

# Transformation

are geometric distortions enacted upon an image. We use transformation to correct distortions or perspective issues from arising from the point of view an image was captured.

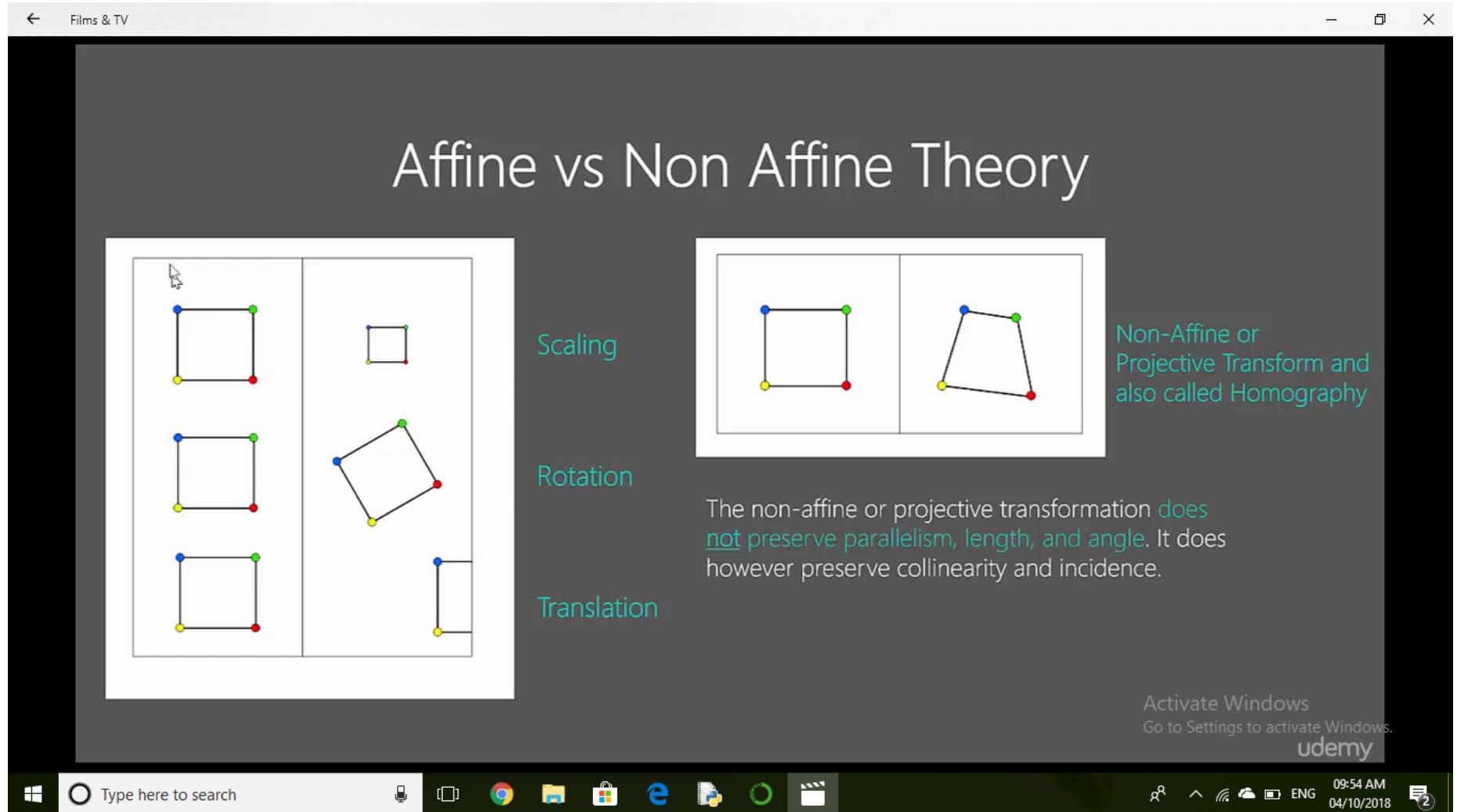
## Types:

- > Affine
- > Non-Affine

```
In [5]: 1 from IPython.display import Image
```

```
In [6]: 1 Image('tutorial images/affine vs non affine.png')
```

```
Out[6]:
```



## Translation

This is an affine transform that simply shifts the position of an image. We use `cv2.warpAffine` to implement these transformations

# warpAffine

## Translation matrix

```
In [7]: 1 Image('tutorial images/translation matrix.png')
```

Out[7]:

Translations

Translation Matrix

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

$T_x$  - Represents the shift along the x-axis (horizontal)  
 $T_y$  - Represents the shift along the y-axis (vertical)

We use the OpenCV function `cv2.warpAffine` to implement these translations

Active Windows  
Go to Settings to activate Windows.

udemy

```
In [18]: 1 import cv2, numpy as np
2 image = cv2.imread('my.JPG')
3 #store height and width of the image
4 height, width = image.shape[:2]
5 quarter_height, quarter_width = height / 4, width / 4
6
7 #      [1 0 Tx]
8 # T = [0 1 Ty]
9
10 #T is our transition matrix
11 T = np.float32([[1, 0, -quarter_width], [0, 1, -quarter_height]])
12
13 #We use warpAffine to transform the image using the matrix, T
14 img_translation = cv2.warpAffine(image, T, (width, height))
15 cv2.imshow('Translation', img_translation)
16 #cv2.imwrite('Translation.png', img_translation)
17 cv2.waitKey()
18 cv2.destroyAllWindows()
```

```
► In [15]: 1 print(T)

[[ 1.    0. -128.]
 [ 0.    1. -175.]
```

```
In [19]: 1 T2 = np.array([[1, 0, -333], [0, 1, 222]], np.float32)
2 T2
```

```
Out[19]: array([[ 1.,    0., -333.],
 [ 0.,    1.,  222.]], dtype=float32)
```