Assignment: Image Warping

CSE 4280/5280

January 24, 2020

1 Objective

In this assignment, you will implement a method for warping images using affine transformations. Your program will create a warped version of the source image using the affine transformation that maps between two triangular shapes. While the affine transformation will be estimated based on the pair of triangles, your program will warp the whole image (i.e., all pixels from the source image will be mapped onto pixels in the destination image),

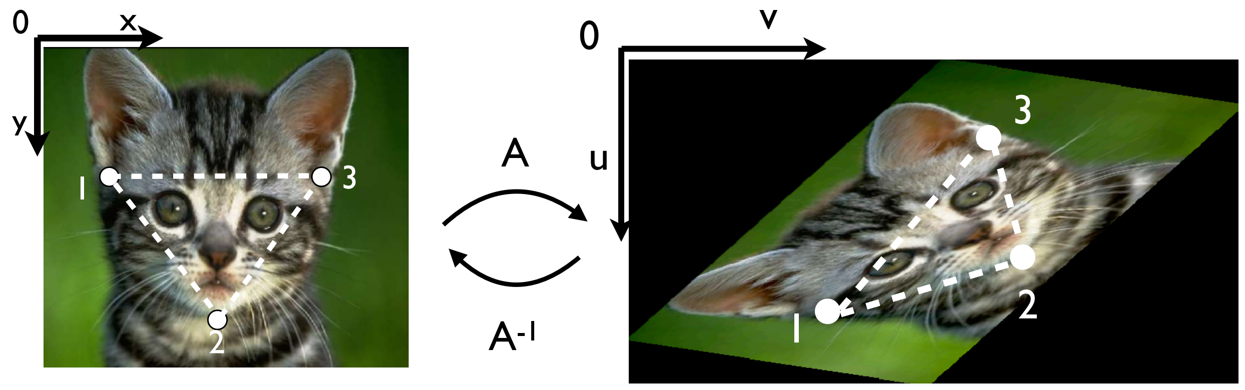

Figure 1: Image warp. Given three pair of corresponding vertices, we can estimate the affine transformation A between two shapes. Transformation A (i.e., forward transformation) allows us to calculate the size of the destination image. The inverse transformation, A^{-1} is used for transferring the correct color values from the source image to the destination image.

2 Warping shapes

In the basic image-warping problem, we are given an image and a destination shape onto which we want the image to be mapped. This problem is analogous to that of texture mapping. Figure 2 (left) shows an image of a circle enclosing an arrow. The destination shape is the triangle delimited by the vertices colored in red, green, and blue. In the figure, the colors indicate correspondence between pairs of vertices of the two shapes.

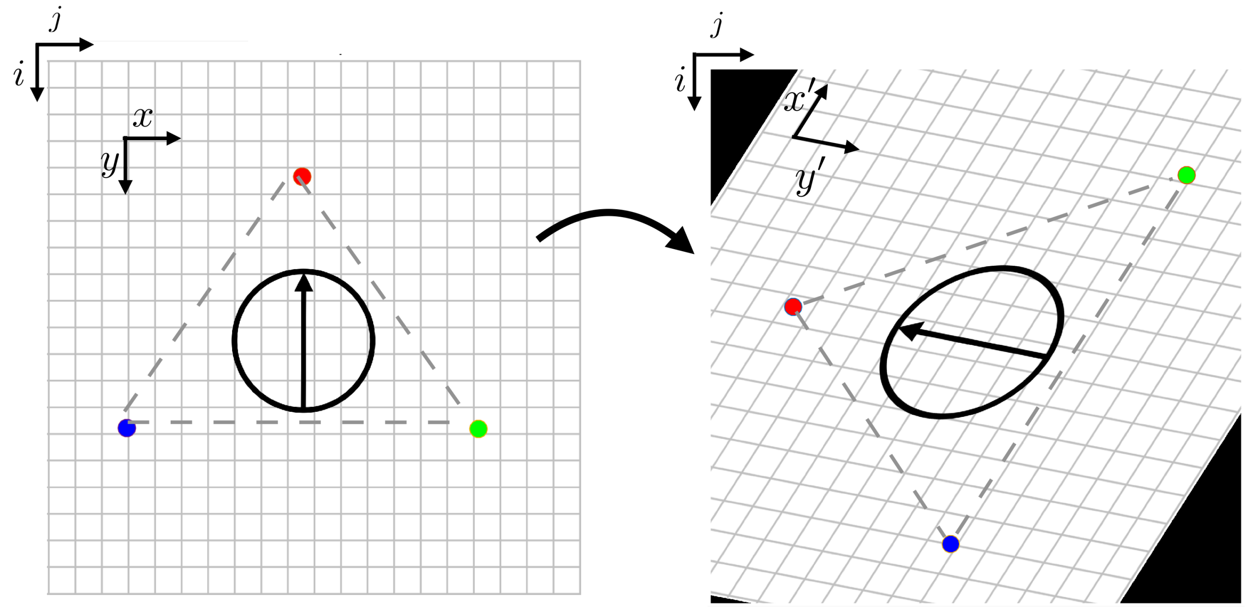

Figure 2: The circle is mapped from one triangle to another via an affine transformation. The affine transformation warps the coordinate system, stretching all patterns accordingly.

