Geometric Transformation and Image Registration

WHAT WILL WE LEARN?

- What do geometric operations do?
- What are they used for?
- What are the techniques used to enlarge/reduce a digital image?
- What are the main interpolation methods used in association with geometric operations?
- What are affine transformations and how can they be performed using MATLAB?
- How can I rotate, flip, crop, or resize images in MATLAB?
- What is image registration and where is it used?

What do geometric operations do?

Geometric operations modify the geometry of an image,

by repositioning pixels in a constrained way.

What are they used for?

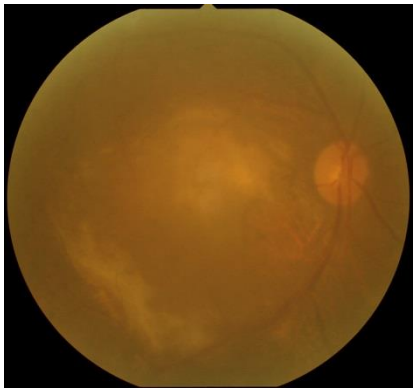
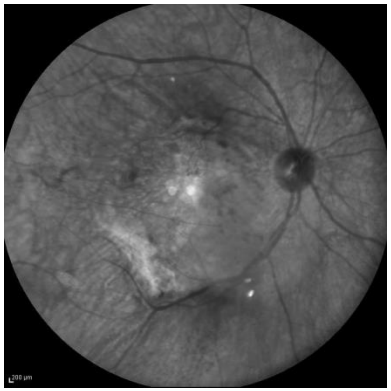
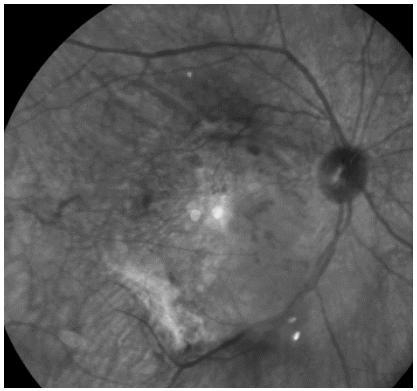
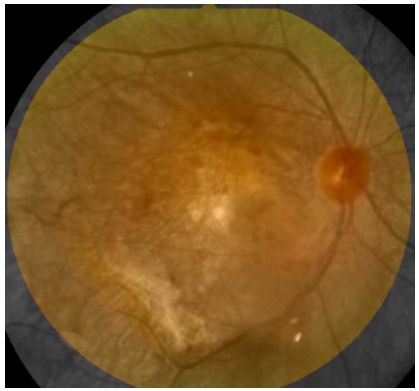
(1) Correcting geometric distortions introduced during the image acquisition process (e.g., due to the use of a fish-eye lens).

What are they used for?

(2) Creating special effects on existing images, such as twirling, bulging, or squeezing a picture of someone's face.

What are they used for?

- **Image registration**—the process of matching the common features of two or more images of the same scene, acquired from different viewpoints or using different equipment.

Two basic components of geometric transformation

1. ***Mapping Function*:** This is typically specified using a set of *spatial transformation equations* (and a procedure to solve them) .

2. ***Interpolation Methods*:** These are used to compute the new value of each pixel in the spatially transformed image.

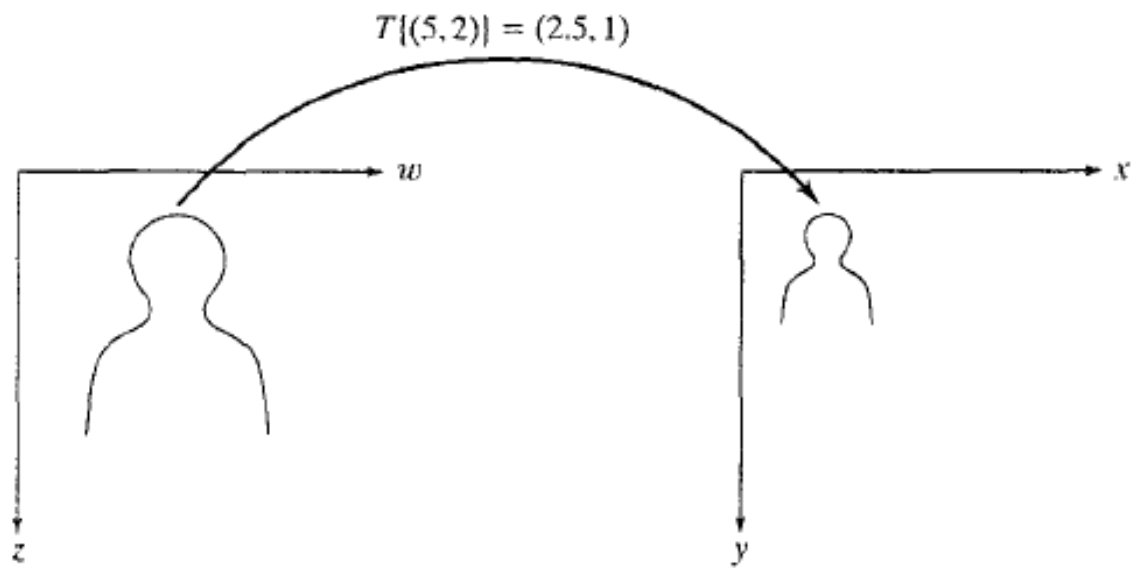
Geometric Spatial Transformations

Suppose that an image, *f*, defined over a *(w, z)* coordinate system, undergoes geometric distortion to produce an image, g, defined over an *(x, y)* coordinate system.

This transformation (of the coordinates) may be expressed as

$$(x, y) = T\{(w, z)\}$$

- For example,
  - if *(x, y) = T {(w, z)} = (w*/2, *z/2),*
  - Output image is simply as shrinking of *f* by half in both spatial dimensions, as illustrated in Figure.



$$x = T_w(w, z)$$

$$y = T_z(w, z)$$

การแปลงสัมพรรค

เมื่อฟังก์ชันส่งค่า T_w และ T_z สามารถเขียนอยู่ในรูปของฟังก์ชันพหุนาม (Polynomial) ของ w และ z ในกรณีที่ และ เป็นผลรวมเชิงเส้นของ x และ y จะเรียกว่า การแปลงสัมพรรค (Affine transformation) ที่กำหนดได้ดังนี้

$$x = t_{11}w + t_{12}z + t_{13}$$

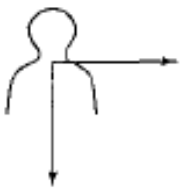
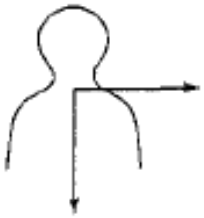
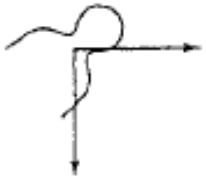
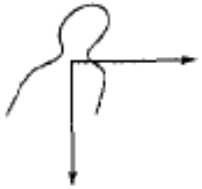
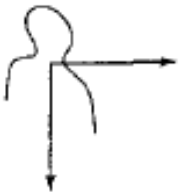
$$y = t_{21}w + t_{22}z + t_{23}$$

