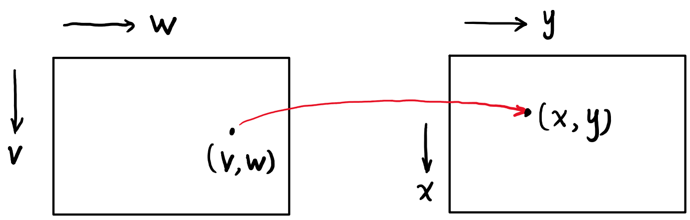
Lecture 9

Geometric Spatial Transformation



$$(x, y) = T(v, w)$$

T ↑ transformation

$$\underbrace{[x \;\; y \;\; 1]}_{1\times 3} = \underbrace{[v \;\; w \;\; 1]}_{1\times 3} \cdot \underbrace{T}_{3\times 3} = [v \;\; w \;\; 1]\begin{bmatrix} t_{11} & t_{12} & 0 \\ t_{21} & t_{22} & 0 \\ t_{31} & t_{32} & 1 \end{bmatrix}$$

Translation:

$$T = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 20 & 20 & 1 \end{bmatrix}$$

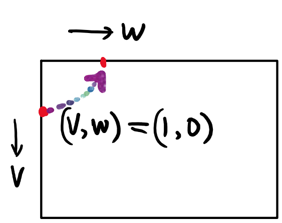
$$\underbrace{[x \;\; y \;\; 1]}_{1\times 3} = \underbrace{[v \;\; w \;\; 1]}_{1\times 3} \cdot \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 20 & 20 & 1 \end{bmatrix} = [v+20 \;\;\; w+20 \;\;\; 1]$$

Rotation:

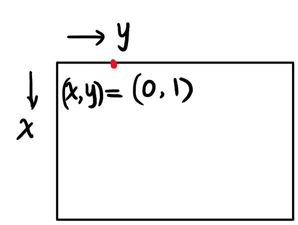
$$\begin{cases} x = v\cos\theta - w\sin\theta \\ y = v\sin\theta + w\cos\theta \end{cases} \Rightarrow \underbrace{[x \;\; y \;\; 1]}_{1\times 3} = \underbrace{[v \;\; w \;\; 1]}_{1\times 3} \cdot \begin{bmatrix} \cos\theta & \sin\theta & 0 \\ -\sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

e.g., $\theta = \frac{\pi}{2}$.

$$\begin{cases} x = -w \\ y = v \end{cases}$$



$\xrightarrow{T}$ $\begin{cases} x = 0 \\ y = 1 \end{cases}$



Counter-clockwise rotation by 90°.