To warp the shape, we use an inverse-mapping technique to avoid the occurrence of holes in the destination matrix. Here, we must calculate the mathematical transformation between pixels on the source image and the corresponding pixels in the destination image. The basic warping algorithm using inverse mapping is listed in Algorithm 1:

Algorithm 1 Inverse image mapping

```
for all $(x', y')$ do                                      ▷ Loop over all pixels is dst image
    $(x, y) \leftarrow \mathcal{A}^{-1}\{(x', y')\}$        ▷ Transform pixel coords from dst to src images
    $I_{dst}(x', y') \leftarrow I_{src}(x, y)$              ▷ Copy pixel color from src image to the dst image
end for
```

A common choice of transformation is the affine transformation, $\mathcal{A}$. The affine transformation is calculated using three (or more) pairs of corresponding vertices of the two shapes (triangles in these notes). With the affine transformation at hand, we can apply the warp to the original image. The result of the warp is shown on the right-hand side of Figure 2.

3 The affine mappings between triangles

3.1 The affine transformation

Two planar triangles are related by an affine transformation. Let $\mathbf{t} = \{(x_1, y_1), (x_2, y_2), (x_3, y_3)\}$ and $\mathbf{t}' = \{(x'_1, y'_1), (x'_2, y'_2), (x'_3, y'_3)\}$ be two triangles. The affine transformation $\mathcal{A}$ that maps a point $(x, y)^\mathsf{T}$ to a point $(x', y')^\mathsf{T}$ is given by:

$$\begin{aligned} x' &= a_{11}x + a_{12}y + a_{13} \\ y' &= a_{21}x + a_{22}y + a_{23}, \end{aligned} \tag{1}$$

where $a_{i,j}$ are the parameters of the transformation. Geometrically, the affine transformation includes rotation, non-uniform scaling, shear, reflection, and translation.

3.2 Homogeneous coordinates

The affine transformation in (1) is not a linear transformation because it does not preserve the origin (i.e., it can shift the entire shape away from the origin). Luckily, we can obtain a linear representation of the affine (i.e., a matrix form) by augmenting the coordinates of the mapped points with an extra non-zero integer, e.g., points $(x_i, y_i)^\mathsf{T}$ become $(x_i, y_i, 1)^\mathsf{T}$ and points $(x'_i, y'_i)^\mathsf{T}$ become $(x'_i, y'_i, 1)^\mathsf{T}$. These augmented coordinates are called *homogenous coordinates*.

3.3 The matrix form of the affine transformation

A linear representation of the transformation allows us to write it in a matrix form. By using homogeneous we can express Equation 1 in a matrix form, i.e.:

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}. \tag{2}$$

or, in short:

$$\mathbf{x}' = A\mathbf{x}. \tag{3}$$

3.4 Estimating the unknown parameters of the affine

To estimate the parameters of the affine transformation, we first re-shape Equation 2 to form the following system of equations:

$$\begin{bmatrix} x & y & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & x & y & 1 \end{bmatrix} \begin{bmatrix} a_{11} \\ a_{12} \\ a_{13} \\ a_{21} \\ a_{22} \\ a_{23} \end{bmatrix} = \begin{bmatrix} x' \\ y' \end{bmatrix}, \tag{4}$$

or, in short:

$$M\mathbf{a} = \mathbf{b}. \tag{5}$$

To solve Equation 5 for **a**, we must invert the matrix on the left-hand side, i.e.:

$$\mathbf{a} = M^{-1}\mathbf{b}, \tag{6}$$

However, with a single pair of points, the system of equations is *under constrained* and cannot be solved. Because the system has 6 unknowns, we need at least 6 rows in the left-hand side matrix. Since each pair of corresponding points yields two constraints of the matrix (i.e., two rows), and we need three pairs of points to form an equation that we can solve. When we are relating triangles via affine transformation, the three points are the vertices of the triangles.

