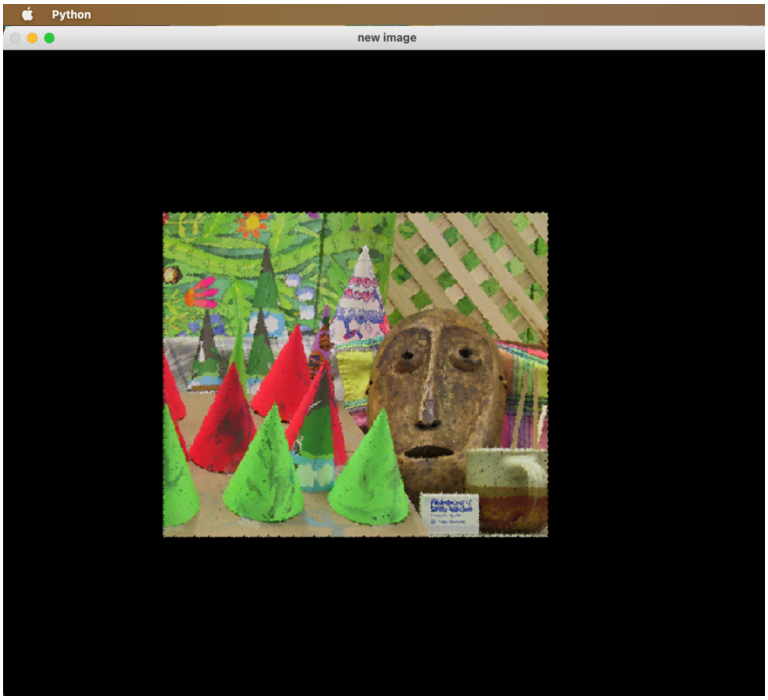
Image rotation assignment

Kareem Tiwari

Creating the border image:

Before the image rotation assignment, we had to take the cones image and make a new empty image so that we were able to copy and the old cones image into the new one with a black border around it spaced evenly. So, we begin this rotation assignment with the new cones image with the black border around it which looks like:



This is the code to create the new cones image with the black border around it:

```
import cv2
import numpy as np
import math

image = cv2.imread("cones1.png")
numRows = image.shape[0]
numCols = image.shape[1]
size1= 2*numRows
size2=2*numCols
cx = size2/2 # grabiing the x -coordinate
cy = size1/2 # grabing the y cooridinate
step = ((math.pi)/4)
theta = step #np.radians(step)
cos = np.cos(theta)
sin = np.sin(theta)
```

```
R = [[cos],[-sin],
    [sin],[cos]]

emptyIm_2= np.zeros((size1,size2,3),np.float32)
newSize=int((size1-numRows)/2)

for i in range(numRows):
   for j in range(numCols):

      emptyIm_2[newSize+i][j+newSize][0] = image[i][j][0]
      emptyIm_2[i+newSize][j+newSize][1] = image[i][j][1]
      emptyIm_2[i+newSize][j+newSize][2] = image[i][j][2]



#xr = cos(cx)-sin(cy) # rotated x coordinate
#yr = sin(cx)+cos(cy) # rotated y coordinate


emptyIm = np.zeros((size1, size2, 3),np.float32)
cv2.imshow("new image", emptyIm_2/255.0)
```

Rotating the new cones image:

Now that the cones image has its black border we have to use a rotation matrix to rotate the new cones image about the center point while rotating at an increment of 45 degrees per rotation. This is the code I used for the rotation of the new cones image.

Code below:

