TNM087 - Image Processing and Analysis

Task 7 – Image Rotation

Background:

Geometric transformations are one of the basic tools in image processing and computer vision. Rotations are a good example illustrating the problems related to coordinate changes.

This function illustrates the changes between the ordinary Euclidean and the polar coordinate system. You also have to implement some kind of interpolation since a grid point in one system will not be mapped to a grid point in the other. Read Chap. 2.1 in Szeliski: Computer Vision for a description or Section 2.4.4 and Section 2.6.5 in Gonzalez-Woods.

3D rotations are much more complicated as you can see from the description in Chap. 2.1.4 in Szeliski: Computer Vision

Task:

Rotate an image around a given center point. To pass it is sufficient to use nearest neighbor interpolation. Using more complicated interpolations (like bilinear) is optional, but highly recommended.

Syntax:

```
Function  RImage = FRotate(OImage, center, degangle )
```

Hints:

Read the Matlab help pages for meshgrid, cart2pol and pol2cart.

center is a vector with two elements center=(C1,C2) where C1,C2 are the coordinates of the center of the rotation. Here C1 gives the position in x-direction and C2 in y-direction. It OImage has 128 rows (height) and 256 colums (width)   and center is (64,128) then you should rotate around the center of the image.

degangle is the rotation angle in degrees and the direction of the rotation is CLOCKWISE

Pixel values for points outside the original image are set to a constant (for example 0 or 255)

For debugging it is perhaps easiest if you generate a rectangular black image with a horizontal white line as OImage. Then it is easy to see if the result is correct. You may use T7Cross.jpg or T7Checkerboard.jpg for testing.

Do NOT use the built-in interpolation functions in Matlab

<u>Optional:</u>

1) In the template the coordinate vectors are stored in arrays like C, R, Theta, Rho, nC, nR etc. For large images that can be a waste of memory. Can you implement the rotation without them (for example using functions)? Which version is faster?
2) There are two approaches how to implement geometrical transformations of images: (A) Start with a pixel in the original image, compute its destination in the new image and set the pixel value at the destination position to the pixel value at the original position. (B) Start with a pixel in the destination image, compute its position in the original image and fetch the pixel value at the destination position from the pixel value at the original position (see lecture). Which one is more efficient?