Module 2

Image Formation

▼ What is a vector?

a list of numbers in a single row or column

```
# row vector
-7   2   8

# column vector
-7
2
8
```

▼ What is a matrix?

an array of numbers in one or more rows and/or columns

```
# matrix
1 -8 7
26 4 3
```

▼ How can 2D points or the pixel coordinates of an image be denoted?

- using coordinates P = (x, y)
- it is sometimes useful to denote a 2D point using the augmented vector V = [x, y, 1]

```
# denote using column vector
x
y
```

▼ How can 2D lines be represented?

- 2D lines can be represented using *l* = (a, b, c).

- The corresponding line equation is ax + by + c = 0.

▼ How can 3D points be represented?

- point coordinates in three dimensions can be written using coordinates P = (x, y, z)
- a 3D point can also be denoted using the augmented vector V = [x, y, z, 1]

```
# column vector
x
y
z
```

▼ How can 3D planes be represented?

- 3D planes can be represented as m = (a, b, c, d)
- The corresponding plane equation is ax + by + cz + d = 0

▼ What are transformations?

- Transformations are used to edit or manipulate an image's geometry
- Examples: scaling, rotations, and translations

▼ What needs to be defined before a transformation?

A transformation must define where every point, before the transformation, ends up after the transformation

▼ What is scaling (transformation)?

- Scaling is a transformation in which the size and/or shape of the image is changed.
- We can scale in 3D by using the following matrix:

$$S = \begin{matrix} m_{00} & 0 & 0 \\ 0 & m_{11} & 0 \\ 0 & 0 & m_{22} \end{matrix}$$

- If you want to scale equally in all dimensions, then set the matrix values
  - m00= m11= m22
- Scaling equally in all dimensions does not change the shape of the image.

- If you do not scale equally in all dimensions, both the size and shape of the image will change.

▼ What is rotation (transformation)?

- A rotation is the motion of an image by holding one point fixed as the center and turning the figure by a certain angle in a particular direction.
- A rotation turns an image through a clockwise or counterclockwise angle about a fixed point known as the center of rotation (or origin).
- For an angle, 0, multiplying a vector in a plane by the matrix rotates the vector counterclockwise about the origin.

$$R = \begin{matrix} \cos(\theta) & -\sin(\theta) \\ \sin(\theta) & \cos(\theta) \end{matrix}$$

- 3D rotations of the x-, y-, and z-axes in a counterclockwise direction about the origin give the matices.

$$R_x = \begin{matrix} 1 & 0 & 0 \\ 0 & \cos(\theta) & \sin(\theta) \\ 0 & -\sin(\theta) & \cos(\theta) \end{matrix}$$

$$R_y = \begin{matrix} \cos(\theta) & 0 & -\sin(\theta) \\ 0 & 1 & 0 \\ \sin(\theta) & 0 & \cos(\theta) \end{matrix}$$

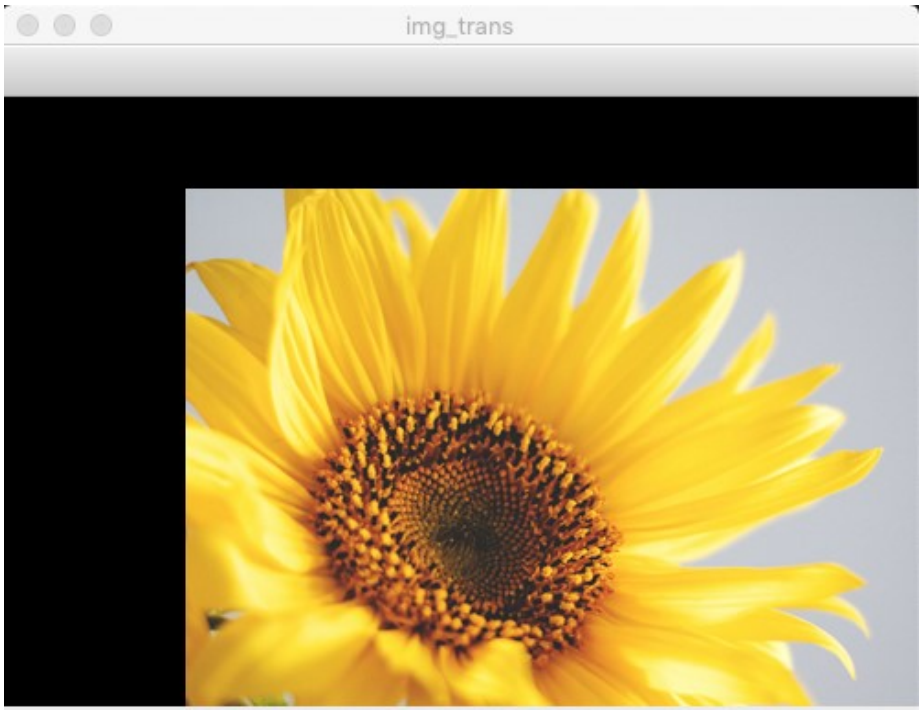
$$R_z = \begin{matrix} \cos(\theta) & \sin(\theta) & 0 \\ -\sin(\theta) & \cos(\theta) & 0 \\ 0 & 0 & 1 \end{matrix}$$

▼ What is a translation (transformation)?