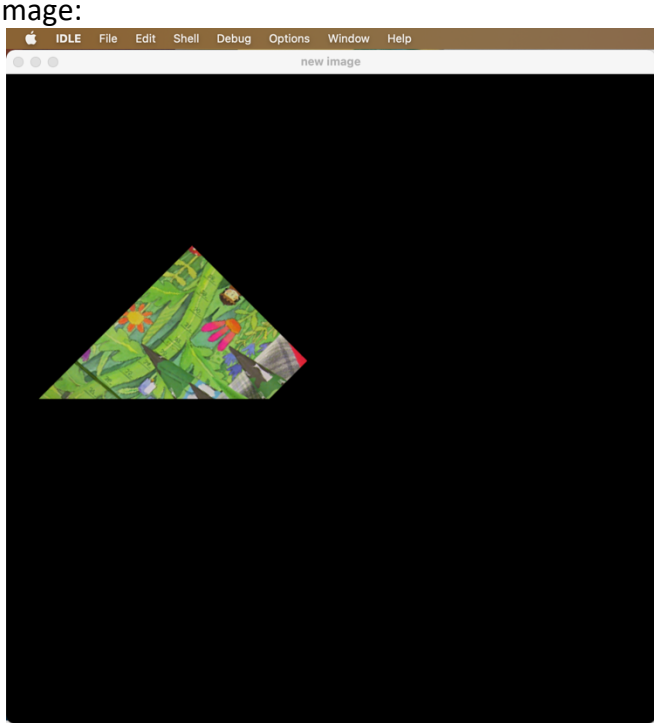
```
emptyIm = np.zeros((size1, size2, 3),np.float32)
cv2.imshow("new image", emptyIm_2/255.0)
while theta <= (2*(math.pi)):#360: #I tried to replace 360 with (2*(math.pi)).
   emptyIm = np.zeros((size1, size2, 3),np.float32)
   theta += step
   for i in range(size1):
      for j in range(size2):
         tx = j-cx
         ty = i-cy
         t = [tx,ty]  # move origin
         r = t #rotate pt
         fx = tx - cx
         fy = ty - cy
         #f = r + (cx,cy)
```

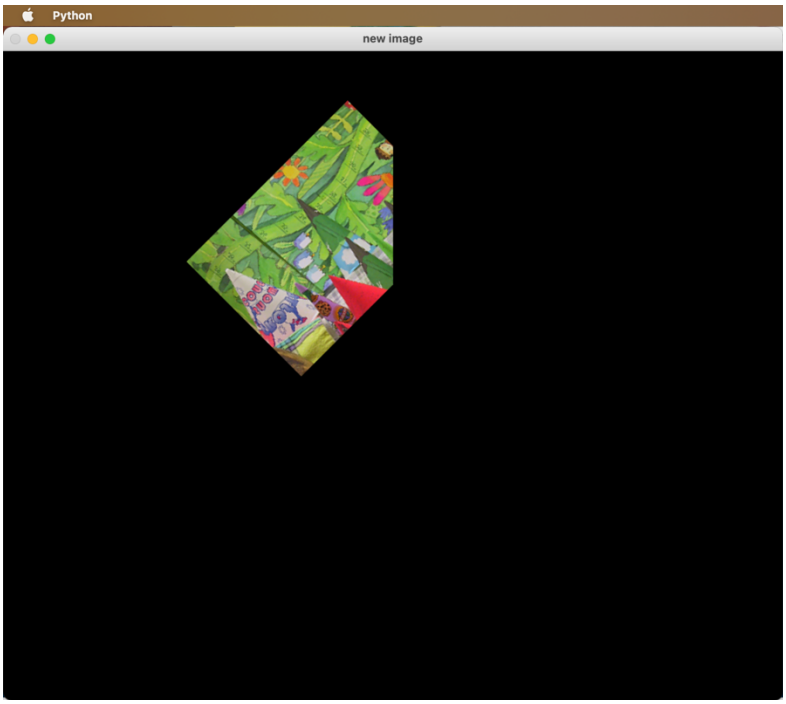
```
xr = (R[0][0])*(t[0])+(R[1][0])*(t[1]) # rotated x coordinate
yr = (R[2][0])*(t[0])+(R[3][0])*(t[1])# rotated y coordinate
a = int(xr +cx)
b = int(yr +cy)


if a>0 and b>0:
   if a < int(size2) and b< int(size1):
```

Problems:

One of the problems that I ran into while creating this includes the loss of some of resolution as well as loss of pieces of the image as it is in the process of rotation. As Each step was being completed the image would lose pieces near the border of the image along with resolution. Below are some of the images of the new cones image losing resolution and pieces of the image:





Final image:

Finally, this is how the fully rotated image with the most minimal loss in resolution and from the image itself. The image is shown below: