Geometric Transformation

Dr. Mongkol Ekpanyapong

Geometric operations

- These are the operations to rotate, flip, crop or resize images

- OpenCV provides some support functions

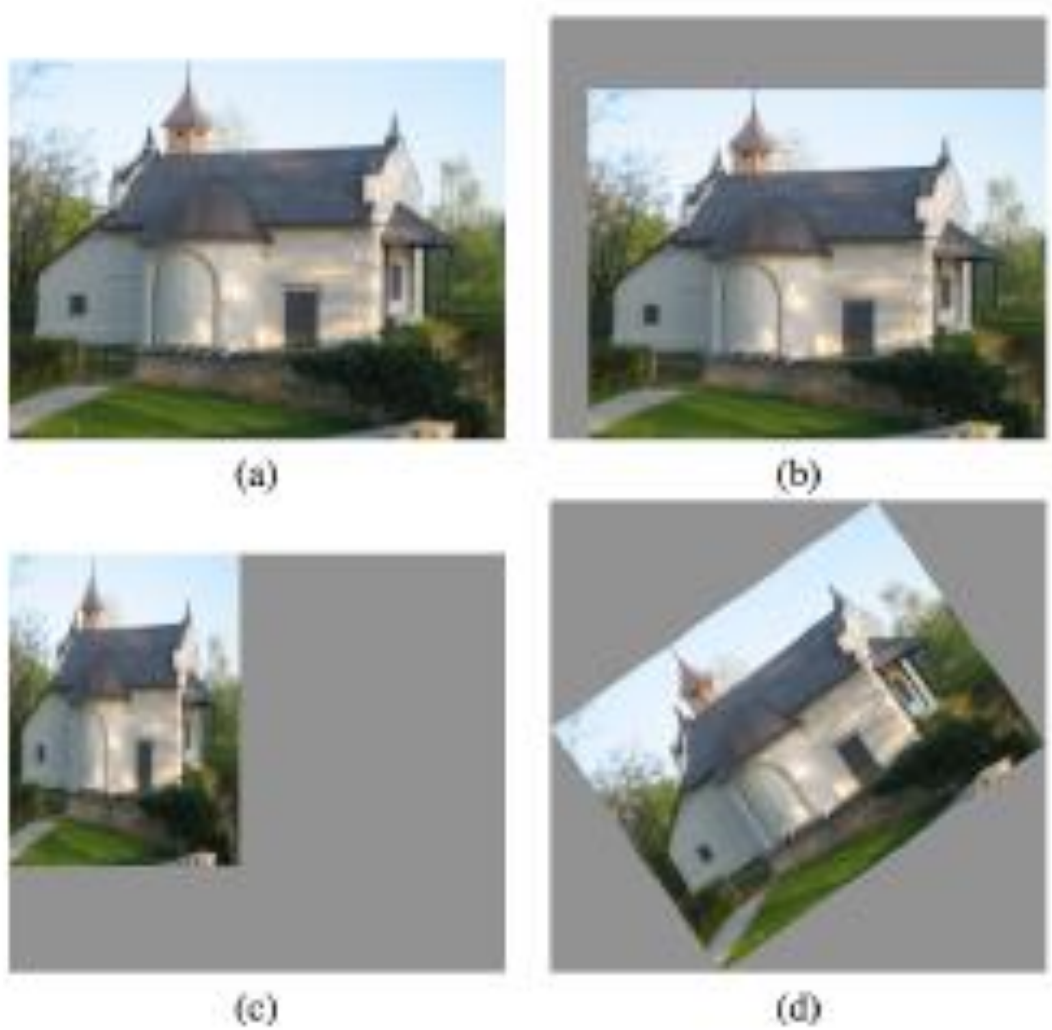
Example of the operations

(a) original

(b) translation

(c) resize

(d) rotation



Geometric operations

Two required basic components:

- Mapping function specify a set of spatial transformation equations

- Interpolation methods used to compute the new value of each pixel in the spatially transformed image

