LINEAR ALGEBRA ASSIGNMENT -UNIT 3

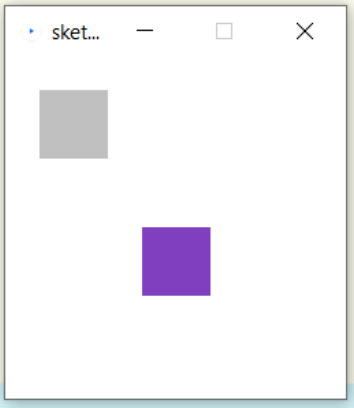
UE20MA251

GAURAV MAHAJAN

SEC:C SRN:PES1UG20CS150

Translation: Moving the Grid

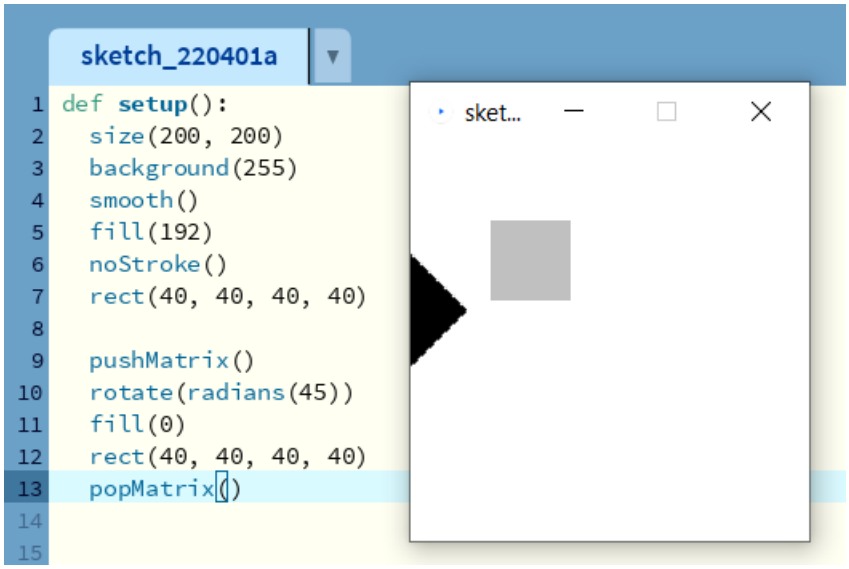
```
def setup():
    size(200, 200)
    background(255)
    noStroke()

    # draw the original position in gray
    fill(192)
    rect(20, 20, 40, 40)

    # draw a translucent red rectangle by changing the coordinates
    fill(255, 0, 0, 128)
    rect(20 + 60, 20 + 80, 40, 40)

    # draw a translucent blue rectangle by translating the grid
    fill(0, 0, 255, 128)
    pushMatrix()
    translate(60, 80)
    rect(20, 20, 40, 40)
    popMatrix()
```



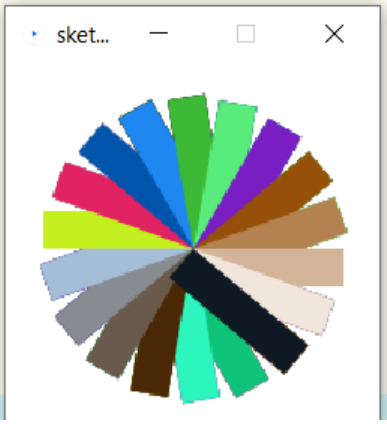
sketch_220401a

```
def setup():
  size(400, 100)
  background(255)
  for i in xrange(10,350,50):
    house(i, 20)

def house(x, y):
  triangle(x + 15, y, x, y + 15, x + 30, y + 15)
  rect(x, y + 15, 30, 30)
  rect(x + 12, y + 30, 10, 15)

def house(x, y):
  pushMatrix()
  translate(x, y)
  triangle(15, 0, 0, 15, 30, 15)
  rect(0, 15, 30, 30)
  rect(12, 30, 10, 15)
  popMatrix()
```

