EL512 --- Image Processing

Geometric Transformations: Warping, Registration, Morphing

Yao Wang
Polytechnic University, Brooklyn, NY 11201

With contribution from Zhu Liu, Onur Guleryuz, and
Partly based on
Lecture Outline

- Introduction
- Image deformation model
- Image warping
- Image registration
- Image morphing
What is Geometric Transformation?

• So far, the image processing operations we have discussed modify the color values of pixels in a given image.

• With geometric transformation, we modify the positions of pixels in an image, but keep their colors unchanged.
  – To create special effects
  – To register two images taken of the same scene at different times
  – To morph one image to another
How to define a geometric transformation?

• Let \((u, v)\) represent the image coordinate in an original image, and \((x, y)\) in a deformed (or warped) image. We use a function pair to relate corresponding pixels in the two images:
  – Forward mapping:
    \[
    \begin{align*}
    x &= x(u, v) \\
    y &= y(u, v), \quad \text{or} \quad x = x(u)
    \end{align*}
    \]
  – Inverse mapping:
    \[
    \begin{align*}
    u &= u(x, y) \\
    v &= v(x, y), \quad \text{or} \quad u = u(x)
    \end{align*}
    \]

• Let \(f(u, v)\) or \(f(u)\) denote the original image and \(g(x, y)\) or \(g(x)\) the deformed image. Then they are related by:
    \[
    \begin{align*}
    g(x, y) &= f(u(x, y), v(x, y)) \\
    f(u, v) &= g(x(u, v), y(u, v)), \quad \text{or} \quad f(u) = g(x(u))
    \end{align*}
    \]
Illustration of Forward and Inverse Mapping Functions

Forward
\( x(u,v), y(u,v) \)

Inverse
\( u(x,y), v(x,y) \)
Translation

- **Translation** is defined by the following mapping functions:

\[
\begin{align*}
    x &= u + t_x \\
    y &= v + t_y \\
\end{align*}
\]

and

\[
\begin{align*}
    u &= x - t_x \\
    v &= y - t_y \\
\end{align*}
\]

- In matrix notation

\[
\begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} u \\ v \end{pmatrix} = \begin{pmatrix} t_x \\ t_y \end{pmatrix}
\]

where

\[
\begin{pmatrix} x \\ y \end{pmatrix}, \quad \begin{pmatrix} u \\ v \end{pmatrix}, \quad \begin{pmatrix} t_x \\ t_y \end{pmatrix}
\]
Scaling

- **Scaling** is defined by

\[
\begin{align*}
x &= s_x u \\
y &= s_y v
\end{align*}
\]

and

\[
\begin{align*}
u &= x / s_x \\
v &= y / s_y
\end{align*}
\]

- **Matrix notation**

\[
x = Su, \quad u = S^{-1} x
\]

where

\[
S = \begin{bmatrix}
s_x & 0 \\
0 & s_y
\end{bmatrix}
\]

- If $s_x < 1$ and $s_y < 1$, this represents a minification or **shrinking**, if $s_x > 1$ and $s_y > 1$, it represents a magnification or **zoom**.

![Diagram showing scaling in the x and y directions with $s_x = 2, s_y = 1/2$.](image)
Rotation

• Rotation by an angle of $\theta$ is defined by

$$\begin{cases} x = u \cos \theta - v \sin \theta \\ y = u \sin \theta + v \cos \theta \end{cases} \text{ and } \begin{cases} u = x \cos \theta + y \sin \theta \\ v = -x \sin \theta + y \cos \theta \end{cases}$$

• In matrix format

$$x = R u, \quad u = R^T x$$

where

$$R = \begin{bmatrix} \cos \theta & -\sin \theta \\ \sin \theta & \cos \theta \end{bmatrix}$$

• $R$ is a unitary matrix: $R^{-1} = R^T$
Translation: \[ x(k, l) = k + 50; \quad y(k, l) = l; \]

Rotation: \[
\begin{align*}
x(k, l) &= (k - x_0)\cos(\theta) + (l - y_0)\sin(\theta) + x_0; \\
y(k, l) &= -(k - x_0)\sin(\theta) + (l - y_0)\cos(\theta) + y_0;
\end{align*}
\]

\[ x_0 = y_0 = 256.5 \text{ the center of the image A, } \theta = \pi/6 \]

By Onur Guleyuz
A geometric transformation refers to a combination of translation, scaling, and rotation, with a general form of

\[ x = RS(u + t) = Au + b, \]
\[ u = A^{-1}(x - b) = A^{-1}x + c, \]

with \( A = RS, b = RSt, c = -t. \)

Note that interchanging the order of operations will lead to different results.
Affine Mapping

- All possible geometric transformations are special cases of the **Affine Mapping**:

  \[
  \begin{align*}
  x &= a_0 + a_1 u + a_2 v \\
  y &= b_0 + b_1 u + b_2 v
  \end{align*}
  \]
  \[x = Au + b\]

  \[
  A = \begin{bmatrix}
  a_1 & a_2 \\
  b_1 & b_2
  \end{bmatrix}, \quad b = \begin{bmatrix}
  a_0 \\
  b_0
  \end{bmatrix}
  \]

- When \( A \) is a orthonormal matrix, it corresponds to a rotation matrix, and the corresponding affine mapping reduces to a geometric mapping.
Affine Mapping Properties

- Mapping straight lines to straight lines
- The mapping between two arbitrary triangles can be captured by an affine mapping
  - Affine mapping coefficients can be uniquely determined from displacements of 3 vertices
How to Determine Affine Parameters from Vertex Correspondences?

• Go through in class
Matlab Functions

- T = MAKETFORM('affine', U, X) builds a TFORM struct for a two-dimensional affine transformation that maps each row of U to the corresponding row of X. U and X are each 3-by-2 and define the corners of input and output triangles. The corners may not be collinear.
- Example
- -------
- Create an affine transformation that maps the triangle with vertices (0,0), (6,3), (-2,5) to the triangle with vertices (-1,-1), (0,-10), (4,4):
- 
  u = [ 0 6 -2]';
  v = [ 0 3 5]';
  x = [-1 0 4]';
  y = [-1 -10 4]';
  tform = maketform('affine',[u v],[x y]);
• $G = \text{MAKETFORM('affine',T)}$ builds a TFORM struct $G$ for an N-dimensional affine transformation. $T$ defines a forward transformation such that $\text{TFORMFWD}(U,T)$, where $U$ is a 1-by-N vector, returns a 1-by-N vector $X$ such that $X = U \ast T(1: N, 1: N) + T(N+1, 1: N).T$ has both forward and inverse transformations. $N=2$ for 2D image transformation.

In MATLAB notation

$$T = \begin{bmatrix} a_1 & b_1 & 0 \\ a_2 & b_2 & 0 \\ a_0 & b_0 & 1 \end{bmatrix} = \begin{bmatrix} A^T & 0 \\ b^T & 1 \end{bmatrix}$$
• $B = \text{IMTRANSFORM}(A, \text{TFORM}, \text{INTERP})$ transforms the image $A$ according to the 2-D spatial transformation defined by $\text{TFORMB}$; $\text{INTERP}$ specifies the interpolation filter

• Example 1

• ---------
• Apply a horizontal shear to an intensity image.

•
• $I = \text{imread}('\text{cameraman.tif}')$;
• $tform = \text{maketform}(\text{'affine'},[1 \ 0 \ 0; .5 \ 1 \ 0; 0 \ 0 \ 1])$;
• $J = \text{imtransform}(I, tform)$;
• $\text{figure, imshow}(I), \text{figure, imshow}(J)$

• Show in class
Horizontal Shear Example

\[
\text{tform} = \text{maketform('affine',[1 0 0; 0.5 1 0; 0 0 1]);}
\]

In MATLAB, ‘affine’ transform is defined by:

\[
[a1,b1,0;a2,b2,0;a0,b0,1]
\]

With notation used in this lecture note

\[
A = \begin{bmatrix} 1 & 0.5 \\ 0 & 1 \end{bmatrix}, \quad b = \begin{bmatrix} 0 \\ 0 \end{bmatrix}
\]

Note in this example, first coordinate indicates horizontal position, second coordinate indicates vertical position.
Bilinear Mapping

\[
\begin{cases}
x = a_0 + a_1u + a_2v + a_3uv \\
y = b_0 + b_1u + b_2v + b_3uv
\end{cases}
\]

- Mapping a square to a quadrangle.
- More generally mapping a quadrangle to another quadrangle.
- 8 parameters.
- Can be completely determined from how the four corners moved.
How to determine the bilinear coefficients from the vertex correspondences?
Different Types of Mapping

Two features of projective mapping:
• Chirping: increasing perceived spatial frequency for far away objects
• Converging (Keystone): parallel lines converge in distance
Polynomial Warping

- The polynomial warping includes all deformations that can be modeled by polynomial transformations:

\[
\begin{align*}
x &= a_0 + a_1 u + a_2 v + a_3 uv + a_4 u^2 + a_5 v^2 + \cdots \\
y &= b_0 + b_1 u + b_2 v + b_3 uv + b_4 u^2 + b_5 v^2 + \cdots 
\end{align*}
\]

- Includes affine and bilinear mapping as special cases
Image Warping by Forward Mapping

- Mapping image \( f(u, v) \) to \( g(x, y) \) based on a given mapping function: \( x(u, v), y(u, v) \).