Mapping and Affine Transformations

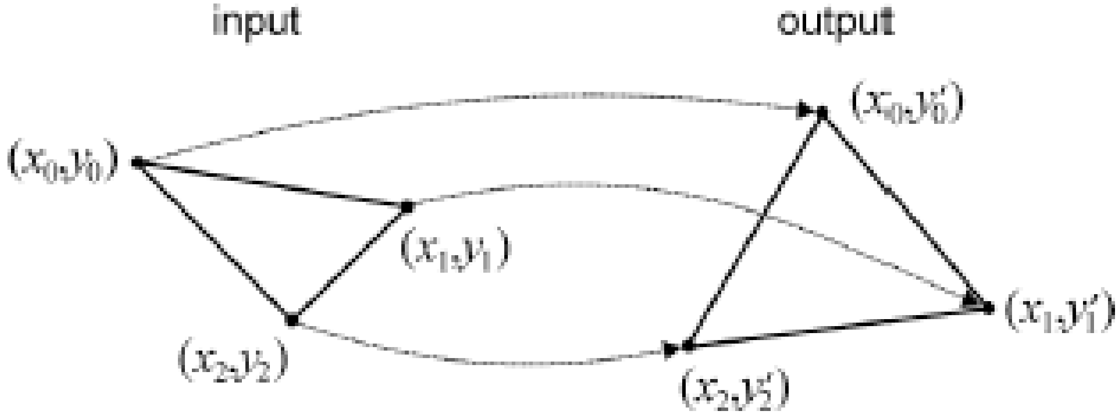
- Given an input image f(x,y), a geometric operation is the process to transform into a new image g(x',y')

$$f(x,y) \rightarrow g(x',y')$$

- To model this, a mapping function is needed

$$(x',y') \rightarrow T(x,y)$$

Example



$$f(x, y) \rightarrow g(x', y')$$

$$(x', y') = T(x, y)$$

The mapping function

- The mapping function is an arbitrary 2D function. It is often specified as 2 separate functions.

$$x' = T_x(x,y)$$

$$y' = T_y(x,y)$$

- In the case that T_x and T_y are linear combinations of x and y, it is called affine transformation

Affine transformation

- The affine transformation can be expressed as:

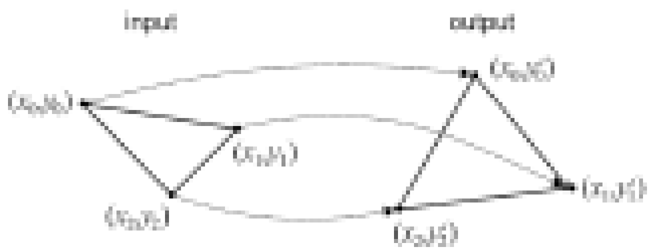
$$x' = a_0x + a_1y + a_2$$

$$y' = b_0x + b_1y + b_2$$

Homogeneous Transformation Matrix:

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} a_0 & a_1 & a_2 \\ b_0 & b_1 & b_2 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