Rotation

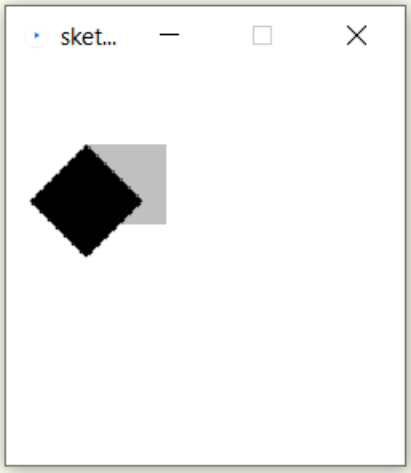

sketch_220401a
1 def setup():
2 size(200, 200)
3 background(255)
4 smooth()
5 fill(192)
6 noStroke()
7 rect(40, 40, 40, 40)
8
9 pushMatrix()
10 rotate(radians(45))
11 fill(0)
12 rect(40, 40, 40, 40)
13 popMatrix()
14
15
sket...

```
def setup():
    size(200, 200)
    background(255)
    smooth()
    noStroke()

def draw():
    if (frameCount % 10 == 0):
        fill(frameCount * 3 % 255, frameCount * 5 % 255,
          frameCount * 7 % 255)
        pushMatrix()
        translate(100, 100)
        rotate(radians(frameCount * 2  % 360))
        rect(0, 0, 80, 20)
        popMatrix()
```



sketch_220401a

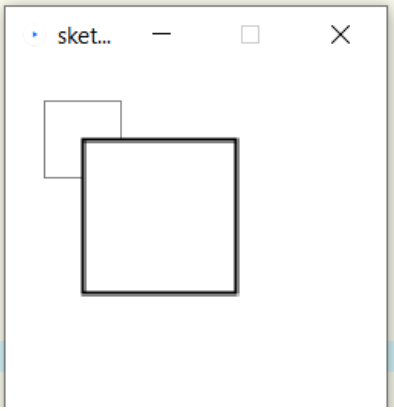
```
def setup():
    size(200, 200)
    background(255)
    smooth()
    fill(192)
    noStroke()
    rect(40, 40, 40, 40)

    pushMatrix()
    # move the origin to the pivot point
    translate(40, 40)

    # then pivot the grid
    rotate(radians(45))

    # and draw the square at the origin
    fill(0)
    rect(0, 0, 40, 40)
    popMatrix()
```



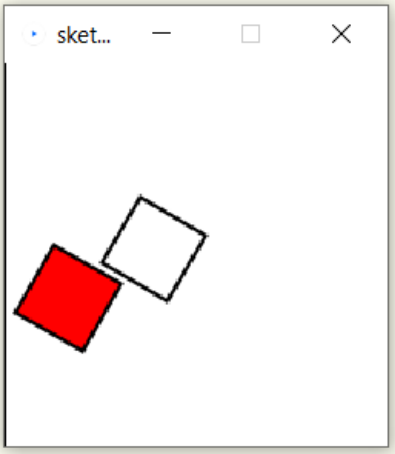
Scaling

sketch_220401a

```
def setup():
    size(200,200)
    background(255)

    stroke(128)
    rect(20, 20, 40, 40)

    stroke(0)
    pushMatrix()
    scale(2.0)
    rect(20, 20, 40, 40)
    popMatrix()
```



