4a), whereas the matrix

$$A = \begin{bmatrix} \cos(\theta) & -\sin(\theta) \\ \sin(\theta) & \cos(\theta) \end{bmatrix}$$

creates a rotation of  $\theta$  about the origin (Figure 4b).

If the  $x$ -coordinate of a vector is multiplied by  $k$ , where  $k > 1$ , we obtain an expansion in the  $x$ -direction (Figure 5a). If the  $y$ -coordinate is multiplied, the expansion is in the  $y$ -direction. In these cases the standard matrices are

$$A = \begin{bmatrix} k & 0 \\ 0 & 1 \end{bmatrix} \quad \text{and} \quad A = \begin{bmatrix} 1 & 0 \\ 0 & k \end{bmatrix},$$

respectively. If  $0 < k < 1$  the result is a compression on the vectors (Figure 5b).

Home Page

Title Page

◀ ▶

◀ ▶

Page 7 of 32

Go Back

Full Screen

Close

Quit

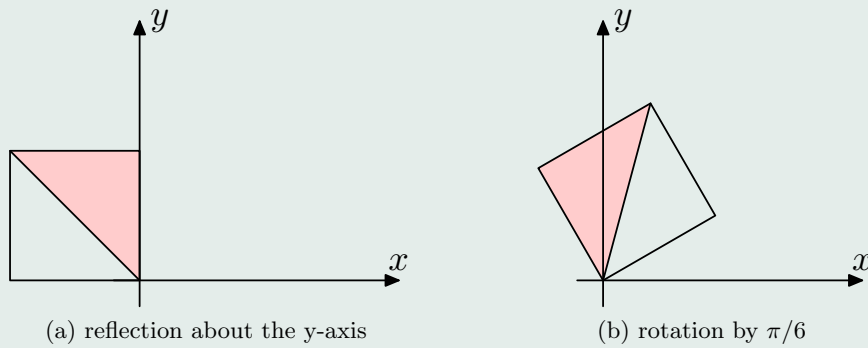


Figure 4: Reflection and Rotation

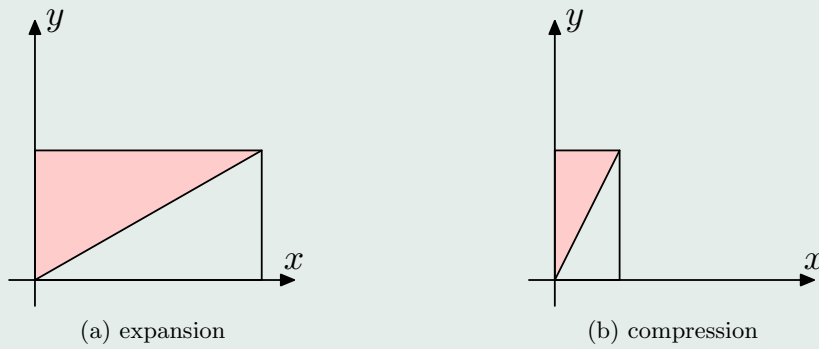


Figure 5: Expansion and Compression

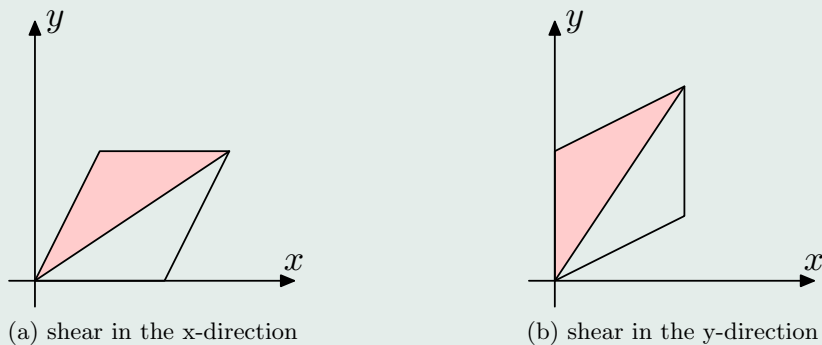


Figure 6: Shears

A shear in the  $x$ -direction is defined as a transformation that moves each point  $(x, y)$  parallel to the  $x$ -axis by an amount  $ky$  to the new position  $(x+ky, y)$ . This means that the further points get from the  $x$ -axis, the further they get moved in the  $x$ -direction (Figure 6a). A shear in the  $y$ -direction moves each point  $(x, y)$  parallel to the  $y$ -axis by an amount  $kx$  to the new position  $(x, y+kx)$  (Figure 6b).

The standard matrices for these operations are

$$A = \begin{bmatrix} 1 & 0 \\ k & 1 \end{bmatrix} \text{ and } A = \begin{bmatrix} 1 & k \\ 0 & 1 \end{bmatrix}.$$

By using the product of multiple standard matrices we can obtain more complex transformations. For example, to transform the unit square into the

shape shown in (Figure 7c) we would use the standard matrix

$$A_1 = \begin{bmatrix} 0 & 2 \\ 1 & 2 \end{bmatrix},$$

which is the product of three linear operations. This can be broken down into elementary steps by calculating a shear in the  $x$ -direction (Figure 7a) with

$$A = \begin{bmatrix} 1 & 2 \\ 0 & 1 \end{bmatrix},$$

an expansion in the  $y$ -direction (Figure 7b) with

$$A_2 = \begin{bmatrix} 1 & 0 \\ 0 & 2 \end{bmatrix},$$

and a reflection across the line  $y = x$  (Figure 7c) with

$$A_3 = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}.$$

The final result is the same whether we perform three calculations or a single one with the matrix

$$\begin{aligned} A &= A_3 A_2 A_1 \\ A &= \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} 1 & 0 \\ 0 & 2 \end{bmatrix} \begin{bmatrix} 1 & 2 \\ 0 & 1 \end{bmatrix} \\ A &= \begin{bmatrix} 0 & 2 \\ 1 & 2 \end{bmatrix}. \end{aligned}$$