Example



$$f(x, y) \rightarrow g(x', y')$$

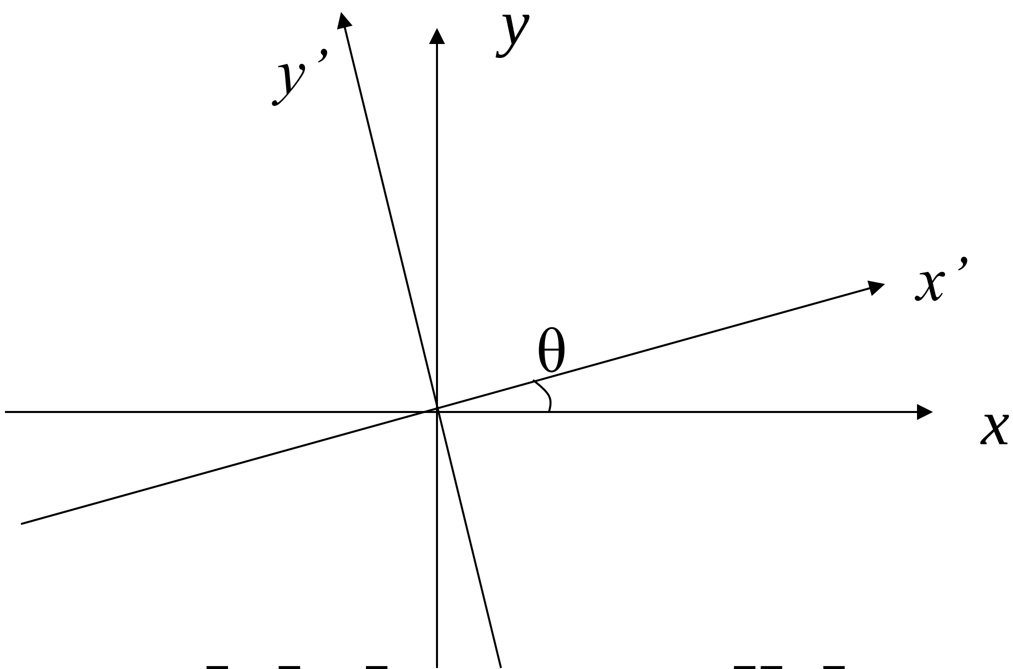
$$(x', y') = T(x, y)$$

$$x' = T_x(x, y) \rightarrow x' = a_0 x + a_1 y + a_2,$$

$$y' = T_y(x, y). \rightarrow y' = b_0 x + b_1 y + b_2.$$

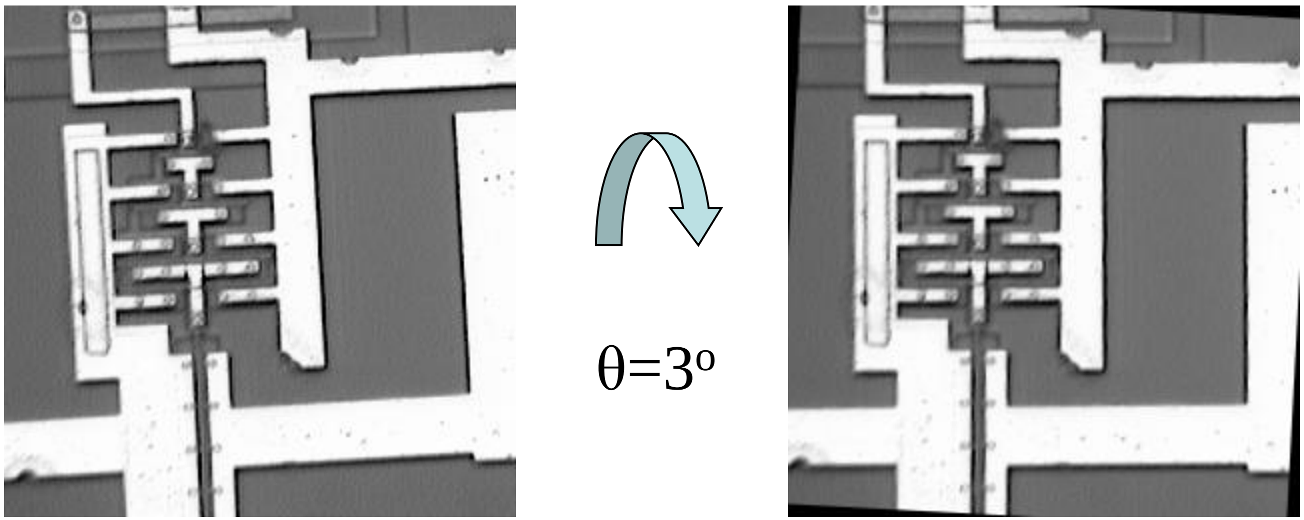
$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} a_0 & a_1 & a_2 \\ b_0 & b_1 & b_2 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

Rotation



$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$$

Rotation Example

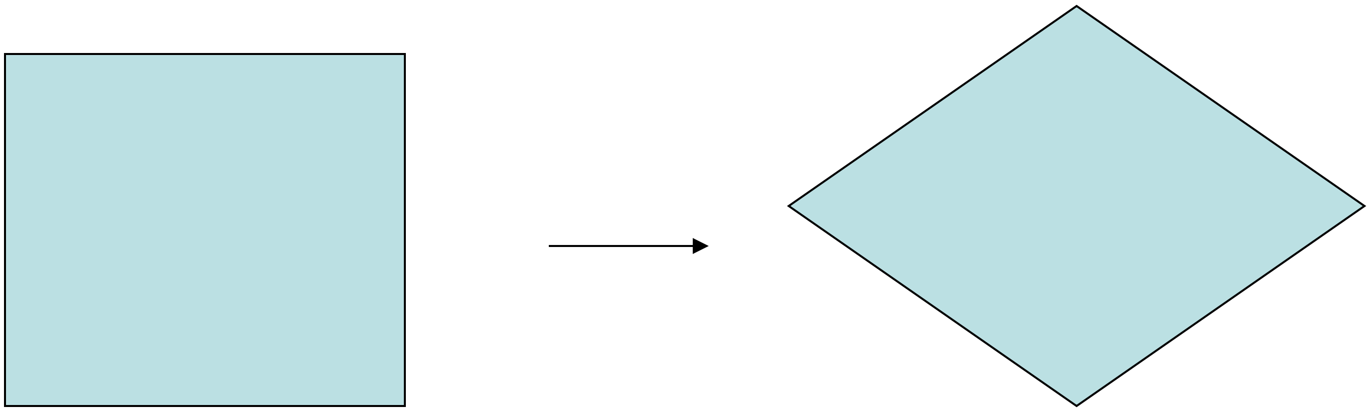


Scale

a=2 $\longrightarrow$

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} a & 0 \\ 0 & 1/a \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$$