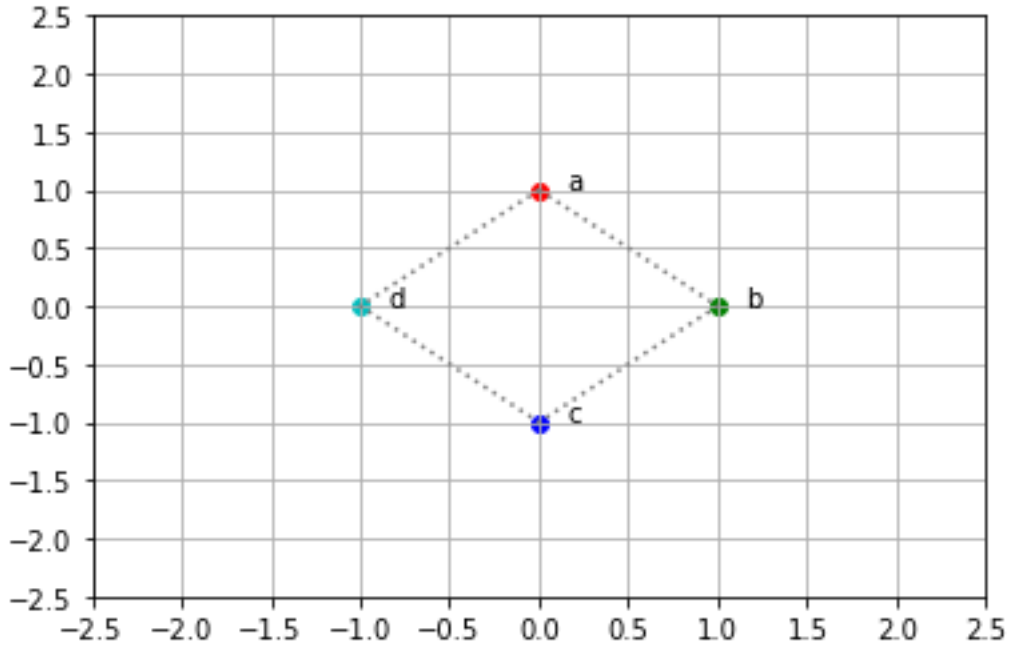
```
def setup():
    size(200, 200)
    background(255)
    smooth()
    line(0, 0, 200, 0)  # draw axes
    line(0, 0, 0, 200)

    pushMatrix()
    fill(255, 0, 0)     # red square
    rotate(radians(30))
    translate(70, 70)
    scale(2.0)
    rect(0, 0, 20, 20)
    popMatrix()

    pushMatrix()
    fill(255)           # white square
    translate(70, 70)
    rotate(radians(30))
    scale(2.0)
    rect(0, 0, 20, 20)
    popMatrix()
```



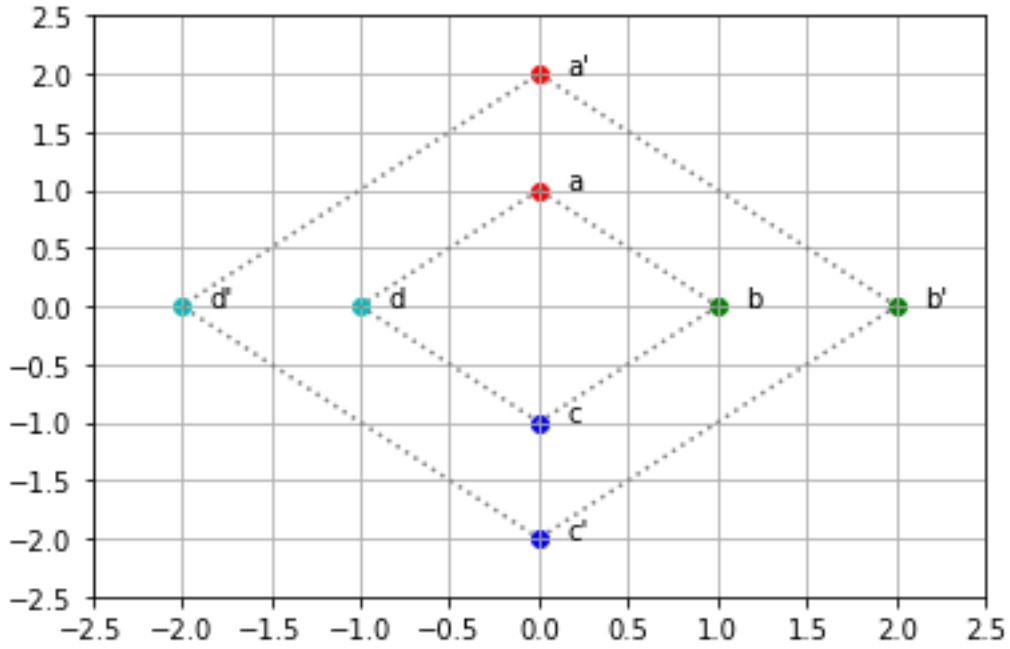
Affine Image Transformations in Python with Numpy, Pillow and OpenCV