การแปลงสัมพรรค

One of the most commonly used forms of spatial transformations is the *affine transform* (Walberg [1990]). The affine transform can be written in matrix form as

$$\begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} t_{11} & t_{12} & t_{13} \\ t_{21} & t_{22} & t_{23} \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} w \\ z \\ 1 \end{bmatrix}$$

ชนิดของ affine transform

Type	Affine Matrix, T	Coordinate Equations	Diagram
Identity	$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$	$x = w$ $y = z$	
Scaling	$\begin{bmatrix} s_x & 0 & 0 \\ 0 & s_y & 0 \\ 0 & 0 & 1 \end{bmatrix}$	$x = s_x w$ $y = s_y z$	
Rotation	$\begin{bmatrix} \cos\theta & \sin\theta & 0 \\ -\sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix}$	$x = w\cos\theta - z\sin\theta$ $y = w\sin\theta + z\cos\theta$	
Shear (horizontal)	$\begin{bmatrix} 1 & 0 & 0 \\ \alpha & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$	$x = w + \alpha z$ $y = z$	
Shear (vertical)	$\begin{bmatrix} 1 & \beta & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$	$x = w$ $y = \beta w + z$	
Translation	$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ \delta_x & \delta_y & 1 \end{bmatrix}$	$x = w + \delta_x$ $y = z + \delta_y$	

tform structure

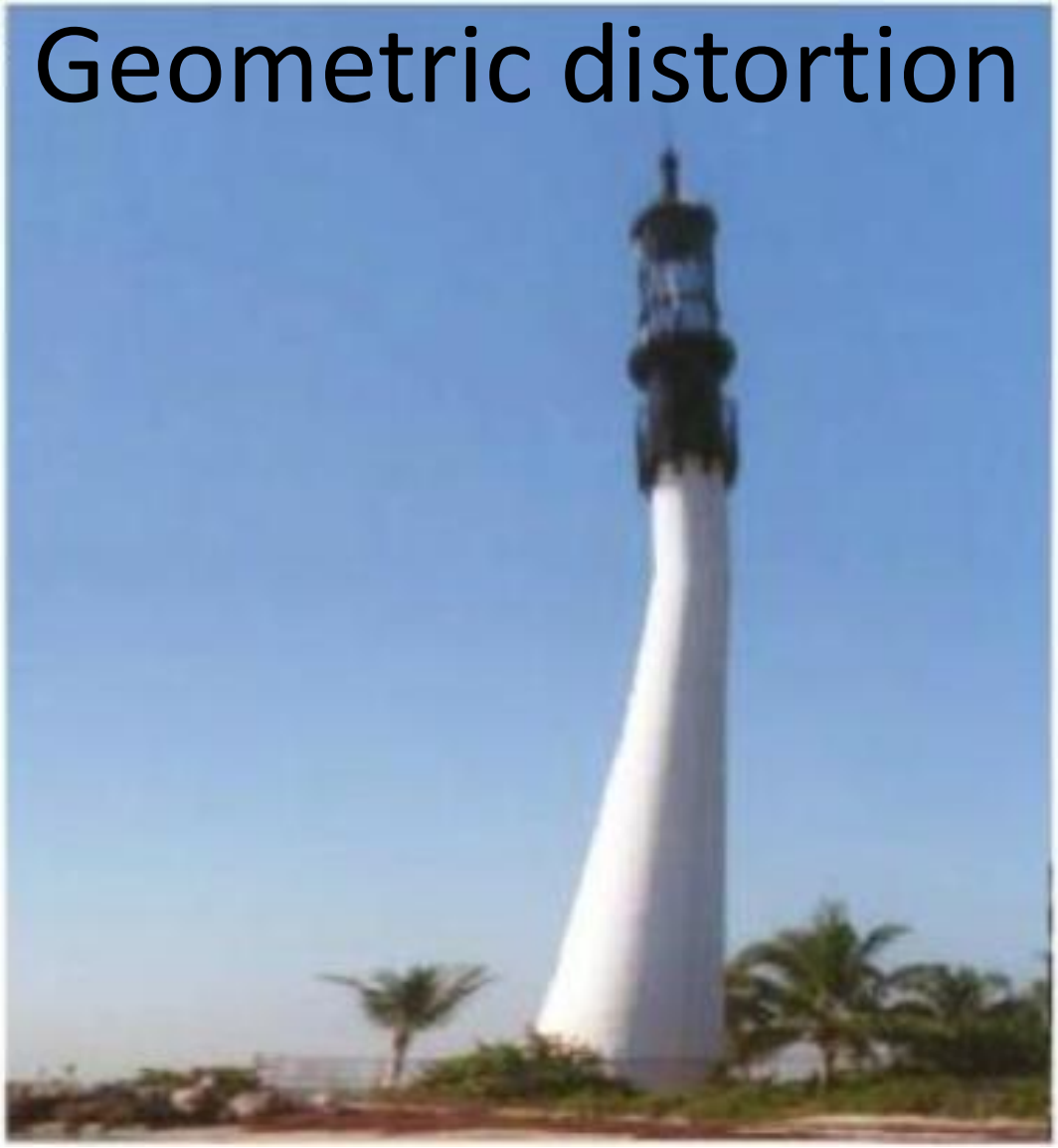
- IPT represents spatial transformations using a so-called *tform structure.*
- A method to create such a structure is by using function **maketform**, whose call syntax

  tform = maketform(transform_type, transform_parameters)

- The first input argument, transform_type, is one of these strings: 'affine’, 'projective', 'box', ‘composite' ,or ‘custom'.