Affine Transform



square

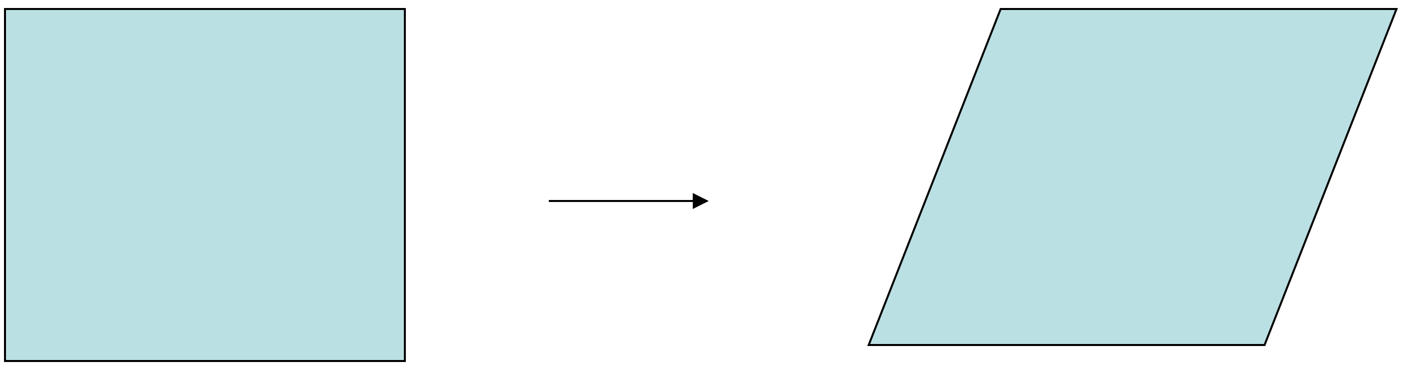
parallelogram

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} d_x \\ d_y \end{bmatrix}$$

Affine Transform Example

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} .5 & 1 \\ .5 & -2 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} 0 \\ 1 \end{bmatrix}$$

Shear



square

parallelogram

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ s & 1 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} d_x \\ d_y \end{bmatrix}$$

Shear Example

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} 1 & 0.5 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} 0 \\ 1 \end{bmatrix}$$

Affine Transformations

- These two equations are often augmented by a third equation that may initially seem frivolous but allows for convenient representation and efficient computation.
  - The validity of the equation 1=0x+0y+1 is obvious but including it as a third constraint within our system allows us to write the affine system in the matrix form of Equation (7.3).
- The 3x3 matrix of coefficients in Equation (7.3) is known as the homogeneous transformation matrix.
  - Using this augmented system, a two-dimensional point is represented as a 3x1 column vector where the first two elements correspond to the column and row while the third coordinate is constant at 1.
  - When a two-dimensional point is represented in this form it is referred to as a homogenous coordinate.
  - When using homogenous coordinates, the transformation of a point V with a transformation matrix A is given by the matrix product AV and is itself a point. In other words, there is closure under transformation.

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} m_{00} & m_{01} & m_{02} \\ m_{10} & m_{11} & m_{12} \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \qquad (7.3)$$

Affine Transformations

- An affine transformation matrix is a six parameter entity controlling the coordinate mapping between source and destination image.
- The following table shows the correspondence between coefficient settings and effect.

	m_{00}	m_{10}	m_{01}	m_{11}	m_{02}	m_{12}
translation	1	0	0	1	δx	δy
rotation	$\cos(\theta)$	$-\sin(\theta)$	$\sin(\theta)$	$\cos(\theta)$	0	0
shear	1	sh_y	sh_x	1	0	0
scale	s_x	0	0	s_y	0	0
horizontal reflection	-1	0	0	1	x_c	0
vertical reflection	1	0	0	-1	0	y_c

Table 7.1. Coefficient settings for affine operations.