[1]
```
import matplotlib.pyplot as plt
import numpy as np
import string

# points a, b and, c
a, b, c, d = (0, 1, 0), (1, 0, 1), (0, -1, 2), (-1, 0, 3)

# matrix with row vectors of points
A = np.array([a, b, c, d])

# 3x3 Identity transformation matrix
I = np.eye(3)
```

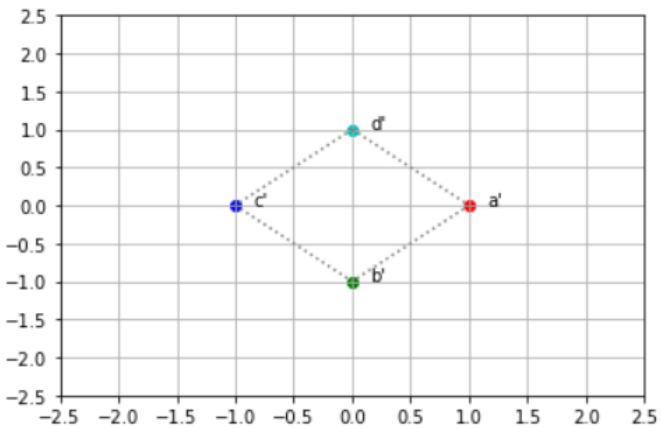
[3]
```
color_lut = 'rgbc'
fig = plt.figure()
ax = plt.gca()
xs = []
ys = []
for row in A:
    output_row = I @ row
    x, y, i = output_row
    xs.append(x)
    ys.append(y)
    i = int(i) # convert float to int for indexing
    c = color_lut[i]
    plt.scatter(x, y, color=c)
    plt.text(x + 0.15, y, f"{string.ascii_letters[i]}")
xs.append(xs[0])
ys.append(ys[0])
plt.plot(xs, ys, color="gray", linestyle='dotted')
ax.set_xticks(np.arange(-2.5, 3, 0.5))
ax.set_yticks(np.arange(-2.5, 3, 0.5))
plt.grid()
```



