R08521609_handson1

March 4, 2021

1 1. Intro to Linear Algebra

1.1 1.1 Scalars, Vectors, Matrices and Tensors

Basic definitions:

- A scalar is a single number or a matrix with a single entry.
- A vector is a 1-d array of numbers. Another way to think of vectors is identifying points in space with each element giving the coordinate along a different axis.

$$x = \begin{bmatrix} x_1 \\ x_2 \\ \cdots \\ x_n \end{bmatrix}$$

- A matrix is a 2-D array where each element is identified by two indices (ROW then COLUMN).

$$A = \begin{bmatrix} A_{1,1} & A_{1,2} & \cdots & A_{1,n} \\ A_{2,1} & A_{2,2} & \cdots & A_{2,n} \\ \cdots & \cdots & \cdots & \cdots \\ A_{m,1} & A_{m,2} & \cdots & A_{m,n} \end{bmatrix}$$

- A tensor is a n-dimensional array with $n > 2$

$$B = \begin{bmatrix} \begin{bmatrix} A_{0,1,1} & A_{0,1,2} & \cdots & A_{0,1,n} \\ A_{0,2,1} & A_{0,2,2} & \cdots & A_{0,2,n} \\ \cdots & \cdots & \cdots & \cdots \\ A_{0,m,1} & A_{0,m,2} & \cdots & A_{0,m,n} \end{bmatrix} ... \begin{bmatrix} A_{k,1,1} & A_{k,1,2} & \cdots & A_{k,1,n} \\ A_{k,2,1} & A_{k,2,2} & \cdots & A_{k,2,n} \\ \cdots & \cdots & \cdots & \cdots \\ A_{k,m,1} & A_{k,m,2} & \cdots & A_{k,m,n} \end{bmatrix} \end{bmatrix}$$

- In the following code section, we will use the Python Image Library (PIL) or Pillow, and NumPy to manipulate an image in the form of an array.

[ ]:
```
from google.colab import drive
drive.mount("/content/drive")

!wget -nc https://raw.githubusercontent.com/brpy/colab-pdf/master/colab_pdf.py
from colab_pdf import colab_pdf
colab_pdf('0304/R08521609_handson1.ipynb')
```

```
Drive already mounted at /content/drive; to attempt to forcibly remount, call
drive.mount("/content/drive", force_remount=True).
File colab_pdf.py already there; not retrieving.


WARNING: apt does not have a stable CLI interface. Use with caution in scripts.


WARNING: apt does not have a stable CLI interface. Use with caution in scripts.

Extracting templates from packages: 100%
```

[ ]:
```
from PIL import Image
import numpy as np
from numpy import asarray
# Open the image from working directory
# Upload the image using the files tab on the left hand side
image = Image.open('ntuce.jpg')
# summarize some details about the image
# print the image format, size and mode. For example, print(image.format)
print(image.format)
print(image.size)
print(image.mode)

# show the image
display(image)
# convert the image to a numpy array using asarray
data = np.asarray(image)
# print the type and shape of data, and the data
print(type(data))
print(data.shape)
print(data)
```

```
JPEG
(500, 281)
RGB
```

```
<class 'numpy.ndarray'>
(281, 500, 3)
[[[ 93 136 187]
  [ 91 137 187]
  [ 91 137 189]
  ...
  [ 85 133 182]
  [ 85 133 181]
  [ 84 132 181]]

 [[ 93 136 187]
  [ 93 136 187]
  [ 94 137 188]
  ...
  [ 86 132 182]
  [ 85 133 181]
  [ 85 133 181]]

 [[ 91 137 186]
  [ 91 137 186]
  [ 91 137 186]
  ...
  [ 86 132 182]
  [ 85 133 181]
  [ 85 133 181]]

 ...
```

```
[[ 76  92  91]
  [ 76  92  92]
  [ 77  91  92]
  ...
  [ 74  84  85]
  [ 74  86  86]
  [ 73  85  85]]

 [[ 63  76  82]
  [ 62  75  81]
  [ 62  73  79]
  ...
  [ 73  83  84]
  [ 69  83  83]
  [ 70  82  82]]

 [[ 47  62  69]
  [ 46  59  68]
  [ 46  59  67]
  ...
  [ 71  83  83]
  [ 71  82  84]
  [ 71  82  84]]]
```

