IP Homework Set #2 – Geometric Operations on Images

Due : 19/12/16

Submission: Mark the files with your name and i.d.

Email the Hw including all m-files, scripts and enhanced images

to: imageprocessinghaifau@gmail.com

Task: Write matlab routines that perform Geometric operations on images.

<Each function start with a new page>

1. **function newimg = rotateImage(img,theta,centerX,center,newSize)**
   - (file name is accordingly **rotateImage.m**)
   - This routine rotates img by theta degrees ANTI-clockwise around the point [centerX,center].

Input:

- img - a grayscale image in the range [0..255]
- theta – angle in degrees anti-clockwise from x-axis.
- centerX,centerY – coordinates of center of rotation (may be outside image).
- newSize – **optional parameter** – [newRows,NewCols] - defines the size of the output image. May be smaller or larger than size of img. Default is size of img.

Output:

- newimg – the rotated image.

Method:

- **Performs the geometric operation using inverse mapping.**
- Creates an array of all target pixel coordinates.
- Calculates the transform to apply on these coordinates.
- Applies the transform in matrix operations on the target pixel coordinates to obtain the source pixel coordinates.
- Use the source pixel coordinates to calculate the color in the source image (img) using Bilinear Interpolation.
- Pixels that have no source (outside the image) should be painted black (0).

Notes:

- Use matlab function **meshgrid** to create an array of all pixel coordinates: [Xcoords,Ycoords] = meshgrid(1:imSizeX,1:imSizeY)
- Do **NOT** loop over the image when applying the transform.
- Use the function **interpolate.m** that you have written
- Use Matlab function **nargin** to determine if number of inputs is less than 5 (newSize not given)

2. **function newimg = scaleImage(img,s,centerX,center,newSize)**
   - (file name is accordingly scaleImage.m)
   - This routine scales img uniformly by s around the point [centerX,center].

Input:

- img - a grayscale image in the range [0..255]
- s – the scale factor must be greater than 0.
- centerX,centerY – coordinates of center of scaling (may be outside the image).
- newSize – optional parameter – [newRows,NewCols] - defines the size of the output image. May be smaller or larger than size of img. Default is size of img.

Output:

- newimg – the scaled image.

Method:

- Performs the geometric operation using inverse mapping.
- Creates an array of all target pixel coordinates. Calculates the transform to apply
  on these coordinates.
- Applies the transform in matrix operations on the target pixel coordinates to obtain the source pixel coordinates.
- Use the source pixel coordinates to calculate the color in the source image (img) using Bilinear Interpolation.
- Pixels that have no source (outside the image) should be painted black (0).
  Notes:
- Use matlab function meshgrid to create an array of all pixel coordinates:
  [Xcoords,Ycoords] = meshgrid(1:imSizeX,1:imSizeY)
- Do NOT loop over the image when applying the transform.
- Use the function interpolate.m that you have written
- Use Matlab function nargin to determine if number of inputs is less than 5 (newSize not given)

3. **function newimg = affineImage(img,sourceCoors,targetCoors,newSize)**
   - (file name is accordingly affineImage.m)
   - This routine applies an affine transformation that maps sourceCoors to targetCoors.

Input:

- img - a grayscale image in the range [0..255]
- sourceCoors – a 2XN array of coordinates [x,y]' in the source image (img)
- targetCoors – a 2XN array of coordinates [x,y]' in the target image (newimg)
- newSize – optional parameter – [newRows,NewCols] - defines the size of the output image. May be smaller or larger than size of img. Default is size of img.

Output:

- newimg – affine transformed image such that sourceCoors are mapped to targetCoors.

Method:

- Performs the geometric operation using inverse mapping.
- Calculates the transform to apply by determining the affine transformation that best maps the source pixels to the target pixels (best under norm2).
- Use the PINV approach described in class to find the vector of affine parameters (a..f).
- Applies the transform in matrix operations on the target pixel coordinates to obtain the source pixel coordinates.
- Use the source pixel coordinates to calculate the color in the source image (img) using Bilinear Interpolation.
- Pixels that have no source (outside the image) should be painted black (0).

