Homework1 , CS 682 Spring20
S M Hasan Mansur

*** script filename: **hw1_cs682_spring20_smansur4.py**
*** script file has been submitted along with this report via Blackboard
*** Please find the results/outcomes of the script in http://mason.gmu.edu/~smansur4/

Problem 1

Installed opencv-python (version 4.1.2.30)
Script has been developed & tested using Python 3.7.4
Script can be run like following:
python hw1_cs682_spring20_smansur4.py -i /path/to/imagefile

Problem 2

For reading an image the following method has been used:
cv2.imread(image_path)
where image_path *represents the path of the image to be read.*

Following method has been used to convert the color image into grayscale image:
cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)

Problem 3

I have implemented the following transformations/changes with the color & gray versions:

Translation: Color image translated Right & Down

First, a transformation matrix is defined as follows where x_shift & y_shift represent the shift along x axis & y axis respectively.
transformation_matrix = np.float32([[1, 0, x_shift], [0, 1, y_shift]])

Then, the shifted image matrix is calculated as follows:
shifted = cv2.warpAffine(image, transformation_matrix, (image_width, image_height))
Where image_width & image_height represent the width & height of original image.

Rotation: Color image rotated 270 degrees

First, a transformation matrix is defined as follows:
transformation_matrix = cv2.getRotationMatrix2D(center, angle, scale)
Where, center – Center of the rotation in the source image,
angle – Rotation angle in degrees,
scale – scale factor.

Then, the rotated image matrix is calculated as follows:
rotated = cv2.warpAffine(image, transformation_matrix, (image_width, image_height))
Where image_width & image_height represent the width & height of original image.

Reflection: Gray scale image reflected vertically

Vertical reflection on the grayscale image is implemented as follows:
cv2.flip(gray, 0), where gray represents the grayscale image

Color space change: BGR to HSV

Color space change from BGR to HSV is implemented as following:
cv2.cvtColor(image, cv2.COLOR_BGR2HSV)

Shearing: Gray scale image sheared horizontally

For horizontal shear, a shear matrix is created as follows:
np.float32([[1, 0.5, 0], [0, 1, 0]])
Then, the sheared image is calculated as follows:
hor_sheared = cv2.warpAffine(gray,shear_M,(image_width, image_height))
Where image_width & image_height represent the width & height of original image.

Blurring: Color image blurred

Following method has been used for blurring the color image
cv2.blur(image,(10,10))

Resizing: Color image resized to double

First, the new dimension is calculated as follows:
dim = (2 * image_width, 2 * image_height)
Then the resized image matrix is calculated as follows:
resized = cv2.resize(image, dim, interpolation = cv2.INTER_CUBIC)

Problem 4

To create the Gaussian Pyramid, the original image is successively downsampled until some desired stopping point is reached. For downsampling, **cv2.pyrDown()** method is used & downsampling is done until the dimension becomes 1x1. A list of sampled images is maintained so that they can be packed together in a single large image.

To pack all the sampled images of different sizes into a single image, first the total height of all the sampled images & the max width are calculated. Then a numpy array is created using these dimensions (total height & max width). Finally, we concat one by one sample image into the numpy array with the order of smallest to the largest. Thus, I implemented the packing of different images into a single image.

space requirement for the pyramid image:
width: 636 pixels, height: 1692 pixels

size of the smallest rectangular image needed to pack the pyramid:
width: 1 pixel, height: 1 pixel

Problem 5

Interesting application:
Blood monitoring system to estimate real time blood loss during medical situations.
Domain:
Healthcare
Developed By:
Gauss Surgical, CA, USA
Link:
https://www.gausssurgical.com/