Transformations

Importing libraries:

*import cv2 as cv*

*import numpy as np*

# Translation:

It basically shifts or translates the image along x and y axes.

*# Read an image*

*img = cv.imread(‘Path to the img’)*

*# Translation Function:*

*def translate(img,x,y):*

*transMat = np.float32([[1,0,x], [0,1,y]])*

*dimensions = (img.shape[0], img.shape[1])*

*return cv.warpAffine(img, transMat, dimensions)*

*# Calling the function*

*translated = translate(img,100,100)*

*cv.imshow(‘Translated’, translated)*

Explanation:

The def translate() func takes three parameters:

* img: The image to translate
* x: The translation distance along x-axis
* y: The translation distance along y-axis

To shift an image we need a transformation matrix called **affine transformation matrix.** In mathematics, affine transformation is defined as:

An affine transformation is any transformation that preserves collinearity (i.e., all points lying on a line initially still lie on a line after transformation) and ratios of distances (e.g., the midpoint of a line segment remains the midpoint after transformation.

In computer vision, it is typically used to correct for geometric distortions or deformations that occur with non-ideal camera angles.

*transMat = np.float32([[1,0,x], [0,1,y]])*

In this line the transformation matrix is defined, **np.float32** specifies that the numbers in the matrix should be 32-bit floating-point numbers (i.e., single-precision floating-point values), which is a requirement of opencv for compatibility reasons.

[1, 0, x]: This specifies the movement along the x-axis. x is the translation amount in pixels.

[0, 1, y:] This specifies the movement along the y-axis. y is the translation amount in pixels.

Note that the translation amounts in pixels are taken as parameters to the function.

The shape of the matrix is always 2 rows and 3 columns, as it is a standard 2D affine transformation matrix.

Then, the dimensions of the image are stored in the dimensions tuple. Remember, the value lying at 0th index is the width and 1st index is the height of the image.

The warpAffine() is a built-in function of opencv which takes the following parameters:

* img
* transMat
* dimensions

It applies an affine transformation to the image. In this case, it uses the transMat translation matrix to shift the image by x pixels along the x-axis and y pixels along the y-axis.

The dimensions argument ensures the translated image has the same size as the original.

And then finally, you call the function and pass in the parameter values and then display the image.

The following values specify in which direction you want to shift the image:

# -x --> Left

# -y --> Up

# x  --> Right

# y  --> Down

# Image Rotation:

# Rotating an Image

*def rotate(img, angle, rotpoint=None):*

*(height, width) = img.shape[:2]*

*if rotpoint is None:*

*rotpoint = (width//2, height//2)*

*rotMatrix = cv.getRotationMatrix2D(rotpoint,angle,scale=1.0)*

*dimensions = (width, height)*

*return cv.warpAffine(img,rotMatrix,dimensions)*

The rotate() function takes following arguments:

* img
* angle: This is the angle by which you want to rotate an image
* rotpoint: rotation point which is none by default

Explanation:

Height and width are stored, and if the rotation point is none, we obtain it by integer division of width and height by 2 and stored in the rotpoint tuple.

The getRotationMatrix2D(rotpoint,angle,scale=1.0), since we do not want to do scaling so we set the default float value of 1.0. By this we obtain the rotation matrix. And finally warpAffine() method is used to rotate the image.

*rotated = rotate(img,270) # You can try with different angles. Try with negative values*

*cv.imshow('Rotated', rotated)*

With positive angle values, the image rotates anti clock-wise, if you want to rotate the image clock-wise use negative angles. You can also rotate a rotated image.

# Flipping:

*# Flipping an image*

*flip = cv.flip(img, -1)*

*# cv.imshow("Original", img)*

*cv.imshow('Flipped', flip)*

The flip function takes two parameters, firstly the image that is to be flipped, and then the flip code which is either 0, 1, or -1.

0 implies flipping the image vertically that is the x-axis.

1 implies flipping the image horizontally over the y-axis.

-1 implies flipping the image both vertically and horizontally.