Home Page

Title Page



Page 10 of 32

Go Back

Full Screen

Close

Quit

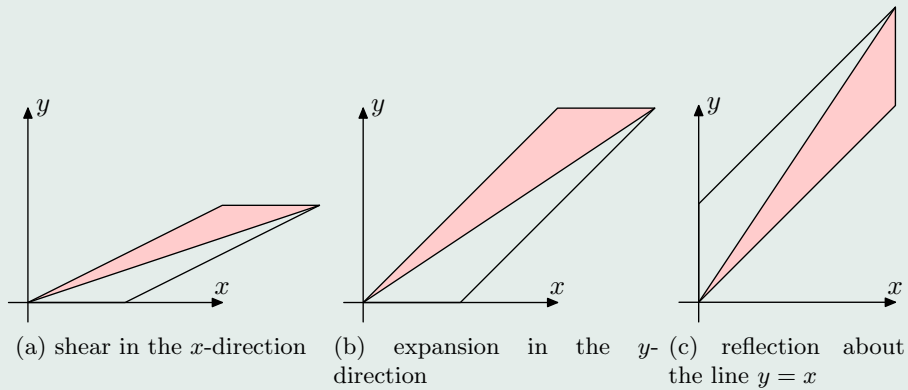


Figure 7: Combination of Basic Linear Operations

[Home Page](#)[Title Page](#)[◀◀](#) [▶▶](#)[◀](#) [▶](#)

Page 11 of 32

[Go Back](#)[Full Screen](#)[Close](#)[Quit](#)

There are a variety of different ways the shape of the unit square can be changed through matrix multiplication. In the program we calculate triangles instead of squares, but this is not a problem because the vectors we are multiplying are indices into each triangle on a point by point basis. Each point is represented by a vector in  $\mathbb{R}^2$ . This means that any point not bound by the enclosing triangle is excluded from the calculation. Another thing to notice is that each one of these operations can be reversed. The standard matrices are all invertible, with the exception of the zero matrix which shrinks everything down to a point in the origin, and projection matrices that flatten everything onto a line. In order to reverse the operation we need to multiply by the inverse of the standard matrix that was used and we are back to what we started with (note  $A^{-1}A = I$ ).

## Affine Transformations

To get all the indices of the pixels in the end triangle, we need to perform an affine transformation. In other words, multiply all the indices by a standard matrix that performs a linear operation on each of the coordinate points in the begin triangle then add a vector  $\mathbf{b}$  to the new points. Let  $\mathbf{v}$  represent any coordinate point in our begin triangle and  $\mathbf{w}$  the corresponding point in the end triangle, then

$$\mathbf{w} = M\mathbf{v} + \mathbf{b}.$$

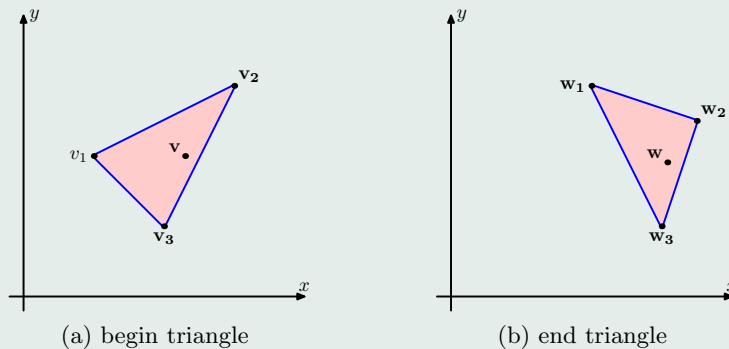


Figure 8: Vertex Point Displacement

It can be shown that

$$\mathbf{v} - \mathbf{v}_3 = c_1(\mathbf{v}_1 - \mathbf{v}_3) + c_2(\mathbf{v}_2 - \mathbf{v}_3) \quad (1)$$

$$\mathbf{v} = c_1\mathbf{v}_1 - c_1\mathbf{v}_3 + c_2\mathbf{v}_2 - c_2\mathbf{v}_3 + \mathbf{v}_3 \quad (2)$$

$$= c_1\mathbf{v}_1 + c_2\mathbf{v}_2 + (1 - c_1 - c_2)\mathbf{v}_3. \quad (3)$$

Setting

$$c_3 = 1 - c_1 - c_2,$$

the vector  $\mathbf{v}$  becomes

$$\mathbf{v} = c_1\mathbf{v}_1 + c_2\mathbf{v}_2 + c_3\mathbf{v}_3.$$

Let

$$\mathbf{w} = c_1\mathbf{w}_1 + c_2\mathbf{w}_2 + c_3\mathbf{w}_3, \quad (4)$$

[Home Page](#)
[Title Page](#)
[◀](#)
[▶](#)
[◀](#)
[▶](#)
[Page 12 of 32](#)
[Go Back](#)
[Full Screen](#)
[Close](#)
[Quit](#)

and

$$\mathbf{w}_i = M\mathbf{v}_i + \mathbf{b} \quad (5)$$

for

$$i = 1, 2, 3.$$

Substitute (5) into (4) and multiply it out;

$$\begin{aligned} \mathbf{w} &= c_1(M\mathbf{v}_1 + \mathbf{b}) + c_2(M\mathbf{v}_2 + \mathbf{b}) + c_3(M\mathbf{v}_3 + \mathbf{b}) \\ &= Mc_1\mathbf{v}_1 + c_1\mathbf{b} + Mc_2\mathbf{v}_2 + c_2\mathbf{b} + Mc_3\mathbf{v}_3 + c_3\mathbf{b}. \end{aligned}$$

