Department of Computer Engineering
CO543: Image Processing - Lab 1
13.01.2022

Section 1: Getting familiar with basic OpenCV functions.

1. Basics of Image read write and visualizations

Cv2 is the most commonly used Python interface released by OpenCV. This interface utilises Python's fast array processing library, NumPy. Hence, almost all the functions provided by OpenCV treats images as NumPy arrays. Therefore, many image processing functions can be achieved using NumPy operations.

Importing necessary libraries.

```
>>> import cv2
>>> import numpy as np
```

Loading an image in the working directory.

```
>>> img = cv2.imread('image.jpg' [, flags])
```
Flags specify the color type of a loaded image.

Example: To return a grayscale image:

```
>>> img = cv2.imread('image.jpg', 0)
```

Refer OpenCV documentation for the list of flags.
Note: If the image does not exist in the current directory, it will not throw an error. But print img will give you none.
Loading an image in a different directory.

```
>>> img = cv2.imread('/home/abc/CO543/image.jpg')
```
Saving an image in the working directory.

```
>>> img = cv2.imread('image.jpg')
>>> cv2.imwrite('saveBack.png',img)
```

Saving an image in a different directory.

```
>>> img = cv2.imread('image.jpg')
>>> cv2.imwrite('/home/abc/CO543/saveBack.png',img)
```

Viewing an image.

```
>>> img = cv2.imread('image.jpg')
>>> cv2.imshow('image',img)
>>> cv2.waitKey(0)              # Wait indefinitely for a keystroke
>>> cv2.destroyAllWindows() # Destroy all created windows upon a keystroke
```

Viewing image details.

```
>>> img = cv2.imread('image.jpg')
>>> print(img.shape) # Number of rows and columns and number of channels (if image is color)
>>> print(img.size) # Number of pixels
```

2. **Image type conversions** viewing image data type.

```
>>> print(img.dtype)
```

Converting from RGB to Grayscale

```
>>>  img=cv2.cvtColor(im,cv2.COLOR_RGB2GRAY)
```

Refer OpenCV documentation for all possible conversions.

3. **Basic Image Processing commands**

Resizing

```
>>> resized_image = cv2.resize(img, (100, 50)) #Reduce image to 100 cols and 50 rows
```

```
>>> resized_image = cv2.resize(img, (0, 0), fx=0.5, fy=0.5) #Reduce both axes by half
```

```
>>> resized_image = cv2.resize(img, (0, 0), fx=0.5, fy=0.5, interpolation=cv2.INTER_NEAREST)    #Specify
interpolation method
```

Rotation

```
# Get dimensions of the image and calculate the center            of the image
>>> (h, w) = img.shape[:2]
>>> center = (w / 2, h / 2)
```

```
# Computing the matrix (M) that can be used for rotating the image
# center - Point around which, the image is rotated
# 180 - Angle by which image is rotated
# 1.0 - Scaling factor (No scaling in this case)
>>> M = cv2.getRotationMatrix2D(center, 180, 1.0)
```

```
# Perform the actual rotation
>>> rotated = cv2.warpAffine(img, M, (w, h))
```

Cropping

Cropping can be achieved using basic numpy array slicing.

```
# startY,startX - starting coordinates where cropping should begin.
# endY,endX - ending coordinates
>>> cropped = image[startY:endY, startX:endX]
```

Complementing

Bitwise-not function can be used for complementing an image.

```
>>> inverted = cv2.bitwise_not(img)
```

Flipping an image

```
# flipMode - If 0, flipping around x-axis
#            If positive, flipping around y-axis
#            If negative, flipping around both axes >>> flipped =
cv2.flip(img, flipMode)
```

Section 2 : CSV to Image conversion

Usually the small size images are stored as CSVs, especially in datasets like MNIST data sets. Here is an example of reading a dataset similar and visualize the images.

```
>>>import pandas as pd
>>>import matplotlib.pyplot as plt
>>>import numpy as np
```

Read the given CSV file
```
>>>img_data = pd.read_csv('Digits_Lab_01.csv')
```
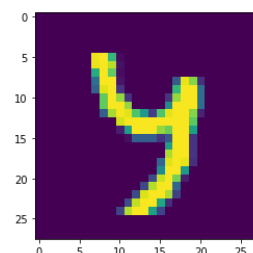
Visualize the data as a table
```
>>> img_data.head()
```

Reshape the rows of data to images (image size – 28x28x1)
```
>>>Reshaped_images = img_data.to_numpy().reshape(-1,28,28,1)
```

Visualize the images
```
>>>sample_img =
np.array(Reshaped_images[12]).reshape((28,28))
>>>plt.imshow(sample_img)
>>>plt.show()
```

Section 3 : Geometric Transformations

Now a days computer vision and image classification has mostly done using deep learning and other machine learning algorithms, due to lack of data and lack of representation of data geometric transformations has been used to augment data.