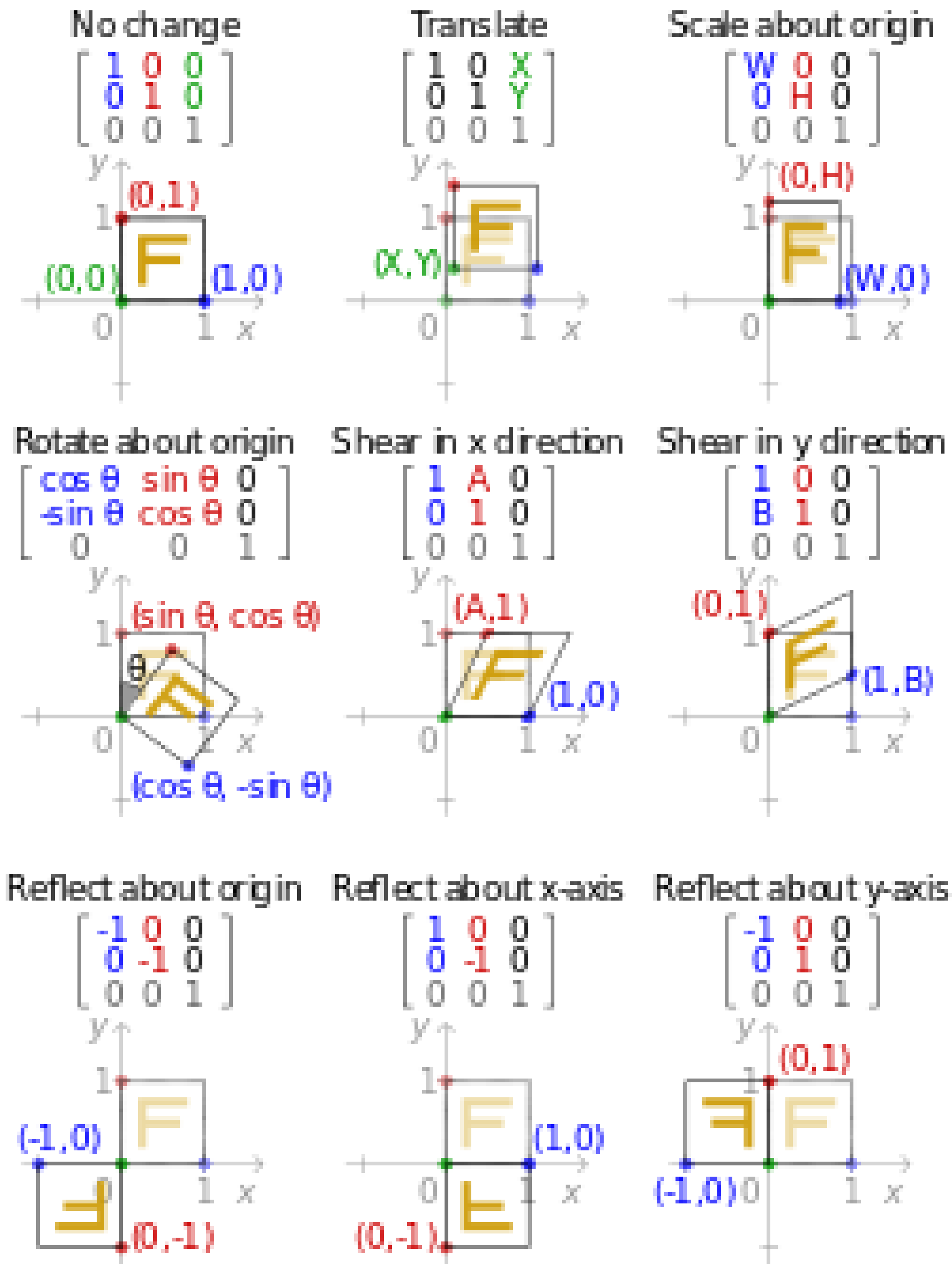
Rotation

- Clockwise

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} \cos\theta & \sin\theta \\ -\sin\theta & \cos\theta \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$$

- Counter clockwise

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$$


No change
1 0 0
0 1 0
0 0 1
(0,1)
(0,0)
(1,0)
Translate
1 0 X
0 1 Y
0 0 1
(X,Y)
Scale about origin
W 0 0
0 H 0
0 0 1
(0,H)
(W,0)
Rotate about origin
cos θ sin θ 0
-sin θ cos θ 0
0 0 1
(sin θ, cos θ)
(cos θ, -sin θ)
θ
Shear in x direction
1 A 0
0 1 0
0 0 1
(A,1)
(1,0)
Shear in y direction
1 0 0
B 1 0
0 0 1
(0,1)
(1,B)
Reflect about origin
-1 0 0
0 -1 0
0 0 1
(-1,0)
(0,-1)
Reflect about x-axis
1 0 0
0 -1 0
0 0 1
(1,0)
(0,-1)
Reflect about y-axis
-1 0 0
0 1 0
0 0 1
(0,1)
(-1,0)