The matrix  $M$  and vector  $\mathbf{b}$  can be factored out

$$\mathbf{w} = M(c_1\mathbf{v}_1 + c_2\mathbf{v}_2 + c_3\mathbf{v}_3) + (c_1 + c_2 + c_3)\mathbf{b}.$$

To obtain  $M$  and  $\mathbf{b}$  we set up a system of equations, using the vertex points that define the triangles we are working with.

Let

$$\mathbf{v}_i = \begin{bmatrix} v_{ix} \\ v_{iy} \end{bmatrix}, \quad \mathbf{w}_i = \begin{bmatrix} w_{ix} \\ w_{iy} \end{bmatrix} \quad (6)$$

be the  $i$ -th vertex points of the begin image and end image, respectively, where  $i = 1, 2, 3$  for the three vertex points. Then

$$\mathbf{w}_i = M\mathbf{v}_i + \mathbf{b}.$$

Let

$$\mathbf{b} = \begin{bmatrix} b_x \\ b_y \end{bmatrix} \quad \text{and} \quad M = \begin{bmatrix} M_1 & M_2 \\ M_3 & M_4 \end{bmatrix}$$



Then we can write equation (6) as

$$\begin{aligned} \begin{bmatrix} w_{ix} \\ w_{iy} \end{bmatrix} &= \begin{bmatrix} M_1 & M_2 \\ M_3 & M_4 \end{bmatrix} \begin{bmatrix} v_{ix} \\ v_{iy} \end{bmatrix} + \begin{bmatrix} b_x \\ b_y \end{bmatrix} \\ \begin{bmatrix} w_{ix} \\ w_{iy} \end{bmatrix} &= \begin{bmatrix} M_1 \\ M_3 \end{bmatrix} v_{ix} + \begin{bmatrix} M_2 \\ M_4 \end{bmatrix} v_{iy} + \begin{bmatrix} b_x \\ b_y \end{bmatrix} \\ \begin{bmatrix} w_{ix} \\ w_{iy} \end{bmatrix} &= \begin{bmatrix} M_1 v_{ix} + M_2 v_{iy} \\ M_3 v_{ix} + M_4 v_{iy} \end{bmatrix} + \begin{bmatrix} b_x \\ b_y \end{bmatrix}. \end{aligned}$$

From this expression it is trivial to set up a system of equations where  $\mathbf{v}_i$  and  $\mathbf{w}_i$  are the vertex points of the triangles and are known. The matrix  $M$  with its 4 elements and the two elements of  $\mathbf{b}$  are the unknowns.

$$\begin{aligned} w_{ix} &= M_1 v_{ix} + M_2 v_{iy} && + b_x \\ w_{iy} &= && M_3 v_{ix} + M_4 v_{iy} + b_y \end{aligned}$$

For each one of the vertex points we obtain two equations, take three vertex points, and you have six equations with six unknowns.

$$\begin{aligned} w_{1x} &= M_1 v_{1x} + M_2 v_{1y} && + b_x \\ w_{1y} &= && M_3 v_{1x} + M_4 v_{1y} + b_y \\ w_{2x} &= M_1 v_{2x} + M_2 v_{2y} && + b_x \\ w_{2y} &= && M_3 v_{2x} + M_4 v_{2y} + b_y \\ w_{3x} &= M_1 v_{3x} + M_2 v_{3y} && + b_x \\ w_{3y} &= && M_3 v_{3x} + M_4 v_{3y} + b_y \end{aligned}$$

Note that the six variables are lined up above each other. The next step is to put this system of equations in matrix form

$$\begin{bmatrix} v_{1x} & v_{1y} & & & & 1 \\ & & v_{1x} & v_{1y} & & 1 \\ v_{2x} & v_{2y} & & & & 1 \\ & & v_{2x} & v_{2y} & & 1 \\ v_{3x} & v_{3y} & & & & 1 \\ & & v_{3x} & v_{3y} & & 1 \end{bmatrix} \begin{bmatrix} M_1 \\ M_2 \\ M_3 \\ M_4 \\ b_x \\ b_y \end{bmatrix} = \begin{bmatrix} w_{1x} \\ w_{1y} \\ w_{2x} \\ w_{2y} \\ w_{3x} \\ w_{3y} \end{bmatrix}$$

and set up the augmented matrix.

$$A = \begin{bmatrix} v_{1x} & v_{1y} & & & 1 & w_{1x} \\ & & v_{1x} & v_{1y} & 1 & w_{1y} \\ v_{2x} & v_{2y} & & & 1 & w_{2x} \\ & & v_{2x} & v_{2y} & 1 & w_{2y} \\ v_{3x} & v_{3y} & & & 1 & w_{3x} \\ & & v_{3x} & v_{3y} & 1 & w_{3y} \end{bmatrix}$$

This matrix can be row reduced to find the solution for the elements of  $M$  and  $b$ .