- **Forward Mapping**
  - For each point \( (u, v) \) in the original image, find the corresponding position \( (x, y) \) in the deformed image by the forward mapping function, and let \( g(x,y) = f(u,v) \).
  - What if the mapped position \( (x,y) \) is not an integer sample in the desired image?
Image Warping by Inverse Mapping

- For each point \((x, y)\) in the image to be obtained, find its corresponding point \((u, v)\) in the original image using the inverse mapping function, and let \(g(x, y) = f(u, v)\).
- What if the mapped point \((u, v)\) is not an integer sample?
  - Interpolate from nearby integer samples!

P’ will be interpolated from \(P_1, P_2, P_3,\) and \(P_4\)
**Interpolation Method**

- **Nearest neighbor:**
  - Round \((u,v)\) to the nearest integer samples
- **Bilinear interpolation:**
  - find four integer samples nearest to \((u,v)\), apply bilinear interpolation
- **Other higher order interpolation methods can also be used**
  - Requiring more than 4 nearest integer samples!
How to find inverse mapping

• Invert the forward mapping

\[
\text{If } \quad x = Au + b \\
\text{Then } \quad u = A^{-1}(x - b)
\]

• Directly finding inverse mapping from point correspondence
MATLAB function: interp2

- \( Z_I = \text{INTERP2}(X,Y,Z,X_I,Y_I, \text{METHOD}) \) interpolates to find \( Z_I \), the values of the underlying 2-D function \( Z \) at the points in matrices \( X_I \) and \( Y_I \).
  - Matrices \( X \) and \( Y \) specify the points at which the data \( Z \) is given.
  - METHOD specifies interpolation filter
    - 'nearest' - nearest neighbor interpolation
    - 'linear' - bilinear interpolation
    - 'spline' - spline interpolation
    - 'cubic' - bicubic interpolation as long as the data is uniformly spaced, otherwise the same as 'spline'
Using ‘interp2’ to realize image warping

- Use inverse mapping
- Step 1: For all possible pixels in output image \((x,y)\), find corresponding points in the input image \((u,v)\)
  - \((X,Y) = \text{meshgrid}(1:M,1:N)\)
  - Apply inverse mapping function to find corresponding \((u,v)\), for every \((x,y)\), store in \((UI,VI)\)
    - Can use \text{tforminv}(\ ) function if you derived the transformation using \text{maketform}().
    - Or write your own code using the specified mapping
- Step 2: Use \text{interp2} to interpret the value of the input image at \((UI,VI)\) from their values at regularly sampled points \((U,V)\)
  - \((U,V) = \text{meshgrid}(1:M,1:N)\)
  - \text{Outimg}=\text{interp2}(U,V,\text{inimg},UI,VI,’linear’);
MATLAB function for image warping

- B = IMTRANSFORM(A,TFORM, INTERP) transforms the image A according to the 2-D spatial transformation defined by TFORM
- INTERP specifies the interpolation filter
- Example 1
  - ---------
  - Apply a horizontal shear to an intensity image.
  -
  - I = imread('cameraman.tif');
  - tform = maketform('affine',[1 0 0; .5 1 0; 0 0 1]);
  - J = imtransform(I,tform);
  - figure, imshow(I), figure, imshow(J)
Horizontal Shear Example

tform = maketform('affine',[1 0 0; .5 1 0; 0 0 1]);
In MATLAB, ‘affine’ transform is defined by:
[a1,b1,0;a2,b2,0;a0,b0,1]

With notation used in this lecture note
\[
A = \begin{bmatrix}
  1 & 0.5 \\
  0 & 1 \\
\end{bmatrix}, \quad b = \begin{bmatrix}
  0 \\
  0 \\
\end{bmatrix}
\]

Note in this example, x, u indicates vertical position, y, v indicate horizontal position
Two-Pass Mapping