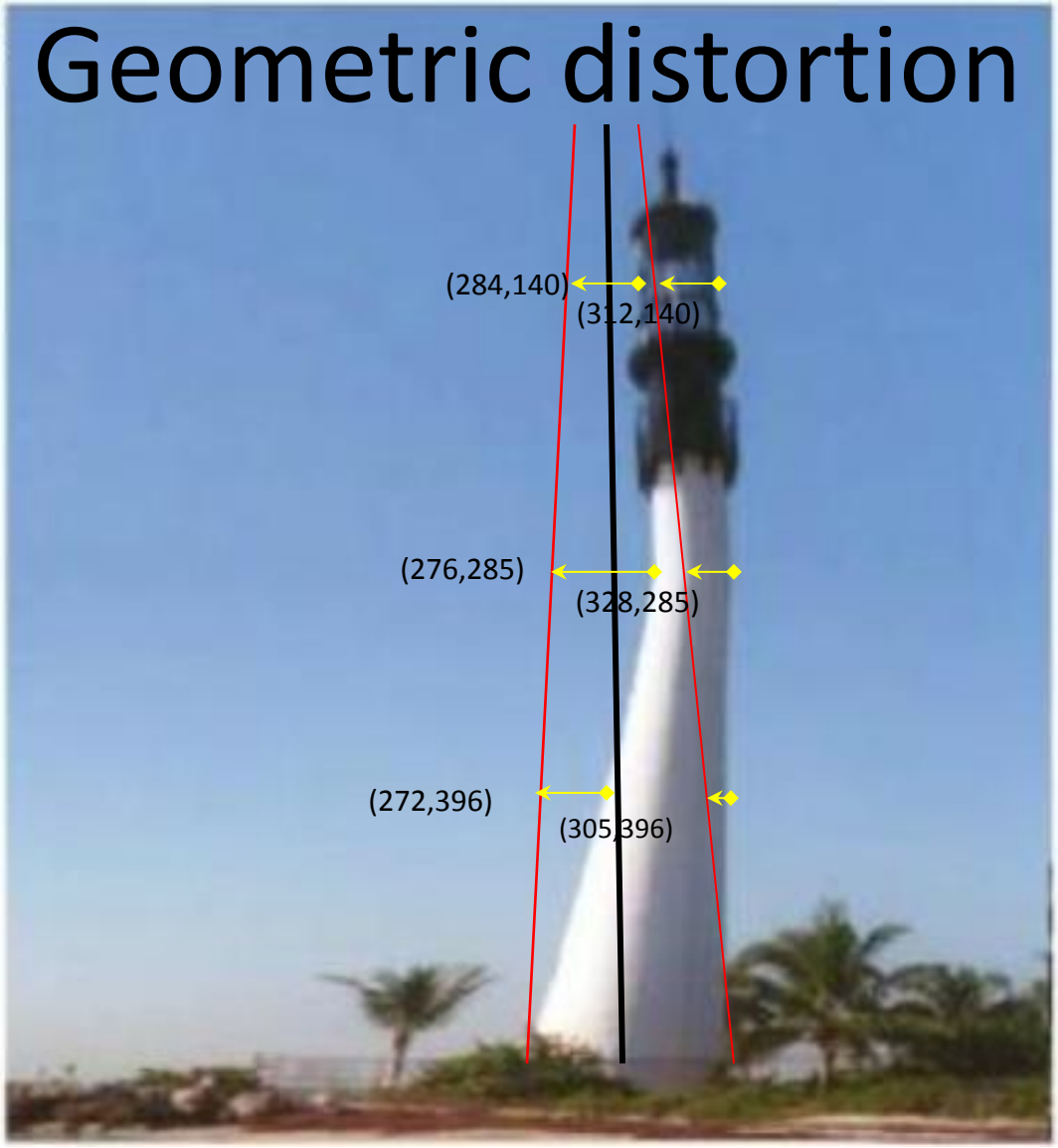
(a)

Geometric distortion



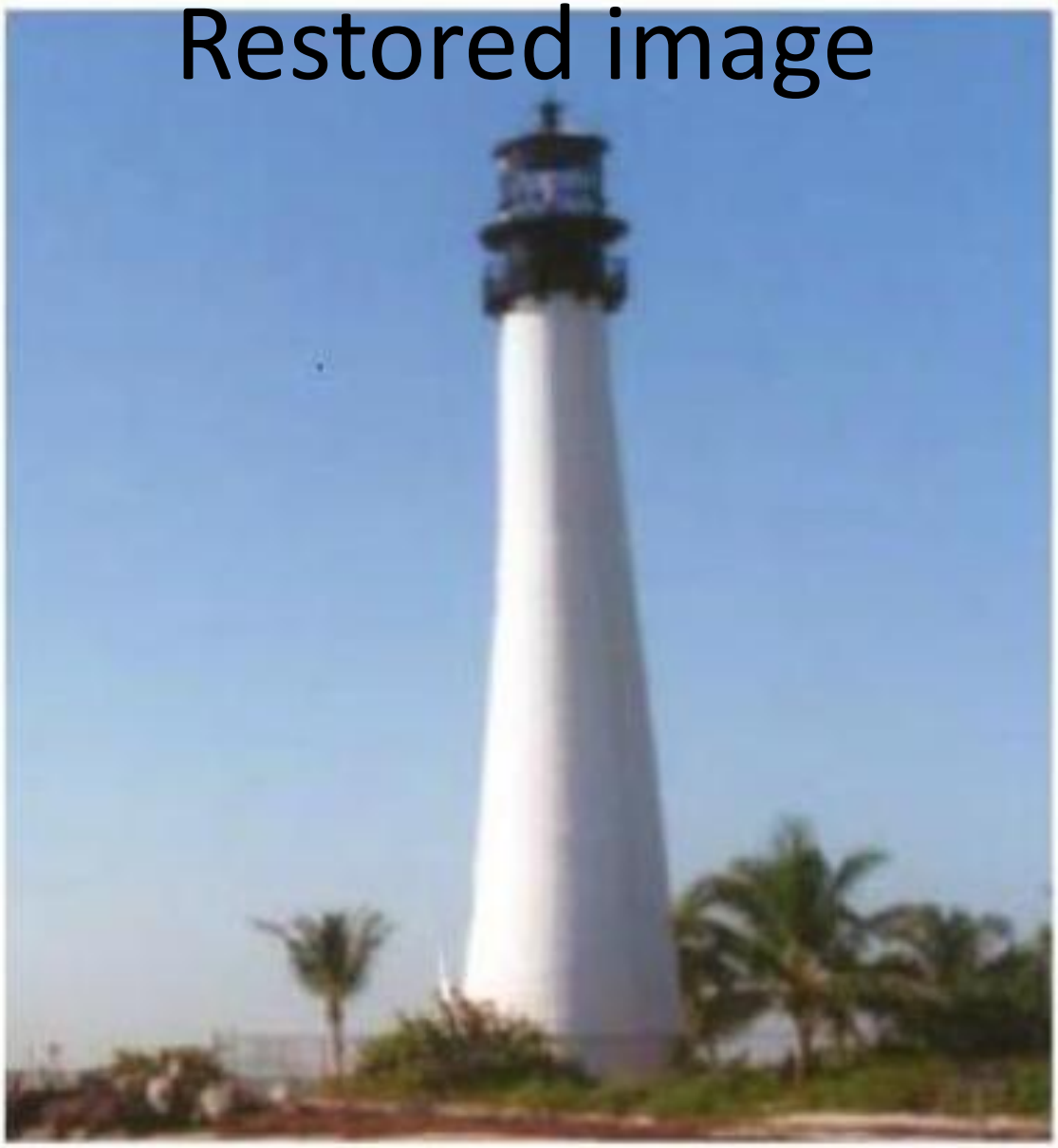
(a)