Example of transformation

Translation by [25,15] pixel

$$\begin{bmatrix} 1 & 0 & 25 \\ 0 & 1 & 15 \\ 0 & 0 & 1 \end{bmatrix}$$

Scaling by a factor 3.5

$$\begin{bmatrix} 3.5 & 0 & 0 \\ 0 & 3.5 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

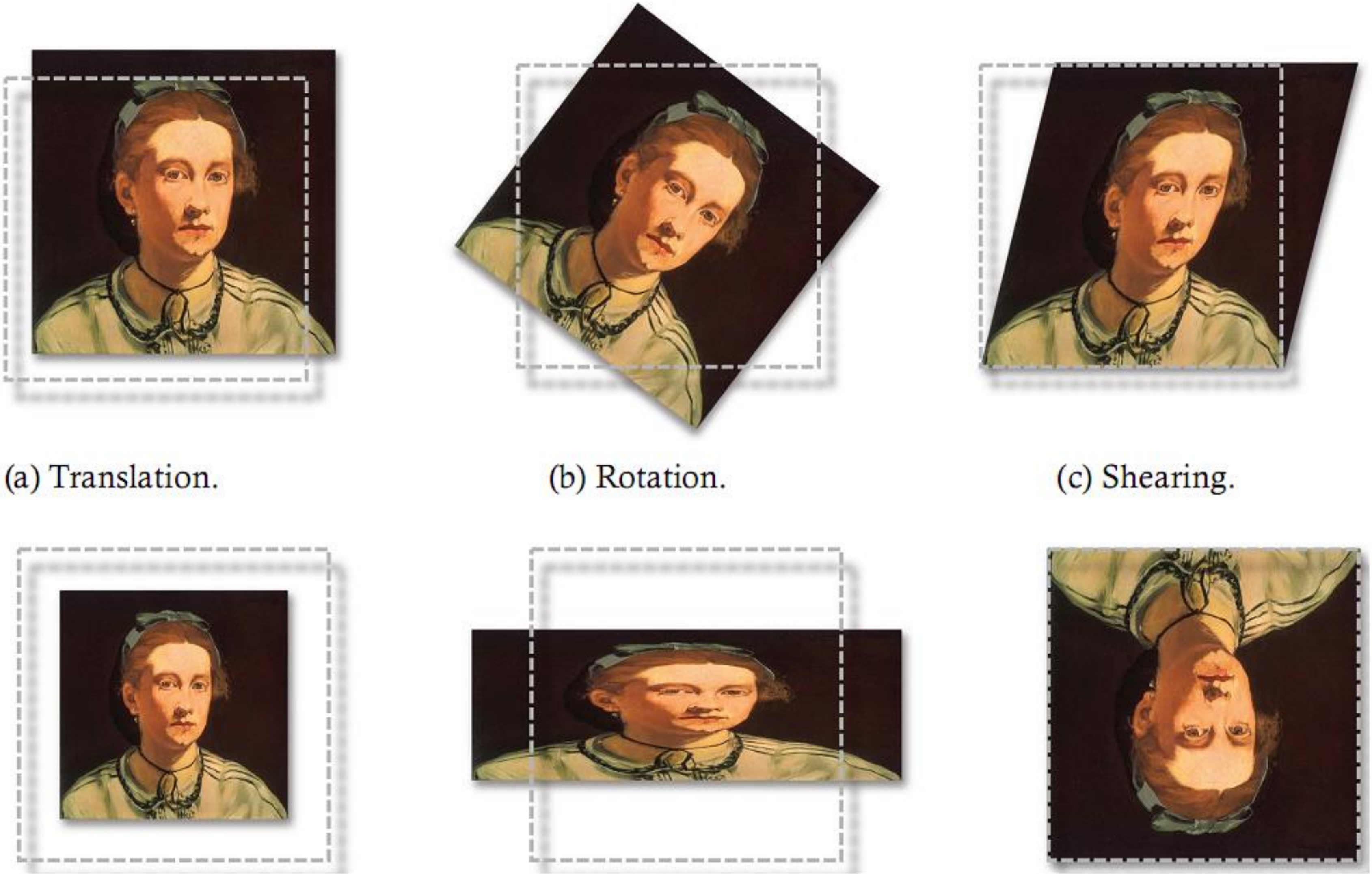
Example of transformation

Rotation by 30 degree (counter clockwise)

$$\begin{bmatrix} 0.866 & -0.5 & 0 \\ 0.5 & 0.866 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Shear by a factor [2,3]

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



(a) Translation. (b) Rotation. (c) Shearing.