2 1.2 Addition

Matrices can be added if they have the same shape:

$$A + B = C$$

Each cell of A is added to the corresponding cell of B:

$$A_{i,j} + B_{i,j} = C_{i,j}$$

i is the row index and j the column index.

$$\begin{bmatrix} A_{1,1} & A_{1,2} \\ A_{2,1} & A_{2,2} \\ A_{3,1} & A_{3,2} \end{bmatrix} + \begin{bmatrix} B_{1,1} & B_{1,2} \\ B_{2,1} & B_{2,2} \\ B_{3,1} & B_{3,2} \end{bmatrix} = \begin{bmatrix} A_{1,1} + B_{1,1} & A_{1,2} + B_{1,2} \\ A_{2,1} + B_{2,1} & A_{2,2} + B_{2,2} \\ A_{3,1} + B_{3,1} & A_{3,2} + B_{3,2} \end{bmatrix}$$

The shape of A, B and C are equal

[ ]:
```
# convert the image into grayscale using 0.2989*R + 0.5870*G + 0.1140*B
data_gray = 0.2989*data[:,:,0] + 0.5870*data[:,:,1]+ 0.1140*data[:,:,2]
# convert to int
data_gray = np.uint8(data_gray)
# print data_gray shape and its value
```

```
print(data_gray.shape)
print(data_gray)

# convert to Pillow image with single channel
image2 = Image.fromarray(data_gray, 'L')
display(image2)
```

```
(281, 500)
[[128 128 129 ... 124 124 123]
 [128 128 129 ... 123 124 124]
 [128 128 128 ... 123 124 124]
 ...
 [ 87  87  86 ...  81  82  81]
 [ 72  71  70 ...  80  78  78]
 [ 58  56  56 ...  79  78  78]]
```

3 1.3 Dot product

$$C_{i,j} = A_{i,k}B_{k,j} = \sum_k A_{i,k}B_{k,j}$$

You can find more examples about the dot product here.

[ ]:
```
# convert the image into grayscale using 0.2989*R + 0.5870*G + 0.1140*B with␣
↪dot product np.dot or @
data_gray = np.dot(data,[0.2989, 0.5870, 0.1140])
data_gray = np.uint8(data_gray)
```

```python
# print data_gray shape
print(data_gray.shape)

# convert to Pillow image with single channel
image2 = Image.fromarray(data_gray, 'L')
display(image2)
# print pixel intensity value of (300,200)
print("Pixel Intesity:")
print(data_gray[200,300])
print(image2.getpixel((300, 200)))
# print pixel rgb value of (200,300) in two different ways, pillow image and numpy array
print("Pixel RGB:")
print(data[200,300])
print(image.getpixel((300, 200)))

# note that the dimension of pillow image and numpy array are different
```

```
(281, 500)
```

```
Pixel Intesity:
48
48
Pixel RGB:
[43 56 28]
(43, 56, 28)
```

4 1.4 (BONUS)Translation, scaling and rotation

Translation using Matrices

$$P' = P + t = (x + t_x, y + t_y) P' \rightarrow \begin{bmatrix} x + t_x \\ y + t_y \end{bmatrix} = \begin{bmatrix} 1 & 0 & t_x \\ 0 & 1 & t_y \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