imwarp

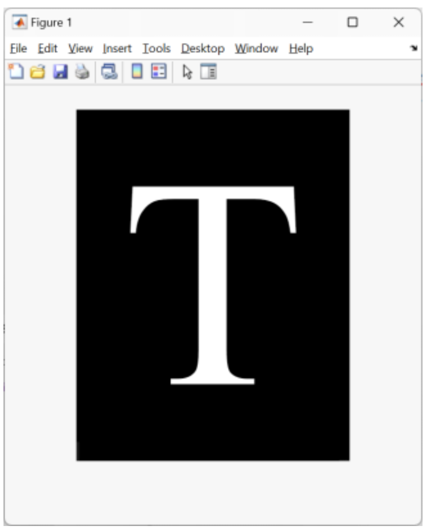
Apply geometric transformation to image

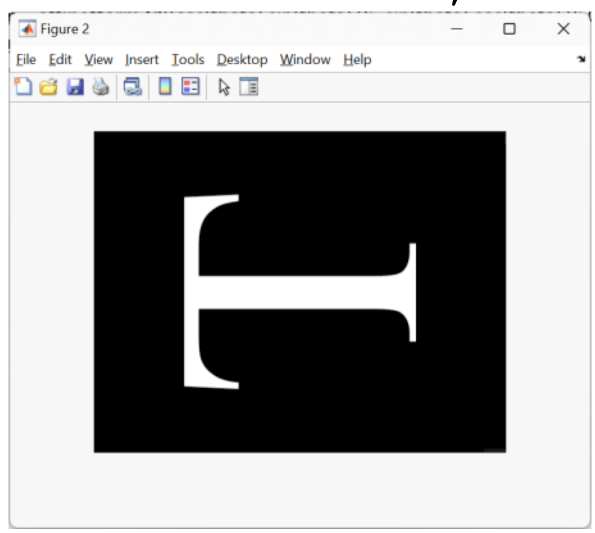
affinetform2d

2-D affine geometric transformation
Since R2022b

```
>> theta = pi/2;
>> I = imread('Fig0236(a)(letter_T).tif');
>> tform = affinetform2d([cos(theta) sin(theta) 0; -sin(theta) cos(theta) 0; 0 0 1]);
>> J = imwarp(I, tform);
>> figure; imshow(J)
```

counter - clockwise rotation by 90°





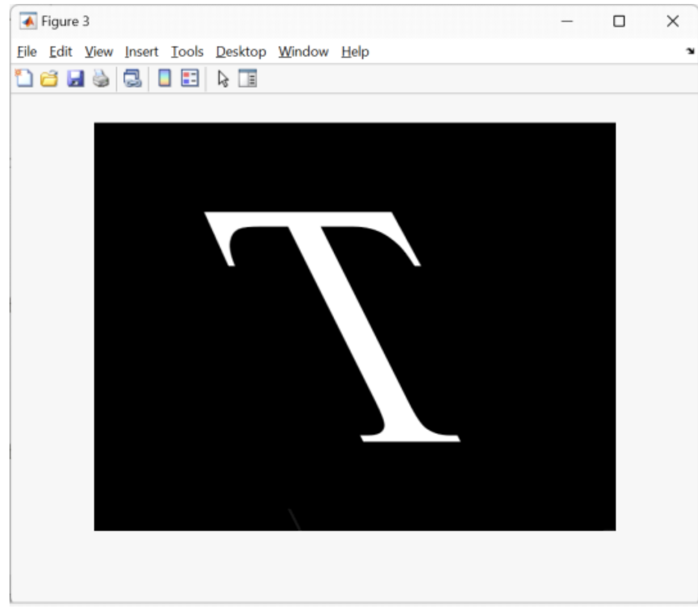
Horizontal Shear

$$x = v$$

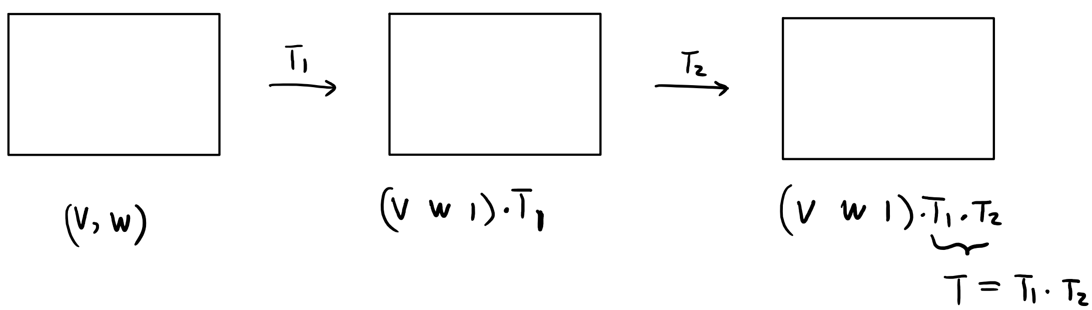
$$y = 0.5v + w \quad \Rightarrow \quad T = \begin{bmatrix} 1 & 0.5 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$$[x \;\; y \;\; 1] = [v \;\; w \;\; 1] \begin{bmatrix} t_{11} & t_{12} & 0 \\ t_{21} & t_{22} & 0 \\ t_{31} & t_{32} & 1 \end{bmatrix}$$