- A translation is a transformation that moves an image by sliding it up, down, sideways, or diagonally without turning it or making it bigger or smaller.
- Translation matrices are the most simple transformation matrices to understand.
- A translation matrix in 3D looks like the matrix below where X, Y, Z are the values that you want to add to your position.

$$T = \begin{matrix} 1 & 0 & 0 & X \\ 0 & 1 & 0 & Y \\ 0 & 0 & 1 & Z \end{matrix}$$

- The following is a translated version of the 2D sunflower image. The top left-most corner is considered to be the origin. The image is translated 100 pixels to the right and 50 pixels down. Rightward and downward movement from the origin is in the positive direction.



- Here is the transformation matrix for this translation:

$$T = \begin{matrix} 1 & 0 & 100 \\ 0 & 1 & 50 \end{matrix}$$

▼ What is a pixel?

- A pixel is a sample of the image intensity quantized to an integer value.
- They are the smallest unit of information that makes up a picture.
- Usually round or square, they are typically arranged in a 2-dimensional grid.

▼ What 3 values does every pixel coordinate contain?

- Each pixel coordinate (x,y) contains three values ranging in intensities of 0 to 255.
- These three 8-bit number values represent the Red Green Blue (RGB) color channels.
- If all three values are at full intensity, that means they are all 255, then the pixel color shows as white. If all three values are muted, that means they are all 0, then the pixel color shows as black.
- The combination of these three values will, in turn, give you a specific shade of the pixel color.

▼ Example: access the pixels matrix and basic properties of the sunflower image in OpenCV

```
import numpy as np
import cv2
img = cv2.imread('sunflower.jpg')
print(img)

print('Shape of the image: {}'.format(img.shape))
print('Image Height: {}'.format(img.shape[0]))
print('Image Width: {}'.format(img.shape[1]))
print('Image Dimension: {}'.format(img.ndim))
```

Output:

```
[[[194 188 181]
  [196 190 183]
  [196 190 183]
  ...
  [169 163 156]
  [168 162 155]
  [166 163 155]]

 [[193 187 180]
  [193 187 180]
  [195 189 182]
  ...
  [169 163 156]
  [169 163 156]
  [166 163 155]]

 [[195 189 182]
  [193 187 180]
  [194 188 181]
  ...
  [166 163 155]
  [168 162 155]
  [167 164 156]]

 ...


Shape of the image: (332, 500, 3)
Image Height: 332
Image Width: 500
Image Dimension: 3
```

- The shape of the image is a 3 layered pixels matrix. The first two numbers are the image's length and width in pixels, and the third number tells you that the image has three color channels: Blue, Green, Red.

- Now, you can access any particular pixel's values as well as each BGR channel separately. For instance, access the values for the pixel located at row 100 and column 50 by passing in the appropriate pixel coordinates vector.

```
print('The values of the pixel located at row 100 and column 50 is: {}'.format(img[100,50]))

# The values of the pixel located at row 100 and column 50 is: [148 245 255]
```

- The output reveals that this pixel has values B = 148, G = 245, R = 255. This particular pixel has a lot of red in it.
- You could have also specifically selected a color channel by additionally passing in the channel's index value.
    - 0 index value for the Blue channel

- 1 index value for the Green channel
- 2 index value for the Red channel

```
print('Value of only the B channel {}'.format(img[ 100, 50, 0]))
print('Value of only the G channel {}'.format(img[ 100, 50, 1]))
print('Value of only the R channel {}'.format(img[ 100, 50, 2]))

# Value of only the B channel 148

# Value of only the G channel 245

# Value of only the R channel 255
```

▼ Example: display color channels of an image separately

- It is possible to extract and display each of these channels separately.
- Splitting the image into separate color components is just a matter of pulling out the correct slice of the image array.
- Use matplotlib to display the image's 3 color channels separately.
  - It is important to note that when displaying images using matplotlib, the color channels are reversed. That is, the color channels are not BGR as in OpenCV, they are RGB.

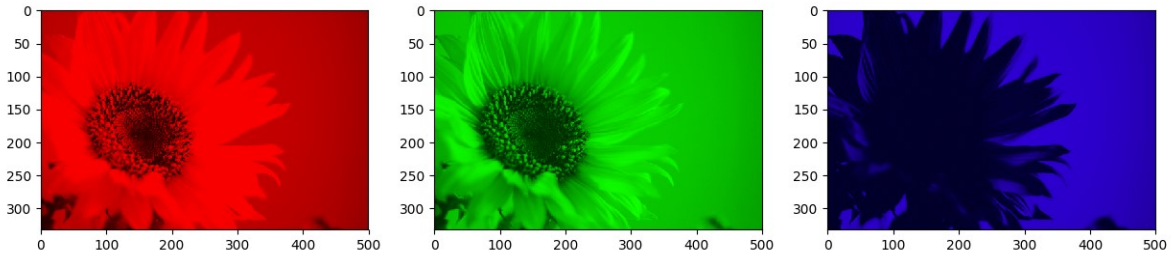
```
import matplotlib.pyplot as plt
fig, ax = plt.subplots(nrows = 1, ncols = 3, figsize = (15,5))

for c, ax in zip(range(3), ax):
#create zeros matrix
split_img = np.zeros(img.shape, dtype = 'uint8')

#access each channel
split_img[:, :, c] = img[:, :, c]

#display each channel
ax.imshow(split_img)

plt.show()
```



- Keep in mind that each of these color components are 3 separate matrices.
- Therefore, **matrix operations can be performed on these matrices**.
- For instance, take a look at this formula:
  - $Y = 0.3\ R + 0.5\ G + 0.2\ B$
- That is, you can edit the image by applying these operations.