(d) Uniform scaling. (e) Nonuniform scaling. (f) Reflection.

Figure 7.1. Linear geometric transformations.

Affine Transformations

- A single homogeneous matrix can also represent a *sequence* of individual affine operations.
- Let A and B represent affine transformation matrices
  - the affine matrix corresponding to the application of A followed by B is given as BA
  - BA is itself a homogeneous transformation matrix.
  - Matrix multiplication, also termed concatenation, therefore corresponds to the sequential composition of individual affine transformations.
  - Note that the order of multiplication is both important and opposite to the way the operations are mentally envisioned.
- While we speak of transform A followed by transform B, these operations are actually composed as matrix B multiplied by (or concatenated with) matrix A.
- Assume, for example, that matrix A represents a rotation of 30 degrees CCW about the origin and matrix B represents a horizontal shear by a factor of .5. The affine matrix corresponding to the rotation followed by shear is given as BA.

$$\begin{bmatrix} 1 & 0.5 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} .866 & -0.5 & 0 \\ 0.5 & .866 & 0 \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 1.116 & -0.067 & 0 \\ 0.5 & .866 & 0 \\ 0 & 0 & 1 \end{bmatrix}. \qquad (7.4)$$

Affine Transformations

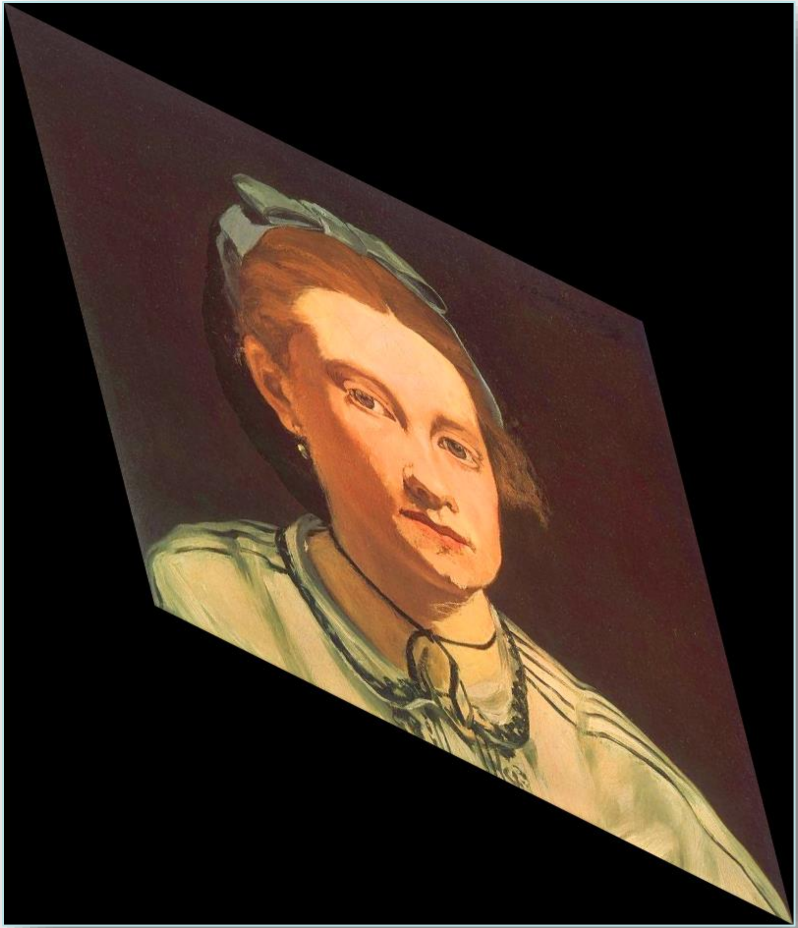
Explain what the following transformation matrices accomplishes