## An Example

Let's do an example to make the concept clear. The goal is to transform the triangle with vertex points  $v_1$ ,  $v_2$  and  $v_3$  (Figure 9a), where

$$v_1 = (1, 1), \quad v_2 = (2, 2) \quad \text{and} \quad v_3 = (3, 1),$$

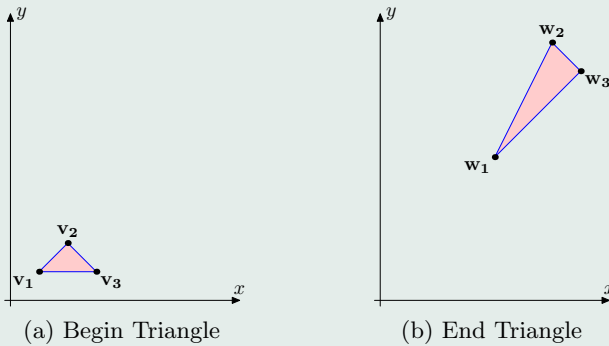


Figure 9: Transforming Triangles

or in vector representation,

$$\mathbf{v}_1 = \begin{bmatrix} 1 \\ 1 \end{bmatrix}, \quad \mathbf{v}_2 = \begin{bmatrix} 2 \\ 2 \end{bmatrix} \quad \text{and} \quad \mathbf{v}_3 = \begin{bmatrix} 3 \\ 1 \end{bmatrix}.$$

The objective is to transform this triangle into a triangle with vertex points  $w_1, w_2$  and  $w_3$  (Figure 9b) where

$$w_1 = (4, 5), \quad w_2 = (6, 9) \quad \text{and} \quad w_3 = (7, 8).$$

Put this in vector representation, then

$$\mathbf{w}_1 = \begin{bmatrix} 4 \\ 5 \end{bmatrix}, \quad \mathbf{w}_2 = \begin{bmatrix} 6 \\ 9 \end{bmatrix} \quad \text{and} \quad \mathbf{w}_3 = \begin{bmatrix} 7 \\ 8 \end{bmatrix}.$$

Now it is possible to set up the system of equations

$$4 = M_1(1) + M_2(1) + b_x$$

$$5 = M_3(1) + M_4(1) + b_y$$

$$6 = M_1(2) + M_2(2) + b_x$$

$$9 = M_3(2) + M_3(2) + b_y$$

$$7 = M_1(3) + M_2(1) + b_x$$

$$8 = M_3(3) + M_4(1) + b_y$$

and put it into an augmented matrix.

$$A = \begin{bmatrix} 1 & 1 & & & 4 \\ & & 1 & 1 & 5 \\ 2 & 2 & & & 6 \\ & & 2 & 2 & 9 \\ 3 & 1 & & & 7 \\ & & 3 & 1 & 8 \end{bmatrix}$$

Using linear algebra this matrix can be row reduced.

$$R = \begin{bmatrix} 1 & & & & 3/2 \\ & 1 & & & 1/2 \\ & & 1 & & 3/2 \\ & & & 1 & 5/2 \\ & & & & 2 \\ & & & & 1 & 1 \end{bmatrix}$$

[Home Page](#)
[Title Page](#)
[◀◀](#)
[▶▶](#)
[◀](#)
[▶](#)

Page 17 of 32

[Go Back](#)
[Full Screen](#)
[Close](#)
[Quit](#)

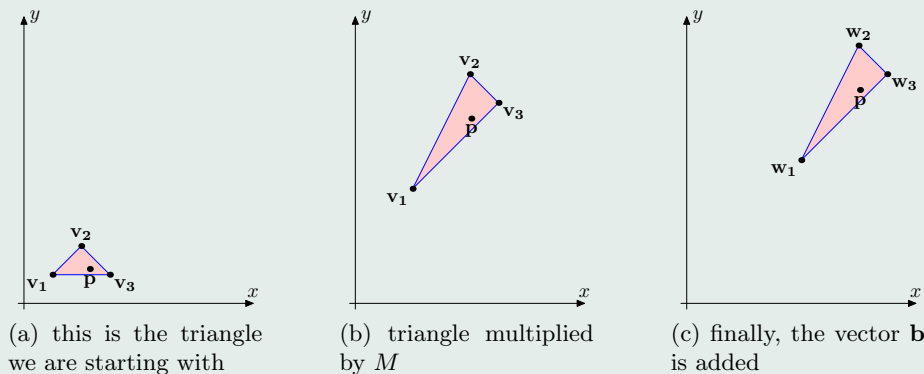
[Home Page](#)
[Title Page](#)
[◀](#)
[▶](#)
[◀](#)
[▶](#)
[Page 18 of 32](#)
[Go Back](#)
[Full Screen](#)
[Close](#)
[Quit](#)


Figure 10: Warping a Triangle

From  $R$  we obtain

$$M = \begin{bmatrix} 3/2 & 1/2 \\ 3/2 & 5/2 \end{bmatrix} \quad \text{and} \quad \mathbf{b} = \begin{bmatrix} 2 \\ 1 \end{bmatrix}.$$