Geometric distortion



(b)

Restored image



tform_type ชนิด ‘custom’

Ex. การปรับมาตรในแนวนอนด้วย 3 และปรับแนวตั้งด้วย 2

$$(x, y) = T(w, z) = (3w, 2z)$$

$$(w, z) = T^{-1}(x, y) = (x/3, y/2)$$

มีซินแท็กซ์คือ xy = forward_fcn(wz, tdata)

```
>> forward_fcn = @(wz, tdata) [3*wz(:,1), 2*wz(:,2)]
```

มีซินแท็กซ์คือ wz = inverse_fn(xy, tdata)

```
>> inverse_fcn = @(xy, tdata) [xy(:,1)/3, xy(:,2)/2]
```

สร้างตัวแปรโครงสร้าง tform

```
>> tform1 = maketform('custom', 2, 2, forward_fcn, inverse_fcn, [])
```

จะได้โครงสร้างดังนี้

```
tform1 =
    ndims_in : 2
    ndims_out : 2
    forward_fcn : @(wz, tdata) [3*wz(: , 1) , 2*wz(: , 2) ]
    inverse_fcn : @(xy, tdata) [xy(:, 1)/3, xy(:, 2)/2 ]
    tdata : [ ]
```

ตัวอย่างคำสั่งต่อไปนี้เป็นการแปลงไปและแปลงกลับ

```
>> wz = [1 1; 3 2]
>> xy = tformfwd(wz, tform1)
xy =
        3  2
        9  4
>> wz2 = tforminv(xy, tform1)
wz2 =
        1  1
        3  2
```

$$(x, y) = T(w, z) = (3w, 2z)$$

$$\begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} 3 & 0 \\ 0 & 2 \end{bmatrix} \begin{bmatrix} w \\ z \end{bmatrix}$$

$$\begin{bmatrix} 3 \\ 2 \end{bmatrix} = \begin{bmatrix} 3 & 0 \\ 0 & 2 \end{bmatrix} \begin{bmatrix} 1 \\ 1 \end{bmatrix}, \quad \begin{bmatrix} 9 \\ 4 \end{bmatrix} = \begin{bmatrix} 3 & 0 \\ 0 & 2 \end{bmatrix} \begin{bmatrix} 3 \\ 2 \end{bmatrix}$$

$$\begin{bmatrix} 3 & 2 \end{bmatrix} = \begin{bmatrix} 1 & 1 \end{bmatrix} \begin{bmatrix} 3 & 0 \\ 0 & 2 \end{bmatrix}, \quad \begin{bmatrix} 9 & 4 \end{bmatrix} = \begin{bmatrix} 3 & 2 \end{bmatrix} \begin{bmatrix} 3 & 0 \\ 0 & 2 \end{bmatrix}$$

$$(w, z) = T^{-1}(x, y) = (x/y, y/2)$$

$$\begin{bmatrix} w \\ z \end{bmatrix} = \begin{bmatrix} 1/3 & 0 \\ 0 & 1/2 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$$

maketform

- For example, one way create an affine tform is to provide the T matrix directly, as in

```
>> T = [2 0 0; 0 3 0; 0 0 1];
>> tform = maketform('affine', T)

tform =

      ndims_in: 2
     ndims_out: 2
   forward_fcn: @fwd_affine
   inverse_fcn: @inv_affine
         tdata: [1x1 struct]
>> tform.tdata

ans =

      T: [3x3 double]
   Tinv: [3x3 double]
```

```
>> tform.tdata
ans =
      T: [3x3 double]
   Tinv: [3x3 double]

>> tform.tdata.T
ans =

    2    0    0
    0    3    0
    0    0    1

>> tform.tdata.Tinv
ans =

   0.5000         0         0
        0    0.3333         0
        0         0    1.0000
```

Forward/Inverse Transformation

- tformfwd computes the forward transformation, $T\{(w,z)\}$, syntax for tformfwd is XY = tformfwd(WZ, tform).

Ex

```
>> WZ =[1 1; 3 2];
>> XY = tformfwd(WZ, tform)

XY =

    2    3
    6    6
```

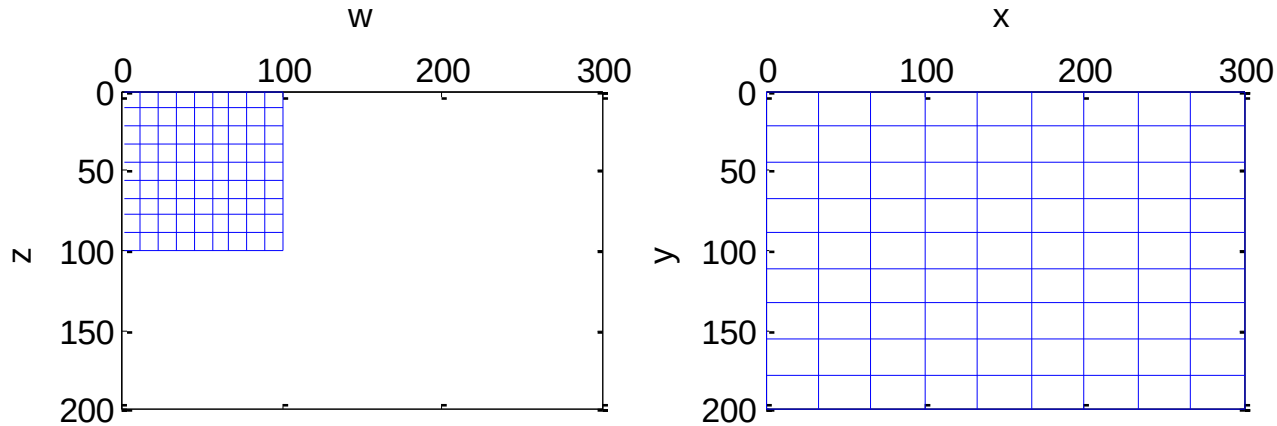
- tforminv computes the inverse transformation, $T^{-1}\{(w,z)\}$.