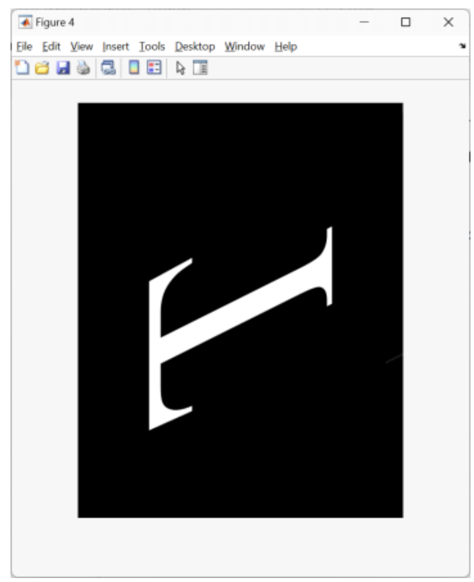
```
>> T = [1 0.5 0;0 1 0;0 0 1]

T =

    1.0000    0.5000         0
         0    1.0000         0
         0         0    1.0000

>> tform = affinetform2d(T);
>> J = imwarp(I, tform);
>> figure; imshow(J)
```



Sequence of transformations




```
>> T1 = [cos(theta) sin(theta) 0; -sin(theta) cos(theta) 0; 0 0 1];
>> T2 = [1 0.5 0;0 1 0;0 0 1];
>> T = T1*T2

T =

    0.0000    1.0000         0
   -1.0000   -0.5000         0
         0         0    1.0000

>> tform = affinetform2d(T);
J = imwarp(I, tform);
figure; imshow(J)
```



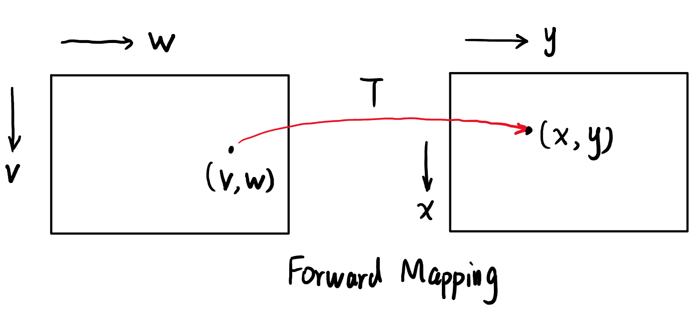
Forward vs. Inverse Mapping

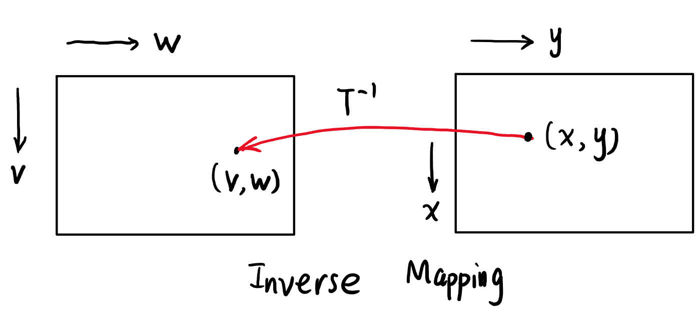
Inverse mapping

–Scanning the output pixel locations and, at each location, (x,y), computes the corresponding location in the input image using $(v,w) = T^{-1}(x,y)$

–It then **interpolates** (using one of the techniques discussed previously among the nearest input pixels to determine the intensity of the output pixel value.

–Inverse mappings are more efficient to implement than forward mappings and are used in numerous commercial implementations of spatial transformations