Now let's see what this does to our triangle  $\triangle \mathbf{v}_1 \mathbf{v}_2 \mathbf{v}_3$  and a point  $\mathbf{p}$  in the triangle (Figure 10a), where

$$\mathbf{p} = \begin{bmatrix} 2.3 \\ 1.2 \end{bmatrix}.$$

First let's multiply our triangle and our point  $\mathbf{p}$  by  $M$  and observe the result (Figure 10b).

$$\mathbf{v}'_1 = M\mathbf{v}_1 = \begin{bmatrix} 3/2 & 1/2 \\ 3/2 & 5/2 \end{bmatrix} \begin{bmatrix} 1 \\ 1 \end{bmatrix} = \begin{bmatrix} 2 \\ 4 \end{bmatrix}$$

$$\mathbf{v}'_2 = M\mathbf{v}_2 = \begin{bmatrix} 3/2 & 1/2 \\ 3/2 & 5/2 \end{bmatrix} \begin{bmatrix} 2 \\ 2 \end{bmatrix} = \begin{bmatrix} 4 \\ 8 \end{bmatrix}$$

$$\mathbf{v}'_3 = M\mathbf{v}_3 = \begin{bmatrix} 3/2 & 1/2 \\ 3/2 & 5/2 \end{bmatrix} \begin{bmatrix} 3 \\ 1 \end{bmatrix} = \begin{bmatrix} 5 \\ 7 \end{bmatrix}$$

$$\mathbf{p}' = M\mathbf{p} = \begin{bmatrix} 3/2 & 1/2 \\ 3/2 & 5/2 \end{bmatrix} \begin{bmatrix} 2.3 \\ 1.2 \end{bmatrix} = \begin{bmatrix} 4.05 \\ 6.45 \end{bmatrix}$$

As we can see the triangle already has the shape of our final triangle but it needs to be moved. Recall that some transformations can't be performed only by matrix multiplication. The last step will be to add the vector  $\mathbf{b}$  (Figure 10c).

$$\mathbf{w}_1 = \mathbf{v}'_1 + \mathbf{b} = \begin{bmatrix} 2 \\ 4 \end{bmatrix} + \begin{bmatrix} 2 \\ 1 \end{bmatrix} = \begin{bmatrix} 4 \\ 5 \end{bmatrix}$$

$$\mathbf{w}_2 = \mathbf{v}'_2 + \mathbf{b} = \begin{bmatrix} 4 \\ 8 \end{bmatrix} + \begin{bmatrix} 2 \\ 1 \end{bmatrix} = \begin{bmatrix} 6 \\ 9 \end{bmatrix}$$

$$\mathbf{w}_3 = \mathbf{v}'_3 + \mathbf{b} = \begin{bmatrix} 5 \\ 7 \end{bmatrix} + \begin{bmatrix} 2 \\ 1 \end{bmatrix} = \begin{bmatrix} 7 \\ 8 \end{bmatrix}$$

$$\mathbf{p}_t = \mathbf{p}' + \mathbf{b} = \begin{bmatrix} 4.05 \\ 6.45 \end{bmatrix} + \begin{bmatrix} 2 \\ 1 \end{bmatrix} = \begin{bmatrix} 6.05 \\ 7.45 \end{bmatrix}$$

[Home Page](#)[Title Page](#)[◀](#) [▶](#)[◀](#) [▶](#)

Page 19 of 32

[Go Back](#)[Full Screen](#)[Close](#)[Quit](#)

[Home Page](#)[Title Page](#)[◀](#) [▶](#)[◀](#) [▶](#)

Page 20 of 32

[Go Back](#)[Full Screen](#)[Close](#)[Quit](#)

Alternatively, we can use homogeneous coordinates to perform our affine transformation through matrix multiplication in one step. This means representing a vector  $\begin{bmatrix} v_x \\ v_y \end{bmatrix}$  in  $\mathbb{R}^2$  as a vector  $\begin{bmatrix} v_x \\ v_y \\ 1 \end{bmatrix}$  in  $\mathbb{R}^3$ . Using this system, translation can be expressed with matrix multiplication, thereby resulting in an affine transformation. This is easily accomplished once we have obtained the matrix  $M$  and the vector  $\mathbf{b}$  by placing them into an augmented matrix and adding a row of zeros, with the exception of the last pivot location which we set equal to 1, at the bottom.

The standard matrix then becomes

$$\begin{bmatrix} M_1 & M_2 & b_x \\ M_3 & M_4 & b_y \\ 0 & 0 & 1 \end{bmatrix}.$$

We can then represent the vertex points and all the points within the triangle as points in  $\mathbb{R}^3$  by simply adding a 1 in the third row of each vector. This will allow us to assemble an augmented matrix with each vertex point and each individual point within the enclosing region for matrix multiplication.