Ex

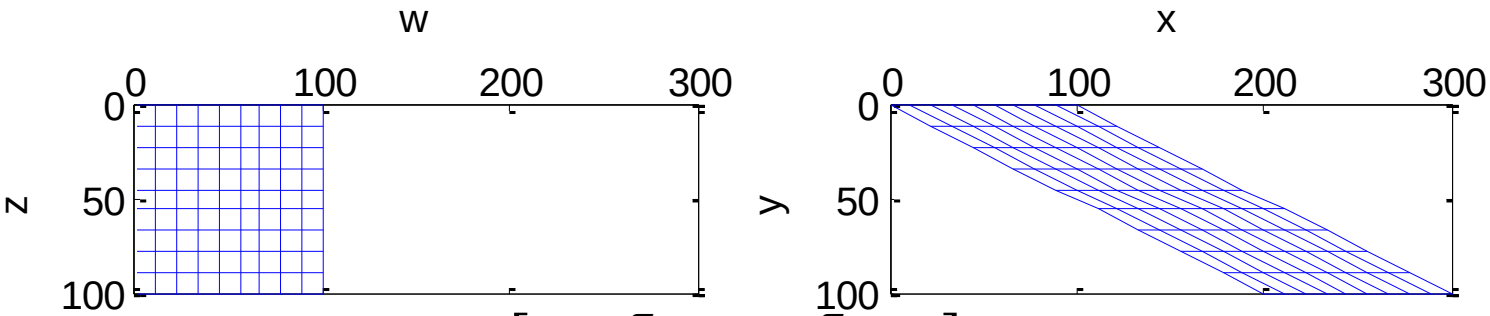
```
>> WZ1=tforminv(XY, tform)

WZ1 =

    1    1
    3    2
```

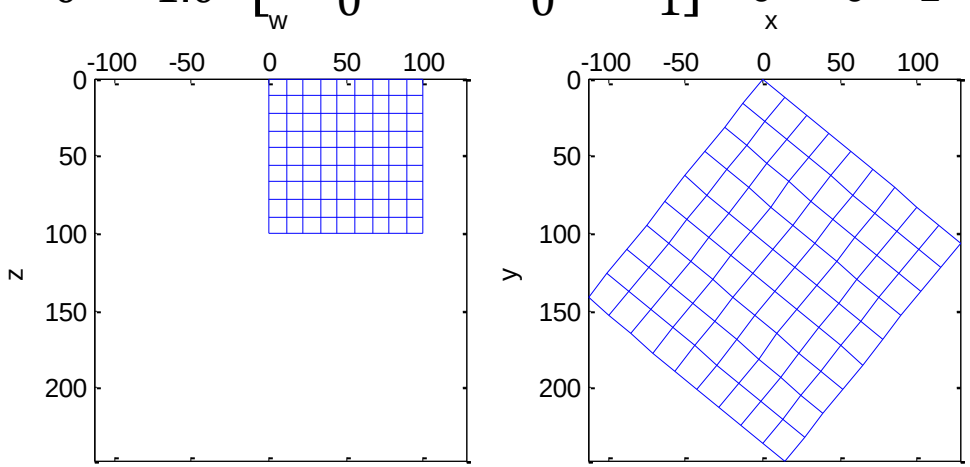


$$T=\begin{bmatrix}3 & 0 & 0\\ 0 & 2 & 0\\ 0 & 0 & 1\end{bmatrix}$$



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

$$\begin{bmatrix}1.5 & 0 & 0\\ 0 & 2.0 & 0\\ 0 & 0 & 1.0\end{bmatrix}\begin{bmatrix}\cos(\frac{\pi}{4}) & \sin(\frac{\pi}{4}) & 0\\ -\sin(\frac{\pi}{4}) & \cos(\frac{\pi}{4}) & 0\\ 0 & 0 & 1\end{bmatrix}\begin{bmatrix}1 & 0 & 0\\ 0.2 & 1 & 0\\ 0 & 0 & 1\end{bmatrix}=\begin{bmatrix}1.2728 & 1.0607 & 0\\ -1.1314 & 1.4142 & 0\\ 0 & 0 & 1\end{bmatrix}$$

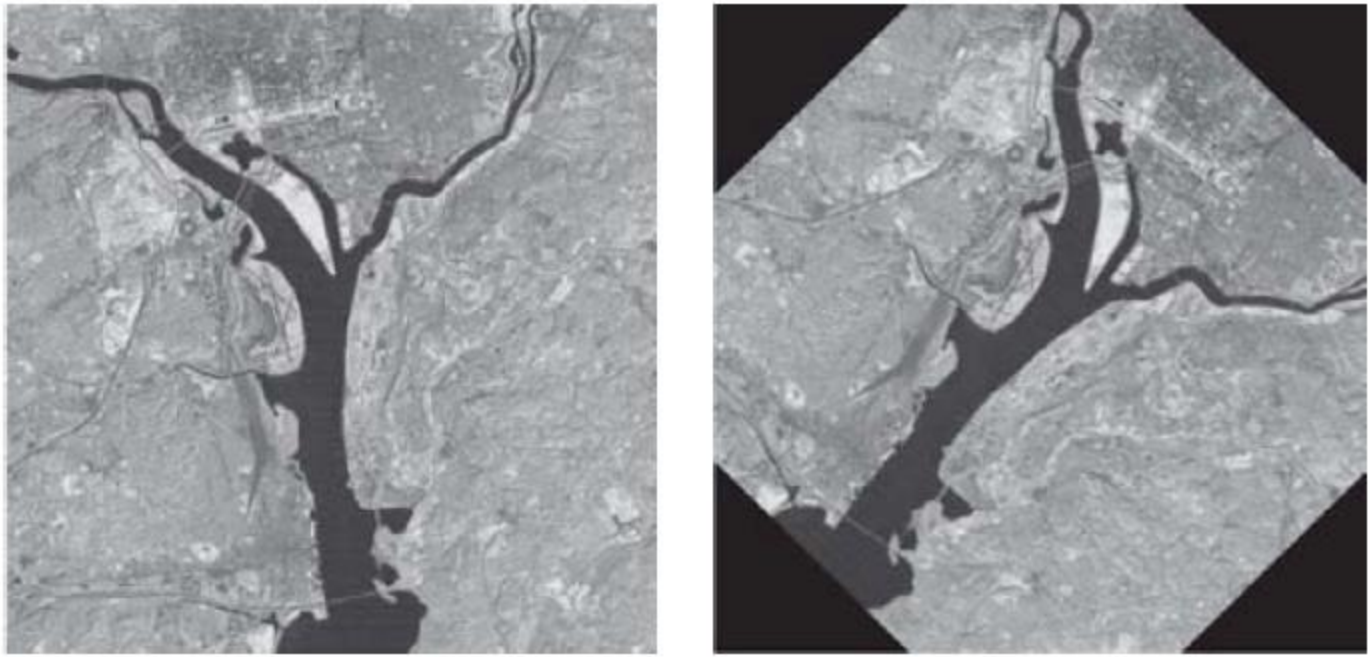


Applying Spatial Transformations to Images

Most computational methods for spatially transforming an image fall into one of two categories:

1. *forward mapping,*
2. *inverse mapping.*

- Methods based on forward mapping scan each input pixel in turn, copying its value into the output image at the location determined by $T\{(w, z)\}$.
- One problem with the forward mapping procedure is that two or more different pixels in the input image could be transformed into the same pixel in the output image, raising the question of how to combine multiple input pixel values into a single output pixel value.