Notes:

- Use matlab function meshgrid to create an array of all pixel coordinates: [Xcoords,Ycoords] = meshgrid(1:imSizeX,1:imSizeY)
- Use matlab function pinv (however note that there is a righthand pinv and a lefthand pinv – so test that you are using correctly).
- Do NOT loop over the image when applying the transform.
- Use the function interpolate.m that you have written
- Use Matlab function nargin to determine if number of inputs is less than 5 (newSize not given)
- To provide input for this function, use Matlab's ginput function.

4. **function sourceGrayVals = interpolate(img,sourceCoors)**
   - (file name is accordingly interpolate.m)
   - This routine calculates the gray values for each coordinate by interpolating the values in img.

Input:

- img - a grayscale image in the range [0..255]
- sourceCoors – a 2XN array of coordinates [x,y]' in the source image (img)

Output:

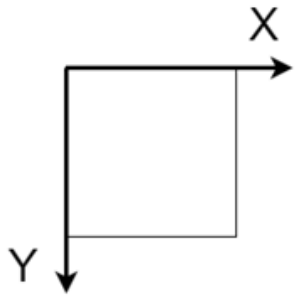
- sourceGrayVals – a 1XN vector of gray values.

Method:

- Performs Bilinear interpolation to evaluate the gray value of the sourceCoors within image img.
- Use Matrix operations only when computing interpolation.
- Pixels that have no source (outside the image) should assigned gray value 0 (black).
- Notes:
- You must be careful not to attempt to access pixels outside image.
- Yet simply erasing these pixels from sourceCoors is not advisable as you will later have to insert gray value 0 for these pixels in the output vector.
- There are many approaches to dealing with this. One trick is to replace all such coordinates with the coordinate [1 1] (which will not cause out of image access).
- Remember these pixels and later replace their gray value with 0. You do not necessarily need to use this method.
- To access matrix entries of a set of coordinates you can use matlab function sub2ind.
- You may use matlab functions floor, ceil. Do not use functions interp.
- Note, on return from this function the vector of gray values can be reshaped to target image size using matlab function reshape.

5. **Write a script that performs the following:**
   - You may choose an image from the course toolbox or any other image in which case, please submit the image as well. You may use a different image per task.
   - a. Show that rotating an image and then rotating it back, does not revert an image back perfectly to the original (due to digitization and interpolation artifacts). Show this also for scaling an image.
   - b. Does rotating then scaling an image produce the same result as first scaling and then rotating with same parameters (up to errors in interpolation/digitization.
   - c. Show an example where an affine transformation does not suffice to map points (a Projective transformation is needed). Hint: use image of railroad with vanishing point. Provide the BODEK with the 2 vectors of coordinates that you are using.
     To do this save the vector (and any other variable you would like to hand in) as a mat file using function save. EXPLAIN why there is no affine transformation in these cases.
   - d. Show an example image where the affine transform maps the SourceCoors to the TargetCoors exactly. Use at least 4 point pairs. Provide the BODEK with an effective way to show this example, i.e. to prove that indeed the the source is mapped to the target.

   - ImageAxes



Important Note:

- The image origin is the top left corner.
  X-axis increases to the right (xaxis <-> cols)
  Y- axis increases downwards (yaxis <-> rows)
- The top left pixel is at coordinate (1,1).
  Thus to rotat/scale so that this pixel remains in place use (centerX,center) = (1,1).

TIPS:

- Start by implementing Nearest Neighbor. See that everything works and then replace with Bilinear interpolation.
- Several images are available in the TOOLBOX (under Homework / Matlab in the course web site).
- You can and should invent/create smart test images that will assist you in evaluating the results of your program.
- You can use the functions in TOOLBOX supplied for reading/writing/viewing images.
- You may also use the matlab functions for reading/writing images but it is YOUR responsibility to make sure it will work on all inputs.
- Do not forget to write documentation in the functions!!!
- Program will be tested on given inputs as well as on new inputs.
- Grade will depend on correct performance and on clean programming and documentation.
- Do not forget to put Names and Student I.D. in the Documentation.