- The idea is to transform each row of \( f(u, v) \) first, to obtain an intermediate image, \( f'(x',y') \), and then each column of this intermediate image is transformed to obtain the final image \( g(x, y) \).

\[
\begin{bmatrix}
  x \\
  y
\end{bmatrix} =
\begin{bmatrix}
  \cos \theta & -\sin \theta \\
  \sin \theta & \cos \theta
\end{bmatrix}
\begin{bmatrix}
  u \\
  v
\end{bmatrix}
\]

Ex: rotation

Step 1: Fixed each row \( x'=u \), find \( y'=usin\theta+vcos\theta \)

\( f'(x', y')=f(u,v) \)

Step2: Fix each column \( y=y' \) find \( x=ucos\theta-vsin\theta \)

\[
\begin{align*}
  u &= x' \\
  v &= \frac{y'-u\sin\theta}{\cos\theta} = \frac{y'-x'sin\theta}{\cos\theta}
\end{align*}
\]

\[
\Rightarrow x = x'\cos\theta - \frac{\sin\theta}{\cos\theta} (y'-x'sin\theta) = \frac{x'-y'sin\theta}{cos\theta}
\]
Example of Image Warping (1)

Wave 1:
\[ x(u,v) = u + 20 \sin\left(\frac{2\pi v}{128}\right) \]
\[ y(u,v) = v \]

Wave 2:
\[ x(u,v) = u + 20 \sin\left(\frac{2\pi u}{30}\right) \]
\[ y(u,v) = v \]

By Onur Guleyuz
Example of Image Warping (2)

WARP

\[ x(u, v) = \text{sign}(u - x_0) \times (u - x_0)^2 / x_0 + x_0; \]
\[ y(u, v) = v \]

SWIRL

\[ x(u, v) = (u - x_0) \cos(\theta) + (v - y_0) \sin(\theta) + x_0; \]
\[ y(u, v) = -(u - x_0) \sin(\theta) + (v - y_0) \cos(\theta) + y_0; \]
\[ r = ((u - x_0)^2 + (v - y_0)^2)^{1/2}, \theta = \pi r / 512. \]

By Onur Guleyuz
Image Registration

• Suppose we are given two images taken at different times of the same object. To observe the changes between these two images, we need to make sure that they are aligned properly. To obtain this goal, we need to find the correct mapping function between the two. The determination of the mapping functions between two images is known as the registration problem.

• Once the mapping function is determined, the alignment step can be accomplished using the warping methods.
How to find the mapping function?

• Assume the mapping function is a polynomial of order N

• Step 1: Identify \( K \geq N \) corresponding points between two images, i.e.

\[
(u_i, v_i) \leftrightarrow (x_i, y_i), i = 1, 2, ..., K.
\]

• Step 2: Determine the coefficients \( a_i, b_i, \) \( i = 0, ..., N-1 \) by solving

\[
\begin{align*}
    x(u_i, v_i) &= a_0 + a_1 u_i + a_2 v_i + \cdots = x_i, \\
    y(u_i, v_i) &= b_0 + b_1 u_i + b_2 v_i + \cdots = y_i,
\end{align*}
\]

• How to solve this?
How to Solve the Previous Equations?

• Convert to matrix equation:

\[
Aa = x, \quad Ab = y
\]

where

\[
A = \begin{bmatrix}
1 & u_1 & v_1 & \cdots \\
1 & u_2 & v_2 & \cdots \\
\vdots & \vdots & \vdots & \ddots \\
1 & u_K & v_K & \cdots \\
\end{bmatrix},
\quad a = \begin{bmatrix}
a_0 \\
a_1 \\
\vdots \\
a_{N-1} \\
\end{bmatrix},
\quad b = \begin{bmatrix}
b_0 \\
b_1 \\
\vdots \\
b_{N-1} \\
\end{bmatrix},
\quad x = \begin{bmatrix}
x_1 \\
x_2 \\
\vdots \\
x_K \\
\end{bmatrix},
\quad y = \begin{bmatrix}
y_1 \\
y_2 \\
\vdots \\
y_K \\
\end{bmatrix}
\]

If \( K = N \), and the matrix \( A \) is non-singular, then

\[
a = A^{-1}x, \quad b = A^{-1}y
\]

If \( K > N \), then we can use a least square solution

\[
a = (A^T A)^{-1} A^T x, \quad b = (A^T A)^{-1} A^T y
\]

If \( K < N \), or \( A \) is singular, then more corresponding feature points must be identified.
Examples