(*a*) Original, 500 × 500 (*b*) Rotated, 500 × 500

imtransform uses inverse mapping instead. An inverse mapping procedure scans each output pixel in turn, computes the corresponding location in the input image using $T^{-1}\{(x, y)\}$, and interpolates among the nearest input image pixels to determine the output pixel value.

imtransform

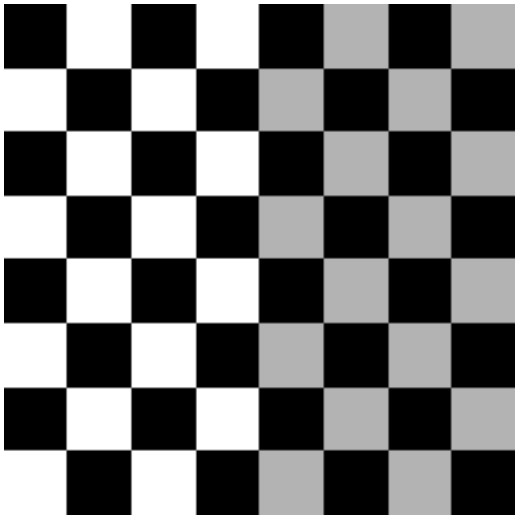
The basic calling syntax for imtransform is

g = imtransform(f, tform, interp)

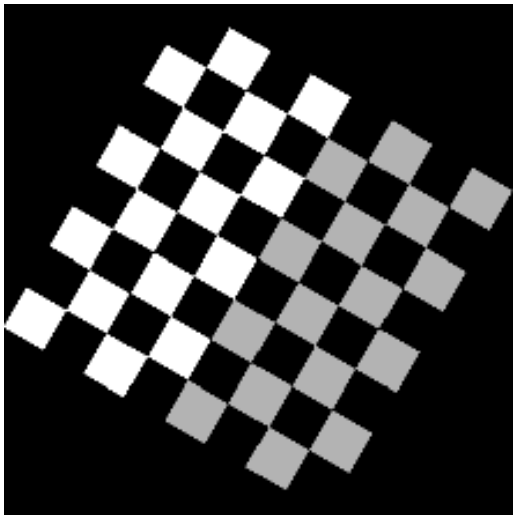
- where interp is a string that specifies how input image pixels are interpolated to obtain output pixels; interp can be either 'nearest', 'bilinear', or bicubic' .
- The interp input argument can be omitted, in which case it defaul to 'bilinear'.

Ex

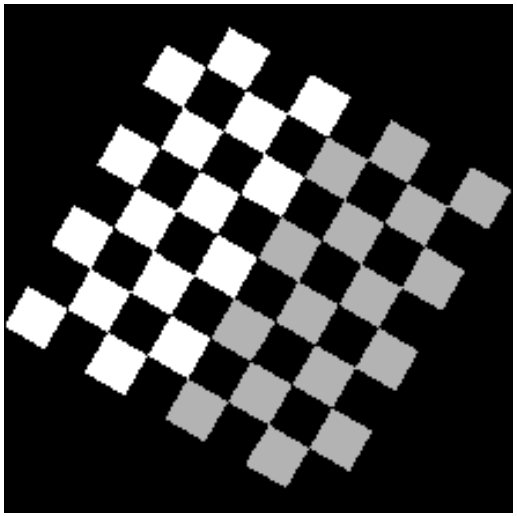
$$\mathbf{T} = \begin{bmatrix} s\cos\theta & s\sin\theta & 0 \\ -s\sin\theta & s\cos\theta & 0 \\ \delta_x & \delta_y & 1 \end{bmatrix}$$



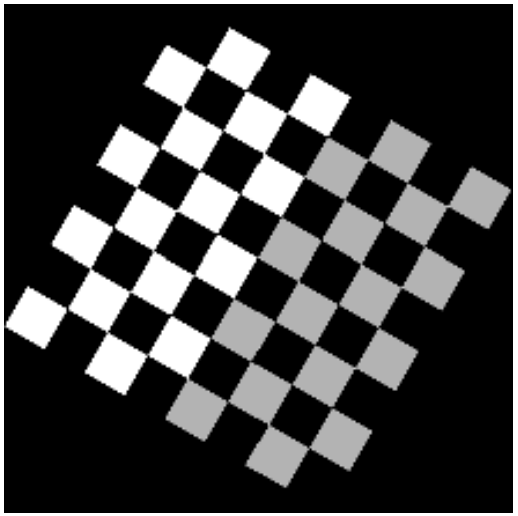
(a) Original image



(b) Linear conformal (bilinear)