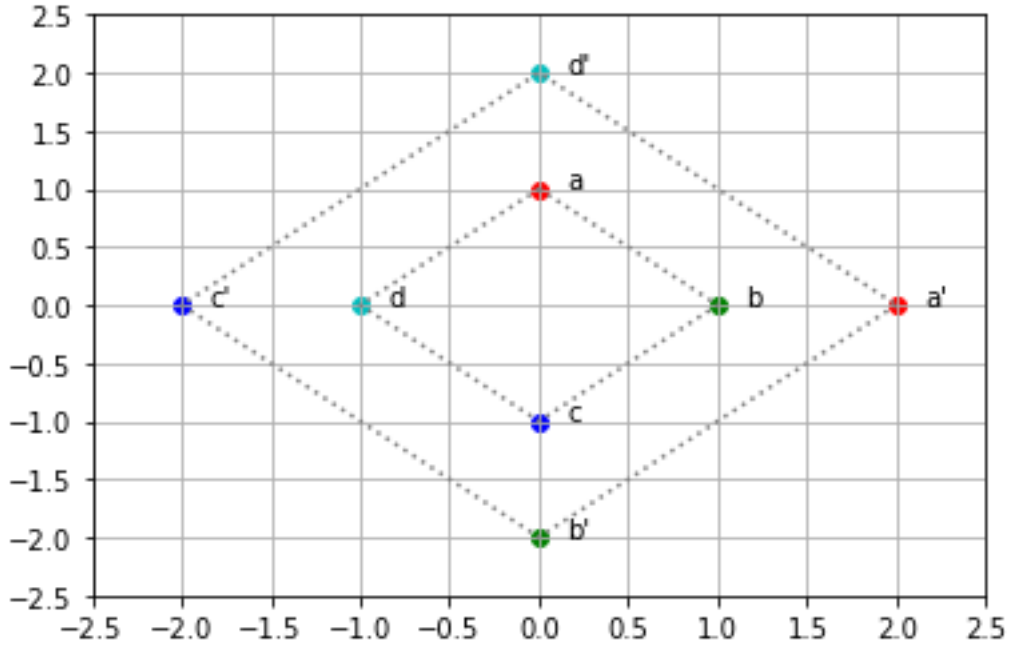

```
# create the scaling transformation matrix
T_s = np.array([[2, 0, 0], [0, 2, 0], [0, 0, 1]])

fig = plt.figure()
ax = plt.gca()
xs_s = []
ys_s = []
for row in A:
    output_row = T_s @ row
    x, y, i = row
    x_s, y_s, i_s = output_row
    xs_s.append(x_s)
    ys_s.append(y_s)
    i, i_s = int(i), int(i_s) # convert float to int for indexing
    c, c_s = color_lut[i], color_lut[i_s] # these are the same but, its good to be explicit
    plt.scatter(x, y, color=c)
    plt.scatter(x_s, y_s, color=c_s)
    plt.text(x + 0.15, y, f"{string.ascii_letters[int(i)]}")
    plt.text(x_s + 0.15, y_s, f"{string.ascii_letters[int(i_s)]}'")

xs_s.append(xs_s[0])
ys_s.append(ys_s[0])
plt.plot(xs, ys, color="gray", linestyle='dotted')
plt.plot(xs_s, ys_s, color="gray", linestyle='dotted')
ax.set_xticks(np.arange(-2.5, 3, 0.5))
ax.set_yticks(np.arange(-2.5, 3, 0.5))
plt.grid()
plt.show()
```




```
T_r = np.array([[0, 1, 0], [-1, 0, 0], [0, 0, 1]])

fig = plt.figure()
ax = plt.gca()
for row in A:
    output_row = T_r @ row
    x_r, y_r, i_r = output_row
    i_r = int(i_r) # convert float to int for indexing
    c_r = color_lut[i_r] # these are the same but, its good to be explicit
    letter_r = string.ascii_letters[i_r]
    plt.scatter(x_r, y_r, color=c_r)
    plt.text(x_r + 0.15, y_r, f"{letter_r}'")

plt.plot(xs, ys, color="gray", linestyle='dotted')
ax.set_xticks(np.arange(-2.5, 3, 0.5))
ax.set_yticks(np.arange(-2.5, 3, 0.5))
plt.grid()
plt.show()
```



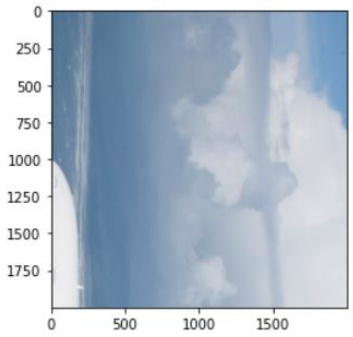
```
T = T_s @ T_r

fig = plt.figure()
ax = plt.gca()

xs_comb = []
ys_comb = []
for row in A:
    output_row = T @ row
    x, y, i = row
    x_comb, y_comb, i_comb = output_row
    xs_comb.append(x_comb)
    ys_comb.append(y_comb)
    i, i_comb = int(i), int(i_comb) # convert float to int for indexing
    c, c_comb = color_lut[i], color_lut[i_comb] # these are the same but, its good to be explicit
    letter, letter_comb = string.ascii_letters[i], string.ascii_letters[i_comb]
    plt.scatter(x, y, color=c)
    plt.scatter(x_comb, y_comb, color=c_comb)
    plt.text(x + 0.15 , y, f"{letter}")
    plt.text(x_comb + 0.15, y_comb, f"{letter_comb}'")
xs_comb.append(xs_comb[0])
ys_comb.append(ys_comb[0])
plt.plot(xs, ys, color="gray", linestyle='dotted')
plt.plot(xs_comb, ys_comb, color="gray", linestyle='dotted')
ax.set_xticks(np.arange(-2.5, 3, 0.5))
ax.set_yticks(np.arange(-2.5, 3, 0.5))
plt.grid()
plt.show()
```