Scaling Equation using Matrices

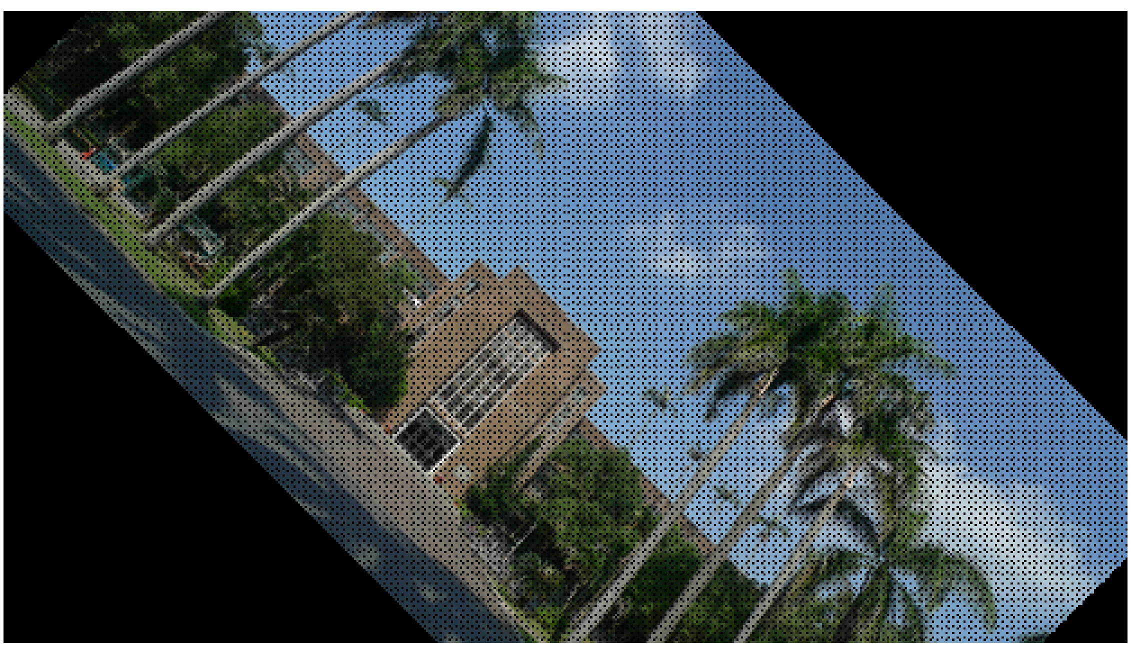
$$P' \rightarrow \begin{bmatrix} s_x x \\ s_y y \\ 1 \end{bmatrix} = \begin{bmatrix} s_x & 0 & 0 \\ 0 & s_y & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

Rotation using Matrices

$$P' \rightarrow \begin{bmatrix} x cos\theta - y sin\theta \\ x sin\theta + y cos\theta \\ 1 \end{bmatrix} = \begin{bmatrix} cos\theta & -sin\theta & 0 \\ sin\theta & cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

[ ]:
```python
# Rotate the image by 45 degree(clockwise)
# only calling the rotate function is prohibited, please change the value pixel␣
↪by pixel
from PIL import Image
import numpy as np
import math

originImg = Image.open('ntuce.jpg')
display(originImg)
image = np.array(originImg)


angle = math.radians(45)
cosine, sine = math.cos(angle), math.sin(angle)
height, width = image.shape[0], image.shape[1]

#create a new canvas
output = np.zeros((height,width,image.shape[2]))

# Choose the pixel as the origin ex. the middle of the picture
x0, y0 = int(width/2), int(height/2)
for j in range(height):
    for i in range(width):
```

```
        new_x = int((i-x0) * cosine - (j-y0) * sine)
        new_y = int((i-x0) * sine + (j-y0) * cosine)
        if ((x0 + new_x < width and x0 + new_x >= 0) and (y0 + new_y < height␣
 ↪and y0 + new_y >= 0)):
            output[y0 + new_y , x0 + new_x,:] = image[j, i,:]

pil_img=Image.fromarray((output).astype(np.uint8))
display(pil_img)
```



[ ]:

```

```