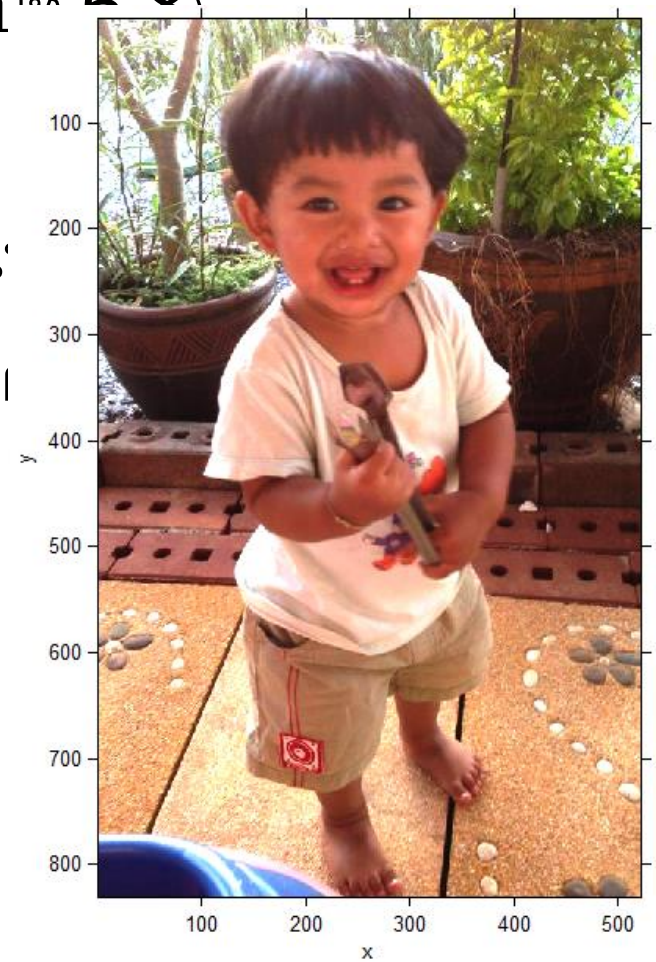
(c) Bicubic interpolation



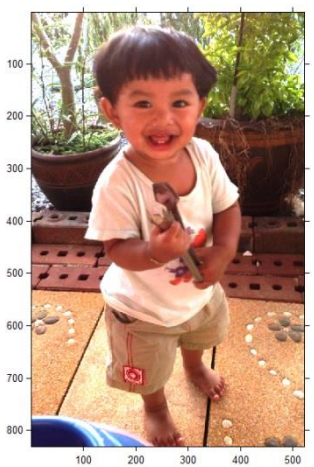
(d) Bicubic interpolation

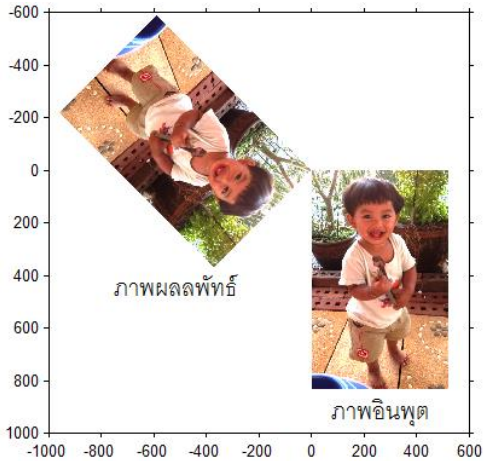
ระบบพิกัดภาพใน MATLAB

```
>> f = imread('PD02Exp.tif');
>> imshow(f)        %แสดงผลภาพ (รูปที่ 6.8)
>> axis on          %แสดงพิกัด
>> xlabel x         %แสดง x บนแกน
>> ylabel y         %แสดง y บนแนว
```

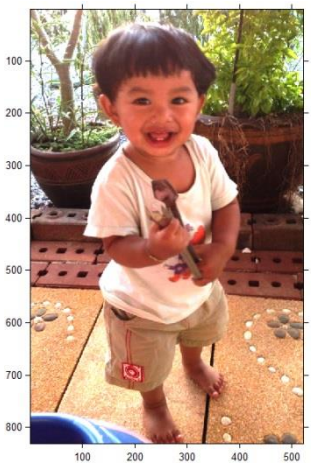


ระบบพิกัดภาพใน MATLAB






```
>> theta = 3*pi/4;
>> T = [cos(theta)  sin(theta) 0; -sin(theta) cos(theta) 0; 0 0 1];
>> tform = maketform('affine', T);
>> [g, xdata, ydata] = imtransform(f, tform, 'FillValue', 255);
>> imshow(f)
>> hold on
>> imshow(g, ‘XData’, xdata, ‘YData’, ydata);
```



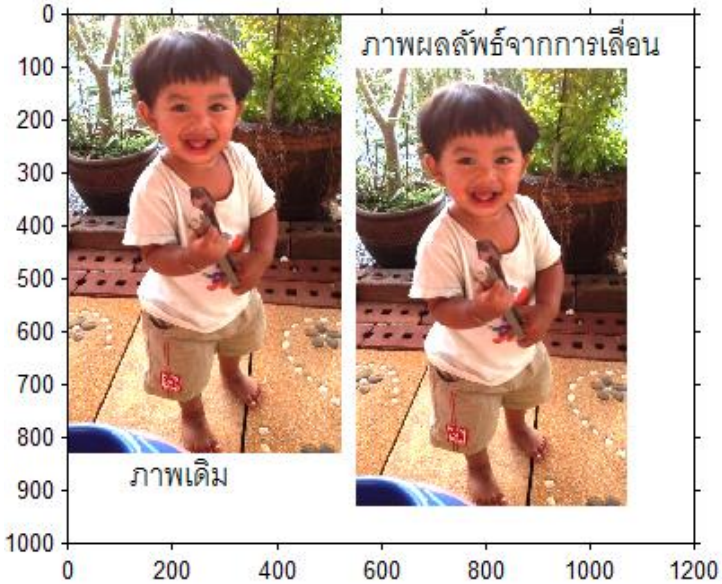


Image Registration

Geometric transformations are used frequently to perform *image registration,* a process that takes two images of the same scene and aligns them so they can be merged for visualization, or for quantitative comparison.

In the following sections, we discuss

1. spatial transformations and how to define and visualize them in MATLAB;
2. how to apply spatial transformation images; and
3. how to determine spatial transformations for use in image registration.

Image Registration

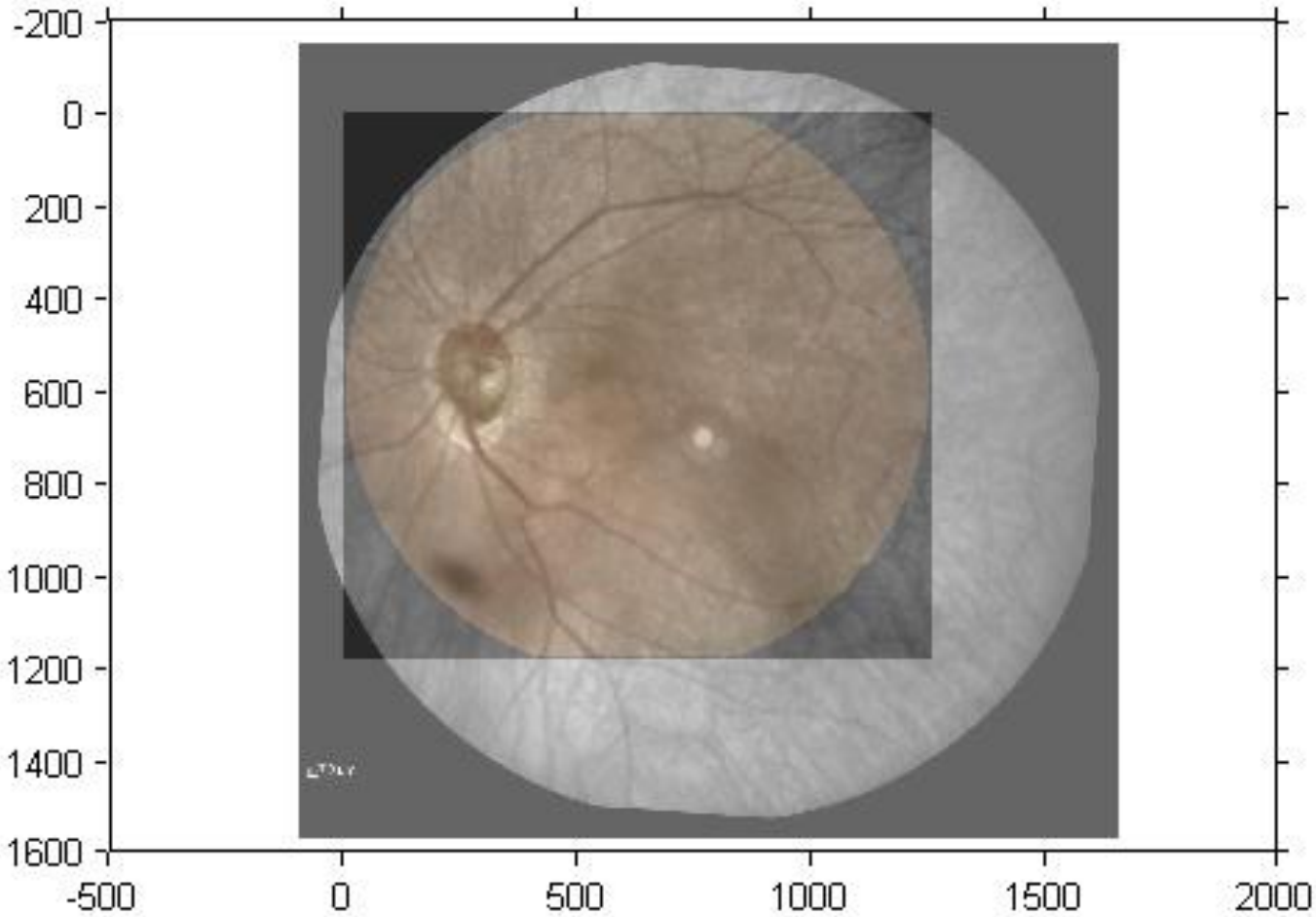
Image registration methods seek to align two images of the same scene.