[15]
```
        [0, -1, 0],
        [1, 0, 0],
        [0, 0, 1]])
    # scale
    T_scale = np.array([
        [2, 0, 0],
        [0, 2, 0],
        [0, 0, 1]])
    # center original to 0,0
    T_neg500 = np.array([
        [1, 0, -500],
        [0, 1, -500],
        [0, 0, 1]])
    T = T_pos1000 @ T_rotate @ T_scale @ T_neg500
    T_inv = np.linalg.inv(T)
```

```
img_transformed = img.transform((2000, 2000), Image.AFFINE, data=T_inv.flatten()[:6], resample=Image.NEAREST)
plt.imshow(np.asarray(img_transformed))
```

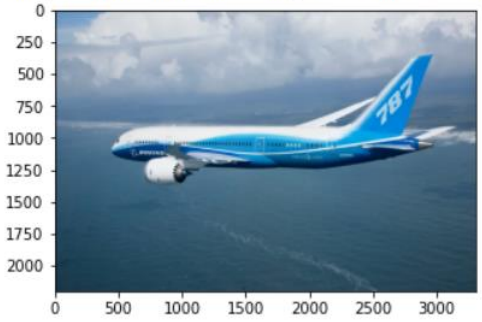
```
<matplotlib.image.AxesImage at 0x7fa9e1d73210>
```



Using pillow

```
from PIL import Image
img = Image.open('/K64998-41.jpg')
plt.figure(figsize=(5, 5))
plt.imshow(np.asarray(img))
```

```
<matplotlib.image.AxesImage at 0x7fa9e1e00b10>
```

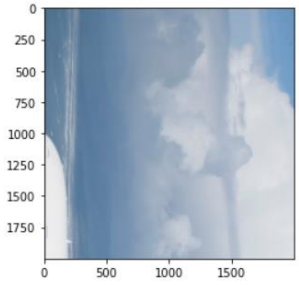



```
[15] # recenter resultant image
     T_pos1000 = np.array([
         [1, 0, 1000],
         [0, 1, 1000],
         [0, 0, 1]])
     # rotate - opposite angle
     T_rotate = np.array([
         [0, -1, 0],
         [1, 0, 0],
         [0, 0, 1]])
     # scale
     T_scale = np.array([
```

```
    [1, 0, 0],
    [0, 0, 1]])
# scale
T_scale = np.array([
    [2, 0, 0],
    [0, 2, 0],
    [0, 0, 1]])
# center original to 0,0
T_neg500 = np.array([
    [1, 0, -500],
    [0, 1, -500],
    [0, 0, 1]])
T = T_pos1000 @ T_rotate @ T_scale @ T_neg500
T_inv = np.linalg.inv(T)
```

```
img_transformed = img.transform((2000, 2000), Image.AFFINE, data=T_inv.flatten()[:6], resample=Image.NEAREST)
plt.imshow(np.asarray(img_transformed))
```

```
<matplotlib.image.AxesImage at 0x7f66cfb91310>
```



```
import cv2
img = cv2.imread('/content/K64998-41.jpg')
plt.imshow(cv2.cvtColor(img, cv2.COLOR_BGR2RGB))
```

```
<matplotlib.image.AxesImage at 0x7f66c60f7b10>
```


```
T_opencv = np.float32(T.flatten()[:6].reshape(2,3))
img_transformed = cv2.warpAffine(img, T_opencv, (2000, 2000))
plt.imshow(cv2.cvtColor(img_transformed, cv2.COLOR_BGR2RGB))
```

```
<matplotlib.image.AxesImage at 0x7f66c60783d0>
```