Some most common geometric transformations are shown down bellow
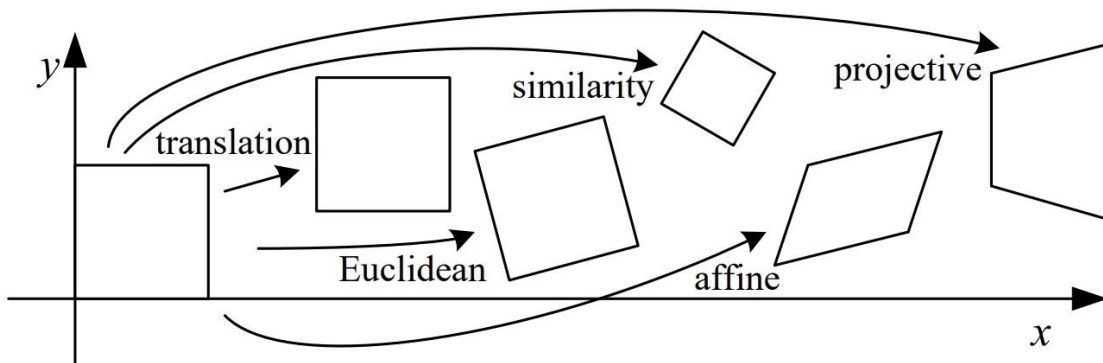
Example of programing Affine Transformation

Read the image and check for the shape
```
>>>img =cv.imread('eb9931c3f39215a193826992d013a17a.jpg')
>>>rows,cols,ch = img.shape
```
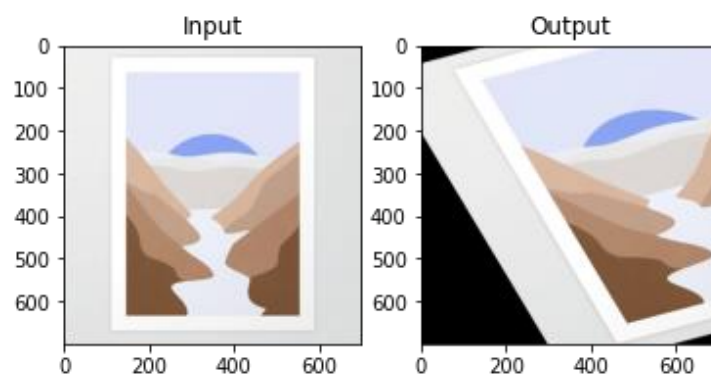
Define the Transformation
```
>>>pts1 = np.float32([[50,50],[200,50],[50,200]])
>>>pts2 = np.float32([[10,100],[200,50],[100,250]])
>>>M = cv.getAffineTransform(pts1,pts2)
```

Do the Transformation
```
>>>dst = cv.warpAffine(img,M,(cols,rows))
```

Visualize the original and the transformed images
```
>>>plt.subplot(121),plt.imshow(img),plt.title('Input')
>>>plt.subplot(122),plt.imshow(dst),plt.title('Output')
>>>plt.show()
```

Section 4 : Lab tasks

Implement basic image processing functions using Python-OpenCV

As you are aware by now, OpenCV library provides many built-in functions to perform basic image processing functions such as complementing, resizing, cropping, rotation and flipping.

1.   Implement the following functions on your own using PythonOpenCV.

   a   imcomplement(I) - Inverts I
   b   flipud(I) - Flips image along x-axis
   c   fliplr(I) - Flips image along y-axis
   d   imresize(I,[x y]) with nearest-neighbour interpolation

2.   Implement the 4 geometric transformation functions using OpenCV in addition to the given example.

Input argument (I) to the function is a NumPy array which is the result of the cv2.imread() function. It should be converted to grayscale before passing to the function for the 1st task.

Submission

Submit zipped notebook and a PDF report file with test images. exxyyy_lab01.zip, where exxyyy is your registration number.