The end result is

$$\begin{bmatrix} w_{1x} & w_{2x} & w_{3x} & p'_x & \dots \\ w_{1y} & w_{2y} & w_{3y} & p'_y & \dots \\ 1 & 1 & 1 & 1 & \dots \end{bmatrix} = \begin{bmatrix} M_1 & M_2 & b_x \\ M_3 & M_4 & b_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} v_{1x} & v_{2x} & v_{3x} & p_x & \dots \\ v_{1y} & v_{2y} & v_{3y} & p_y & \dots \\ 1 & 1 & 1 & 1 & \dots \end{bmatrix} \quad (7)$$

$$\begin{aligned}
 \begin{bmatrix} w_{1x} & w_{2x} & w_{3x} & p'_x \\ w_{1y} & w_{2y} & w_{3y} & p'_y \\ 1 & 1 & 1 & 1 \end{bmatrix} &= \begin{bmatrix} M_1 & M_2 & b_x \\ M_3 & M_4 & b_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} v_{1x} & v_{2x} & v_{3x} & p_x \\ v_{1y} & v_{2y} & v_{3y} & p_y \\ 1 & 1 & 1 & 1 \end{bmatrix} \\
 &= \begin{bmatrix} 3/2 & 1/2 & 2 \\ 3/2 & 5/2 & 1 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 2 & 3 & 2.3 \\ 1 & 2 & 1 & 1.2 \\ 1 & 1 & 1 & 1 \end{bmatrix} \\
 &= \begin{bmatrix} 4 & 6 & 7 & 6.05 \\ 5 & 9 & 8 & 7.45 \\ 1 & 1 & 1 & 1 \end{bmatrix}.
 \end{aligned}$$

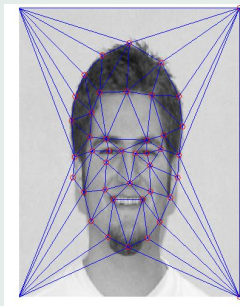
After discarding the last row, we have

$$\begin{bmatrix} 4 & 6 & 7 & 6.05 \\ 5 & 9 & 8 & 7.45 \end{bmatrix}.$$

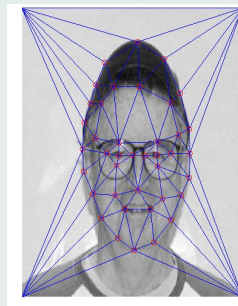
## Warping an Image

This triangle was successfully warped from one shape and position into another. Also all the points of the triangle are warped if the affine transformation is applied to every one of them. This is the main idea of warping an image, where the key is that some amount of triangles are laid above the image. The affine transformation maps every pixel in the image to a new shape and therefore

[Home Page](#)
[Title Page](#)
[◀](#)
[▶](#)
[◀](#)
[▶](#)
[Page 21 of 32](#)
[Go Back](#)
[Full Screen](#)
[Close](#)
[Quit](#)



(a) begin image



(b) end image

Figure 11: Triangulation of a Pair of Images

distorts the image. In the previous section we successfully warped one triangle. What can be done with one triangle can be done with many triangles. Figure 11 shows a pair of images with 82 triangles in each image and photographs of two faces. Each photo can be thought of as 82 triangles which make up the entire picture. Note that each triangle in the end image corresponds to one triangle in the begin image and that the triangles enclosing boundary contains specific facial features that match the features of the corresponding triangle. The task at hand is to perform an affine transformation of every triangle in the begin image to warp it into the shape of the corresponding triangle in the end image (Figure 12). This is done with the method described earlier by finding a unique 2-by-2 matrix  $M$  and a two-dimensional vector  $\mathbf{b}$  for each triangle. In the



(a) begin image



(b) end image

Figure 12: Warp of an Image

example (Figure 11) we have 82 triangles and we need 82 affine transformations to accomplish a warp of the image. The result is an image that is in the shape of the end image, but still has the same gray scale values as the begin image. Before we can do a morph we still need to know how to do one more thing.

## Time-Varying Warps

If a series of intermediate frames are captured as the image warps a time-varying warp can be produced. This creates the illusion of a motion picture that gradually changes from the begin image into the end image. The technique behind this is to let the vertex points of the triangles continuously move from their

starting point, which is their location in the begin image to their destination, which is their location in the end image. Their motion along a line is a function with respect to time. Let our time units range from  $t = 0$  to  $t = 1$ , where  $t = 0$  corresponds to the begin image and  $t = 1$  corresponds to the end image.

Suppose  $\mathbf{u}_i(t)$  denotes the position of the  $i$ -th vertex point at time  $t$ . The point travels along the line at a constant velocity. This means that  $\mathbf{u}_i(0) = \mathbf{v}_i$  (its position in the begin image) and  $\mathbf{u}_i(1) = \mathbf{w}_i$  (its position in the end image). For any time in between the position is given by

$$\mathbf{u}_i(t) = (1 - t)\mathbf{v}_i + t(\mathbf{w}_i).$$

Equipped with this knowledge it is possible to set up a triangulation for any given time  $t$  and perform a warp from the begin image to this new intermediate position. Figure 13 is the result of computing the warps at five different times. When put together these frames create a motion picture of the face slowly being distorted.

If this procedure is applied to the end image as well and if the results are reversed, in other words, we start with the warped image and work towards the original image (Figure 14) we have the basis for a morph.

