# Image processing miniproject – Rotation

## Algorithm

A rotation in image processing is performed by using the rotation matrix. This matrix is used to calculate the new x and y coordinates for a pixel in the image, also called x’ and y’. The rotation matrix for forward mapping is illustrated bellow.



The matrix uses an angle, cosine and sine to calculate the x’ and y’ points of the pixels in the image. This is because cosine and sine values generate a circle called the unit circle. Therefor cosine to an angle multiplied by x, will be two points that lies on the same circle arc. In order to only get one point, one must subtract y multiplied by sin to angle. The same rules apply when calculating a rotation on a y coordinate, though in this case it is x multiplied by sine of an angle plus y multiplied by cosine of an angle.

The rotation performed to an image when using forward mapping is done counter clockwise. Using forward mapping might also cause holes in the image. These holes can mostly be avoided by using backward mapping. The difference in the algorithm between backward and forward mapping, is that index 1,2 I changed from negative to positive and that index 2,1 is changed to negative. Using backward mapping to rotate an image will rotate the image clockwise. The rotation matrix for backward mapping is illustrated below.



## Code

import cv2

import math

import numpy as np

class rotator:

angle = 30.0

x = 330

y = 330

radians = float(angle\*(math.pi/180))

img = cv2.imread('lena.jpg',0)

width,height = img.shape

def showImg(name, self):

cv2.imshow(name, self.img)

self.img = np.pad(self.img, (self.height) ,'constant', constant\_values=0)

self.width,self.height = self.img.shape

def printWH(self):

print(self.width)

print(self.height)

def getImage(self):

return self.img

#Rotates an image using forward mapping

def forward(self, img):

empty = np.zeros((self.width,self.height),dtype="uint8")

for i in range(self.width):

for j in range(self.height):

#forward mapping

x = int((i-self.x)\*math.cos(self.radians)-(j-self.y)\*math.sin(self.radians))+self.x

y = int((i-self.x)\*math.sin(self.radians)+(j-self.y)\*math.cos(self.radians))+self.x

if x < self.width and y < self.height and x>0 and y > 0:

empty[i,j] = self.img[int(x),int(y)]

else:

pass

return empty

#Rotates an image using backward mapping

def backward(self, img):

empty = np.zeros((self.width,self.height),dtype="uint8")

for i in range(self.width):

for j in range(self.height):

#forward mapping

x = int((i-self.x)\*math.cos(self.radians)+(j-self.y)\*math.sin(self.radians))+self.x

y = int(-(i-self.x)\*math.sin(self.radians)+(j-self.y)\*math.cos(self.radians))+self.x

if x < self.width and y < self.height and x>0 and y > 0:

empty[i,j] = self.img[int(x),int(y)]

else:

pass

return empty

#Rotates an image using forward mapping and then rotates it back using backward mapping

def backwardForward(self, img):

empty = np.zeros((self.width,self.height),dtype="uint8")

for i in range(self.width):

for j in range(self.height):

#forward mapping

xO = int((i-self.x)\*math.cos(self.radians)-(j-self.y)\*math.sin(self.radians))+self.x

yO = int((i-self.x)\*math.sin(self.radians)+(j-self.y)\*math.cos(self.radians))+self.x

x = int((xO-self.x)\*math.cos(self.radians)+(yO-self.y)\*math.sin(self.radians))+self.x

y = int(-(xO-self.x)\*math.sin(self.radians)+(yO-self.y)\*math.cos(self.radians))+self.x

if x < self.width and y < self.height and x>0 and y > 0:

empty[i,j] = self.img[int(x),int(y)]

else:

pass

return empty

def main():

rotator.showImg('normal', rotator)

rotator.printWH(rotator)

cv2.imshow('forward', rotator.forward(rotator, rotator.getImage(rotator)))

cv2.imshow('backward', rotator.backward(rotator, rotator.getImage(rotator)))

cv2.imshow('foward image backward', rotator.backwardForward(rotator, rotator.getImage(rotator)))

cv2.waitKey(0)

cv2.destroyAllWindows

if \_\_name\_\_ == '\_\_main\_\_':

main()

## Explanation of code

## Documentation of program