- If we want to use an affine mapping to register to images, we need to find 3 or more pairs of corresponding points.
- If we have only 3 pairs, we can solve the mapping parameters exactly as before.
- If we have more than 3 pairs, these pairs may not all be related by an affine mapping. We find the “least squares fit” by solving an over-determined system of equations.
Example
MATLAB function: cp2tform()

TFORM=CP2TFORM(INPUT_POINTS,BASE_POINTS,TRANSFORM_TYPE)

• returns a TFORM structure containing a spatial transformation.
• INPUT_POINTS is an M-by-2 double matrix containing the X and Y coordinates of control points in the image you want to transform.
• BASE_POINTS is an M-by-2 double matrix containing the X and Y coordinates of control points in the base image.
• TRANSFORMTYPE can be 'nonreflective similarity', 'similarity', 'affine', 'projective', 'polynomial', 'piecewise linear' or 'lwm'.
How to find corresponding points in two images?

• Which points to select in one image (image 1)?
  – Ideally choose “interesting points”: corners, special features
  – Can also use points over a regular grid

• How to find the corresponding point in the other image (image 2)?
  – Put a small block around the point in image 1
  – Find a block in image 2 that matches the block pattern the best
  – Exhaustive search within a certain range.
Exhaustive Block Matching Algorithm (EBMA)

Target frame

$R_x$

$R_y$

$B'_m$

Best match

$B_m$

Current block

Anchor frame

©Yao Wang, 2003
MATLAB Example

- Register an aerial photo to an orthophoto.

```
unregistered = imread('westconcordaerial.png');
figure, imshow(unregistered)
figure, imshow('westconcordorthophoto.png')
load westconcordpoints % load some points that were already picked

t_concord = cp2tform(input_points,base_points,'projective');
info = imfinfo('westconcordorthophoto.png');
registered = imtransform(unregistered,t_concord,...
    'XData',[1 info.Width], 'YData',[1 info.Height]);
figure, imshow(registered)
```
Image Morphing

- Image morphing has been widely used in movies and commercials to create special visual effects. For example, changing a beauty gradually into a monster.
- The fundamental techniques behind image morphing is image warping.
- Let the original image be $f(\mathbf{u})$ and the final image be $g(\mathbf{x})$. In image warping, we create $g(\mathbf{x})$ from $f(\mathbf{u})$ by changing its shape. In image morphing, we use a combination of both $f(\mathbf{u})$ and $g(\mathbf{x})$ to create a series of intermediate images.
Examples of Image Morphing

Cross Dissolve

\[ I(t) = (1-t)*S + t*T \]

Mesh based

Image Morphing Method

• Suppose the mapping function between the two end images is given as \( x = u + d(u) \). \( d(u) \) is the displacement between corresponding points in these two images.

• In image morphing, we create a series of images, starting with \( f(u) \) at \( k = 0 \), and ending at \( g(x) \) at \( k = K \). The intermediate images are a linear combination of the two end images:

\[
h_k(u + s_k d) = (1 - s_k) f(u) + s_k g(u + d(u)), \quad k = 0, 1, ..., K,
\]

where \( s_k = k / K \).
MATLAB function for selecting control points

- CPSELECT(INPUT,BASE) returns control points in CPSTRUCT. INPUT is the image that needs to be warped to bring it into the coordinate system of the BASE image.
- Example
  - cpselect('westconcordaerial.png','westconcordorthophoto.png')
Demo

- Show work by senior students
Homework

• You are given two pictures of the same scene, taken at different times. In order to align the two pictures, you need to find a mapping function between the two pictures based on some common feature points. Suppose you were able to extract N (N>=3) feature points in both images that correspond to the same set of object features, with image coordinates given as (u_k,v_k) and (x_k,y_k),k=1,2,…,N. Also, suppose you want to use an affine mapping to approximate the actual unknown mapping function. How would you determine the affine mapping parameters?

• Suppose you want make a panoramic picture of a wide landscape out of two separately captured pictures with a certain overlap. Propose an algorithm for stitching up the two images to make a panorama. (list the steps involved).

• Computer assignment: Write a matlab program that implements rotation of an image by a certain angle, and apply it on a selected image. The rotation center should be the image center. Please note that you should not use the ‘imrotate()’ or ‘imtransform()’ function in MATLAB. You should write your own program, which can call “interp2()”.
Reading

• Prof. Yao Wang’s Lecture Notes, Chapter 10.
• R. Gonzalez, “Digital Image Processing,” Section 5.11