## Picture Density

The grayscale component of the image is stored in matrix form and the concentration of color of a given pixel is referred to as picture density. The density of the image, which we will define as  $\rho$ , must also be mapped for each point. The values range from 0 to 255 and go from white to black for the colormap

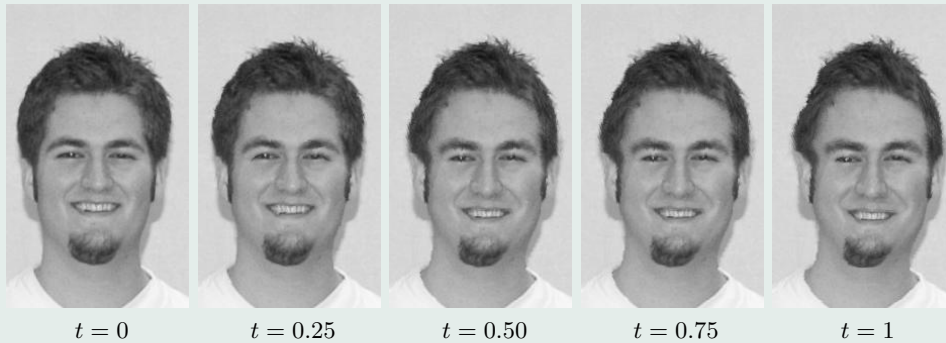


Figure 13: Time-Varying Warp of Begin Image

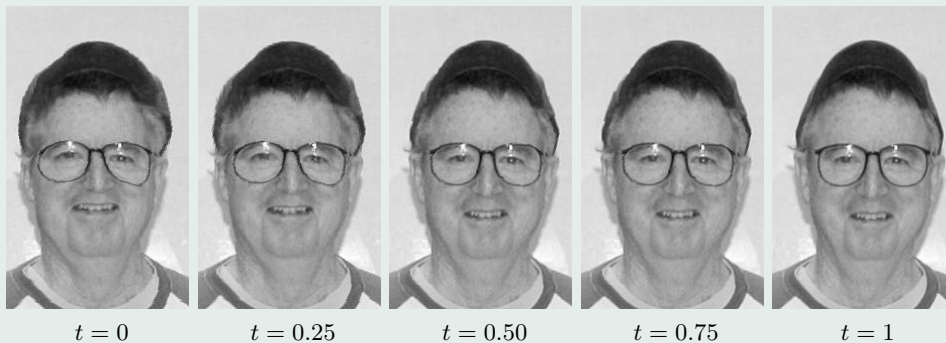


Figure 14: Reversed Time-Varying Warp of End Image

Home Page

Title Page

◀▶

◀▶

Page 25 of 32

Go Back

Full Screen

Close

Quit

[Home Page](#)[Title Page](#)[◀◀](#) [▶▶](#)[◀](#) [▶](#)

Page 26 of 32

[Go Back](#)[Full Screen](#)[Close](#)[Quit](#)

that we are using. If we are mapping the point  $v$  to  $w$  and the corresponding gray values are  $\rho_v = 100$  and  $\rho_w = 200$  respectively, then we need to obtain a weighted value as the point changes color. The weight depends on the distance that the point has traveled at each step. For example, if we are constructing eleven frames for the morphed image, then the value of  $\rho(u)$  will change by  $1/10$  for each frame, where  $\rho_t(u)$  is the density at time  $t$  in frame  $u$ . The equation to obtain the value at each step is

$$\rho_t(u) = (1 - t)\rho_0 + t\rho_1$$

where  $t$  is the distance the point has transversed along the path from  $v$  to  $w$  and  $u$  is the location at that distance. This will provide a smooth transition from the color of  $v$  to the color of  $w$ . The transition from  $v$  to  $w$  would produce

$$\rho_v = 100 \rightarrow 110 \rightarrow 120 \rightarrow \dots \rightarrow 190 \rightarrow 200 = \rho_w.$$

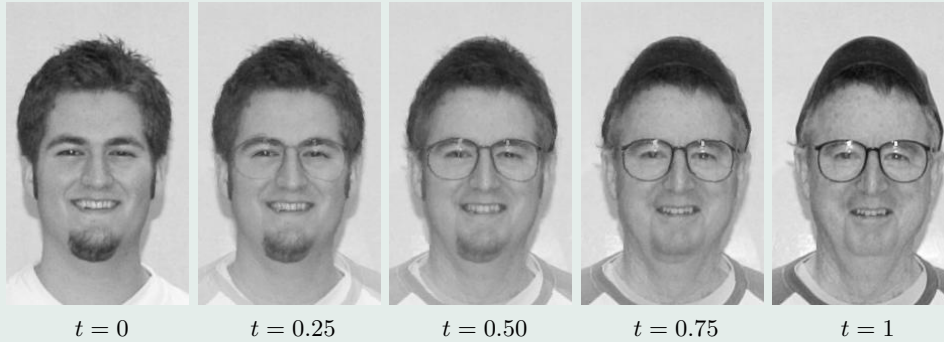


Figure 15: Morph with 5 frames

## References

- [1] D. Arnold. For all his help on this project.
- [2] G. Strang. *Introduction to Linear Algebra, Third Ed.*
- [3] H. Anton, C. Rorres. *Elementary Linear Algebra with Applications, 8th Ed.*
- [4] S. Johansson. *Computing in Matlab*