1.0	0.25	0.0
0.0	1.0	0.0
0.0	0.0	1.0

Affine Transformations

Explain what the following transformation matrices accomplishes

1.0	0.25	0.0
0.5	1.0	0.0
0.0	0.0	1.0



Affine Transformations

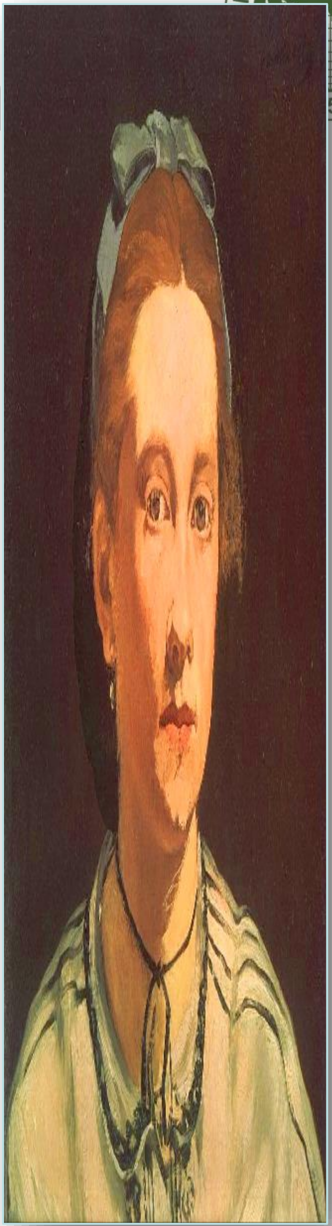
Explain what the following transformation matrices accomplishes

.87	0.5	0.0
-0.5	.87	0.0
0.0	0.0	1.0

Affine Transformation

Explain what the following transformation matrix accomplishes

0.5	0.0	0.0
0.0	2.0	0.0
0.0	0.0	1.0



Scaling

```
import cv2
import numpy as np
from matplotlib import pyplot as plt
img = cv2.imread('images/cameraman2.tif')
scale = cv2.resize(img,None,fx=2, fy=2, interpolation = cv2.INTER_CUBIC)

cv2.imshow("original",img)
cv2.imshow("scale",scale)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

Translation

```
import cv2
import numpy as np
from matplotlib import pyplot as plt
img = cv2.imread('images/cameraman2.tif')
rows,cols,depth = img.shape
M = np.float32([[1,0,50],[0,1,50]])
translation = cv2.warpAffine(img,M,(cols,rows))

cv2.imshow("original",img)
cv2.imshow("translation",translation)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

Rotation

```
import cv2
import numpy as np
from matplotlib import pyplot as plt
img = cv2.imread('images/cameraman2.tif')
rows,cols,depth = img.shape

M =  cv2.getRotationMatrix2D((cols/2,rows/2),90,1)
rotation = cv2.warpAffine(img,M,(cols,rows))

cv2.imshow("original",img)
cv2.imshow("rotation",rotation)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

Shear

```
import cv2
import numpy as np
import math
from matplotlib import pyplot as plt
img = cv2.imread('images/cameraman2.tif')
rows,cols,depth = img.shape
m = 1/math.tan(3.1416/3)
M = np.float32([[1,m,0],[0,1,0]])
rotation = cv2.warpAffine(img,M,(cols+(int)(cols/2),rows))

cv2.imshow("original",img)
cv2.imshow("rotation",rotation)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

Reflection

```
import cv2
import numpy as np
import math
from matplotlib import pyplot as plt
img = cv2.imread('images/cameraman2.tif')
rows,cols,depth = img.shape

M = np.float32([[-1,0,cols],[0,1,0]])
reflex = cv2.warpAffine(img,M,(cols,rows))

cv2.imshow("original",img)
cv2.imshow("reflex",reflex)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

```
import cv2
import numpy as np
from matplotlib import pyplot as plt

img = cv2.imread('images/cameraman2.tif')
rows,cols,ch = img.shape
pts1 = np.float32([[50,50],[200,50],[50,200]])
pts2 = np.float32([[10,100],[200,50],[100,250]])
M = cv2.getAffineTransform(pts1,pts2)
dst = cv2.warpAffine(img,M,(cols,rows))
cv2.circle(img,(50, 50),3,(0,255,0),-1);
cv2.circle(img,(200, 50),3,(0,255,0),-1);
cv2.circle(img,(50, 200),3,(0,255,0),-1);
plt.subplot(121),plt.imshow(img),plt.title('Input')
cv2.circle(dst,(10, 100),3,(0,255,0),-1);
cv2.circle(dst,(200, 50),3,(0,255,0),-1);
cv2.circle(dst,(100, 250),3,(0,255,0),-1);
plt.subplot(122),plt.imshow(dst),plt.title('Output')
plt.show()
    return 0;
}
```

Question?

- Can you perform clockwise rotation by 45 degree, shearing in x direction by a factor of 2 and scaling by the factor of 1.5 in y direction on camera man image

Questions?