- For example, it may be of interest to align two or more images taken at roughly the same time, but using different instruments, such as an MRI (magnetic resonance imaging) scan and a PET (positron emission tomography) scan.
- Or, perhaps the images were taken at different times using the same instrument, such satellite images of a given location taken several days, months, or even years apart.
- In either case, combining the images or performing quantitative analysis and comparisons requires compensating for geometric aberrations caused by differences in camera angle, distance, and orientation; sensor resolution; shift in subject position; and other factors.

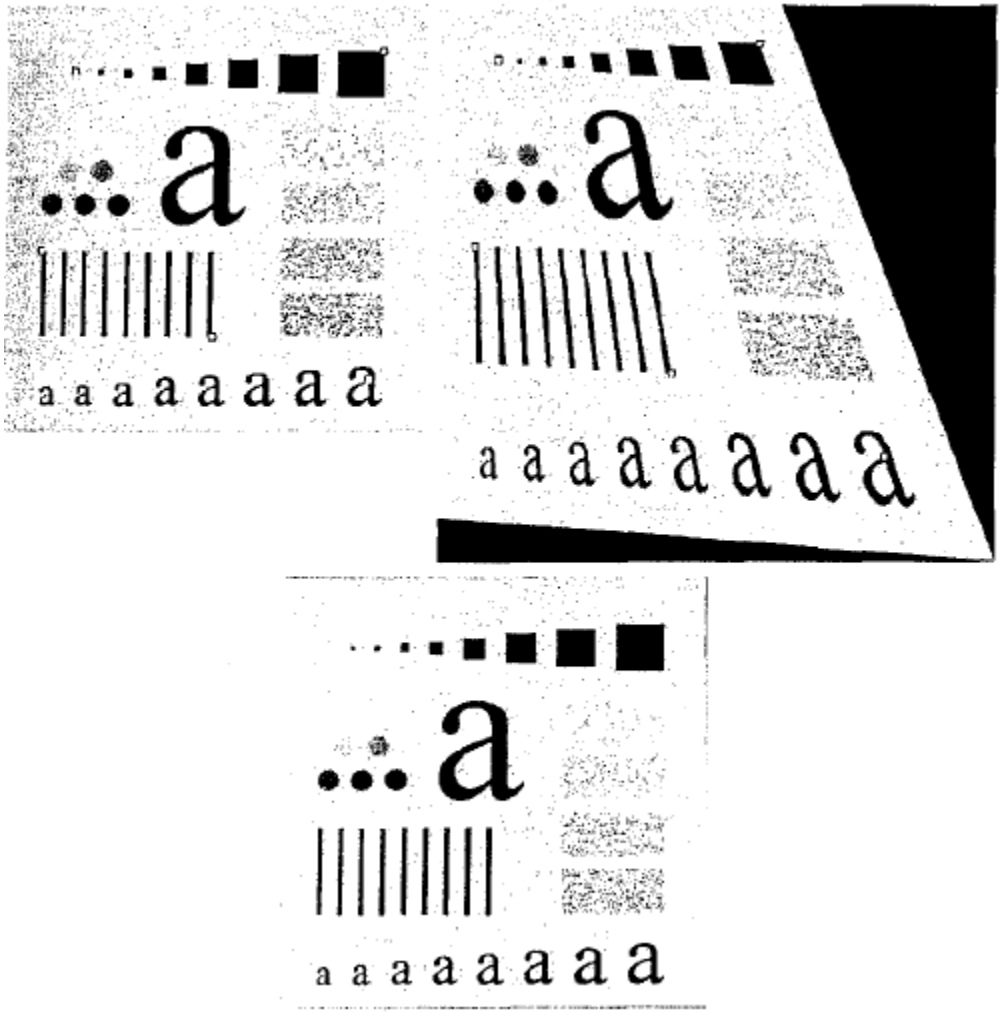
การกำหนดตำแหน่งโดยผู้ใช้

สร้างตัวแปรในการแปลง

```
>> tform = cp2tform(input_points, base_points, ‘affine’);
>> [x xdata, ydata] = imtransform(f, tform);
```



cp2tform



```
>> basepoints = [83 81; 450 56; 43 293; 249 392; 436 442];
>> inputpoints = [68 66; 375 47; 42 286; 275 434; 523 532];
>> tform = cp2tform(inputpoints, basepoints, 'projective');
>> gp = imtransform(g, tform, 'XData', [1 502], 'YData', [1 502]);
```

Transformation Type	Description	Functions
Affine	Combination of scaling, rotation, shearing, and translation. Straight lines remain straight and parallel lines remain parallel.	`maketform` `cp2tform`
Box	Independent scaling and translation along each dimension; a subset of affine.	`maketform`
Composite	A collection of spatial transformations that are applied sequentially.	`maketform`
Custom	User-defined spatial transform; user provides functions that define T and T^{-1}.	`maketform`
Linear conformal	Scaling (same in all dimensions), rotation, and translation; a subset of affine.	`cp2tform`
LWM	Local weighted mean; a locally-varying spatial transformation.	`cp2tform`
Piecewise linear	Locally varying spatial transformation.	`cp2tform`
Polynomial	Input spatial coordinates are a polynomial function of output spatial coordinates.	`cp2tform`
Projective	As with the affine transformation, straight lines remain straight, but parallel lines converge toward vanishing points.	`maketform` `cp2tform`