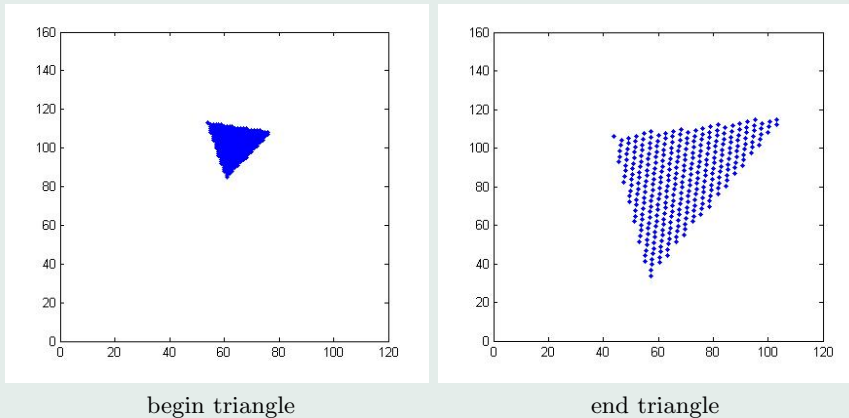


Figure 16: Morph of a Triangle with a finite Number of Points

## Appendix

### Trouble with Discrete Values

When writing a Matlab program that morphs two images, we ran into some problems. First, the idea of multiplying a triangle with a matrix works fine if there are an infinite number of points in the triangle, and if we are multiplying each one of them with our matrix. Unfortunately when handling an image we are dealing with discrete values. The picture only has a finite number of pixels that we can multiply with. Imagine a triangle that is being stretched from

Home Page

Title Page

◀ ▶

◀ ▶

Page 29 of 32

Go Back

Full Screen

Close

Quit

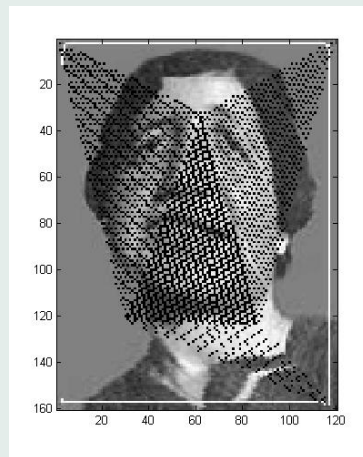


Figure 17: A Warp that went wrong

[Home Page](#)[Title Page](#)[◀◀](#) [▶▶](#)[◀](#) [▶](#)

Page 30 of 32

[Go Back](#)[Full Screen](#)[Close](#)[Quit](#)

its begin shape into a larger end shape (Figure 16). Because there are only a certain amount of pixels in the begin triangle, and the end triangle consists of more pixels than the begin triangle, the result of the warp is an image that has gaps in it. The amount of pixels that where enough to cover the area of the begin triangle are no longer sufficient to cover the area of the end triangle. Figure 17 is an example of a warp that went wrong due to this issue.

The way to resolve this issue is to reverse the process of the warp. The conventional way to warp the image would be to take the index of each pixel, multiply it by the standard matrix  $M$  and add the vector  $\mathbf{b}$  (Equation 8) and the pixel with the resulting index must have the same gray scale value as the one we started with.

If we start with the warped shape of the triangle and go reverse to retrieve the picture density of every pixel in the end triangle we will have a value for every one of those pixels and there will be no missing pixels.

We know that

$$\mathbf{w} = M\mathbf{v} + \mathbf{b}$$

is the equation for warping an object. It is easy to reverse the process.

$$\mathbf{w} = M\mathbf{v} + \mathbf{b} \tag{8}$$

$$\mathbf{w} - \mathbf{b} = M\mathbf{v} \tag{9}$$

$$M^{-1}(\mathbf{w} - \mathbf{b}) = \mathbf{v} \tag{10}$$

The first step will be to subtract the vector  $\mathbf{b}$  from all of our points (Equation 9), then we need to multiply the result by the inverse of the standard matrix  $M$

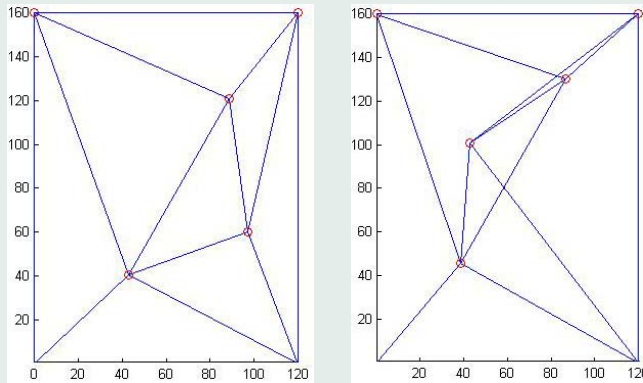


Figure 18: Overlapping Triangles

(Equation 10) and now we have an equation where we can plug in  $\mathbf{w}$  to obtain its corresponding gray scale value, which is equal to the gray scale value at  $\mathbf{v}$ .

## Overlapping Triangles

Another issue with warps are overlapping triangles. If a vertex point in the end triangulation is moved across the line of another triangle (Figure 18) we are going to run into troubles. The problem is that now some points are defined to belong to more than one triangle. Since the standard matrix and the vector  $\mathbf{b}$  are different for every triangle it is not clear which matrix and which vector to apply.



The only way we could resolve this issue was by carefully choosing our vertex points to make sure the triangles don't overlap.

## Experiment on Your Own

If you want to experiment with the program the instructions are as follows:

1. Go to the url provided below
2. Download and save the zip to a local directory in the MatLab path.
3. Extract all the files.
4. Open MatLab and set the current directory to the folder with the extracted files.
5. Go to MatLab's command line prompt and type 'metamorphosis'.
6. When asked to choose a begin image type in the name as shown on screen.
7. Choices should look like 'choice' with quotes.

The MatLab files that generate the morph can be found at

[http://online.redwoods.edu/instruct/darnold/laproj/fall2005/  
petertim/archive.zip](http://online.redwoods.edu/instruct/darnold/laproj/fall2005/petertim/archive.zip)