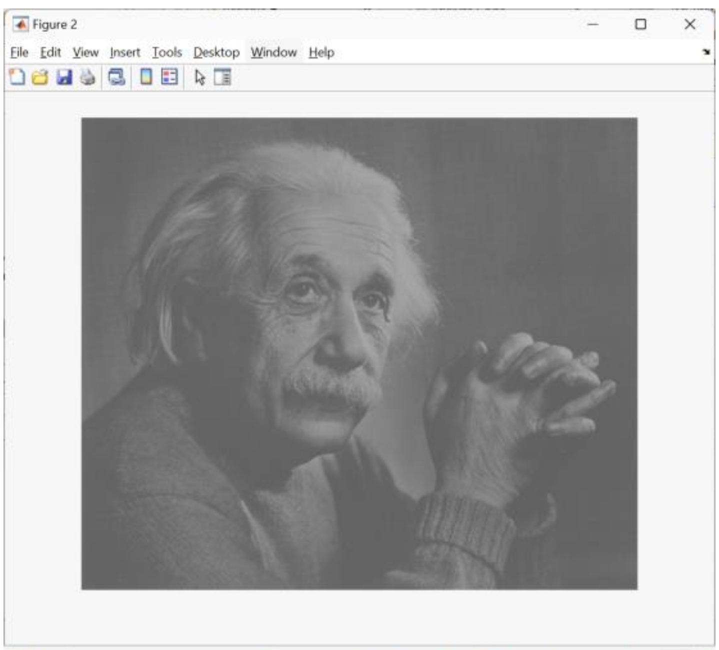


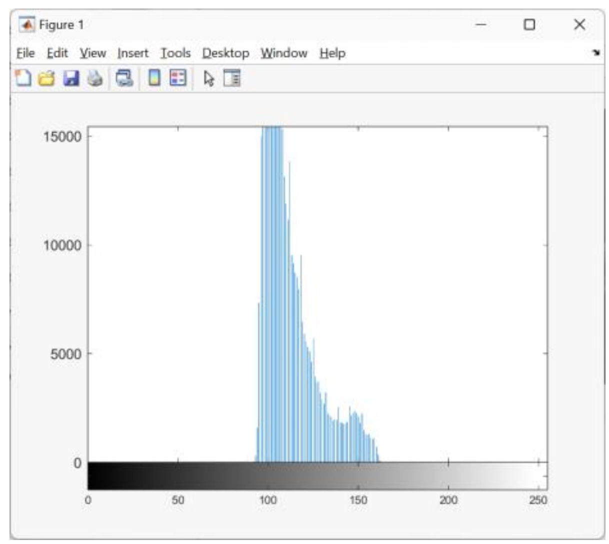


Probabilistic Methods

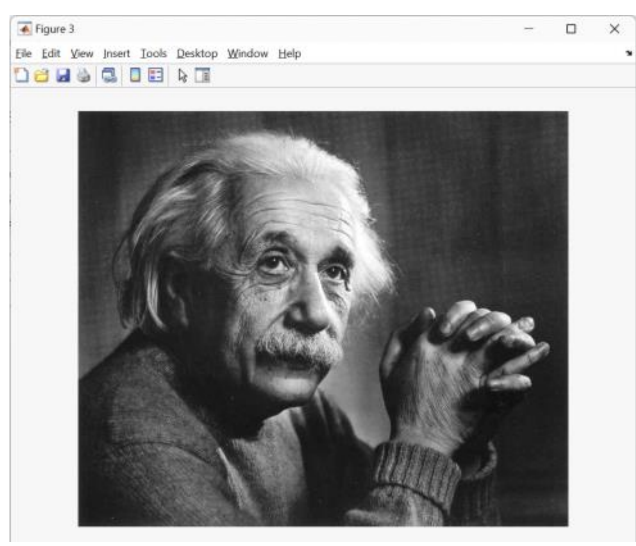
http://www.ece.uah.edu/~dwpan/course/ee604/code/ch2/fig2_41.m

```
% STD as a measure of intensity contrast
I = imread('Fig0241(a)(einstein low contrast).tif');
I_1d = reshape(I, 1, 679*800);
imhist(I);
std(double(I_1d))
I = imread('Fig0241(a)(einstein med contrast).tif');
I = imread('Fig0241(a)(einstein hig contrast).tif');
```






```
>> std(double(I_1d))
ans =
  14.2924
```




```
>> I = imread('Fig0241(c)(einstein high contrast).tif');
>> I_1d = reshape(I, 1, 679*800);
imhist(I);
figure; imshow(I)
std(double(I_1d))
ans =
  49.2428
```