When using the vertices of two triangles as measurements, Equation 4 becomes:

$$\begin{bmatrix} x_1 & y_1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & x_1 & y_1 & 1 \\ x_2 & y_2 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & x_2 & y_2 & 1 \\ x_3 & y_3 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & x_3 & y_3 & 1 \end{bmatrix} \begin{bmatrix} a_{11} \\ a_{12} \\ a_{13} \\ a_{21} \\ a_{22} \\ a_{23} \end{bmatrix} = \begin{bmatrix} x'_1 \\ y'_1 \\ x'_2 \\ y'_2 \\ x'_3 \\ y'_3 \end{bmatrix}, \tag{7}$$

which we can also re-write by grouping together all x-coordinates and y-coordinates. The following form can be convenient for programming purposes:

$$\begin{bmatrix} x_1 & y_1 & 1 & 0 & 0 & 0 \\ x_2 & y_2 & 1 & 0 & 0 & 0 \\ x_3 & y_3 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & x_1 & y_1 & 1 \\ 0 & 0 & 0 & x_2 & y_2 & 1 \\ 0 & 0 & 0 & x_3 & y_3 & 1 \end{bmatrix} \begin{bmatrix} a_{11} \\ a_{12} \\ a_{13} \\ a_{21} \\ a_{22} \\ a_{23} \end{bmatrix} = \begin{bmatrix} x'_1 \\ x'_2 \\ x'_3 \\ y'_1 \\ y'_2 \\ y'_3 \end{bmatrix}, \tag{8}$$

or:

$$\begin{bmatrix} \mathbf{x}_{3\times1} & \mathbf{y}_{3\times1} & \mathbf{1}_{3\times1} & \mathbf{0}_{3\times1} & \mathbf{0}_{3\times1} & \mathbf{0}_{3\times1} \\ \mathbf{0}_{3\times1} & \mathbf{0}_{3\times1} & \mathbf{0}_{3\times1} & \mathbf{x}_{3\times1} & \mathbf{y}_{3\times1} & \mathbf{1}_{3\times1} \end{bmatrix} \begin{bmatrix} a_{11} \\ a_{12} \\ a_{13} \\ a_{21} \\ a_{22} \\ a_{23} \end{bmatrix} = \begin{bmatrix} \mathbf{x}'_{3\times1} \\ \mathbf{y}'_{3\times1} \end{bmatrix}. \tag{9}$$

4 Algorithms

The complete algorithm for the affine warping using triangles is given in Algorithm 2.

Algorithm 2 Affine Image Warp

```
 1: procedure affineWarp( )
Require:
      $I_{src}$                                                   ▷ Source image
      $\mathbf{x} = \{(x_i, y_i)\}_{i=1}^{3}$                    ▷ Vertices from source triangle
      $\mathbf{x}' = \{(x_i', y_i')\}_{i=1}^{3}$                 ▷ Vertices from destination triangle

 2:     $A \leftarrow estimateAffine(\mathbf{x}, \mathbf{x}')$    ▷ Estimate transformation between two triangles
 3:     $I_{dst} \leftarrow inverseMapping(A, \mathbf{x}, \mathbf{x}', I_{src})$    ▷ Warp pixels from src to dst image
 4:     $displayImage(I_{dst})$
 5: end procedure

 6: function $I_{dst}$ = inverseMapping($A, \mathbf{x}, \mathbf{x}', I_{src}$)
 7:     $n_{rows}, n_{cols} \leftarrow getDstImageSize(\mathbf{x}')$    ▷ Obtain size of destination image
 8:     $I_{dst} \leftarrow zeros(n_{rows}, n_{cols})$    ▷ Initialize destination image
 9:     for all $(x', y')$ do    ▷ Loop over all pixels is dst image
10:         $(x, y) \leftarrow A^{-1}\{(x', y')\}$    ▷ Obtain pixel coords from dst image to src image
11:         $I_{dst}(x', y') \leftarrow I_{src}(x, y)$    ▷ Copy pixel color from src image to dst
12:     end for
13:     return $I_{dst}$
14: end function

15: function $A$ = estimateAffine($\mathbf{x}, \mathbf{x}'$)    ▷ 3 pairs of corresponding vertices
16:     Create matrix $M$ and vector $\mathbf{b}$ using the 3 pairs of vertices    ▷ Eq. 5 and Eq. 8
17:     Calculate $\mathbf{a} \leftarrow M^{-1}\mathbf{b}$    ▷ Affine coefficients (Eq. 6)
18:     $A \leftarrow reshape(\mathbf{a}, 3 \times 3)$    ▷ Create the (homogenous) affine matrix (Eq. 3)
19:     return $A$
20: end function

21: function $n_{rows}, n_{cols}$ = getDstImageSize($\mathbf{x}'$)
22:     $x_{min} \leftarrow min(x_i')$    ▷ Max. and Min. of polygon boundaries
23:     $x_{max} \leftarrow max(x_i')$
24:     $y_{min} \leftarrow min(y_i')$
25:     $y_{max} \leftarrow max(y_i')$
26:     $n_{rows} \leftarrow y_{max} - y_{min}$    ▷ Num of rows (i.e., height)
27:     $n_{cols} \leftarrow x_{max} - x_{min}$    ▷ Num of cols (i.e., width)
28:     return $n_{rows}, n_{cols}$
29: end function
```