CSE 166: Image Processing, Spring 2019 – Assignment 1

Instructor: Ben Ochoa

Due: Wednesday, April 10, 2019, 11:59 PM

Instructions

- Review the academic integrity and collaboration policies on the course website.
- This assignment must be completed individually.
- This assignment contains both math and programming problems.
- Programming aspects of this assignment must be completed using MATLAB.
- Unless specified below, you may not use MATLAB functions contained in toolboxes, including the image processing toolbox. Use the MATLAB `which` command to determine which toolbox a function is contained in. If you are unsure about using a specific function, then ask the instructor for clarification.
- You must prepare a report as a pdf file. The report must contain your solutions and results, and all of your MATLAB source code as a listing in the appendix of your report.
- Additionally, you must create a zip file containing all of your MATLAB source code.
- Your source code must contain a file `main.m` which runs all code necessary to produce results for your report. `main.m` should run start to finish without error. Use relative paths to read input data. For questions which require numerical output, `main.m` should print a message indicating what question is being answered followed by the numerical output for that question. Example: `display('Problem 2b')` followed by your answer to problem 2b. The instructors should be able to reproduce your report by running `main.m`
- You must submit both files (.pdf and .zip) on Gradescope. Further, you must mark each problem on Gradescope in the pdf file.
- It is highly recommended that you begin working on this assignment early.

Problems

1. **2D transformation matrices (5 points)**

   Given the 2D transformation matrices

   $$\mathtt{H}_\mathtt{t} = \begin{bmatrix} 1 & 0 & t_x \\ 0 & 1 & t_y \\ 0 & 0 & 1 \end{bmatrix} \text{ and } \mathtt{H}_\mathtt{R} = \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

   show that $\mathtt{H} = \mathtt{H}_\mathtt{t}^{-1}\mathtt{H}_\mathtt{R}\mathtt{H}_\mathtt{t}$ is a 2D Euclidean transformation matrix.

2. **Programming: Transform images (30 points)**

   (a) **2D transformation matrix (5 points)**

   Develop a MATLAB function called `rotateAboutCenterTransformation` that calculates the 2D transformation matrix that rotates an image about its center. The function inputs are image width, image height, and rotation angle. The function output is the 2D transformation matrix.

   Include the numerical results output of the function `rotateAboutCenterTransformation` with image width = 640, image height = 480, and rotation angle = $\pi/6$ in your report with sufficient precision such that it can be evaluated (hint: use `format shortg` in MATLAB prior to displaying your results).

   (b) **Image transformation and interpolation (25 points)**

      i. **Nearest neighbor interpolation (10 points)**

      Develop a MATLAB function called `transformImageNearestNeighbor` that transforms an image using the nearest neighbor interpolation method. The function inputs are an image and a 2D transformation matrix. The function output is the transformed image. The function must set pixels to black in the output image that inversely map to pixels outside of the input image boundaries.

      Develop a MATLAB script called `hw1_nearest_neighbor.m` that uses `imread` to read the input image `cameraman.tif` (included with MATLAB); rotates the image about the center at rotation angles $\pi/6$, $\pi/4$, and $\pi/2$; and writes the corresponding output images to `cameraman_nearest_neighbor_rot30.png`, `cameraman_nearest_neighbor_rot45.png`, and `cameraman_nearest_neighbor_rot90.png`. The script must call the function `rotateAboutCenterTransformation` to calculate each 2D transformation matrix (hint: use the `size` function to determine the width and height of the image). The script must call the function `transformImageNearestNeighbor` to apply the transformation. Use `imwrite` to write each output image in MATLAB.

      Include in your report the input image and the output image for each rotation angle.

      ii. **Linear interpolation (15 points)**

      Develop a MATLAB function called `transformImageLinear` that transforms an image using the linear interpolation method. The function inputs are an image and a 2D transformation matrix. The function output is the transformed image. The function must set pixels to black in the output image that inversely map to pixels outside of the input image boundaries.

      Develop a MATLAB script called `hw1_linear.m` that uses `imread` to read the input image `cameraman.tif`; rotates the image about the center at rotation angles $\pi/6$, $\pi/4$, and $\pi/2$; and writes the corresponding output images to `cameraman_linear_rot30.png`, `cameraman_linear_rot45.png`, and `cameraman_linear_rot90.png`. The script must call the function

`rotateAboutCenterTransformation` to calculate each 2D transformation matrix (hint: use the `size` function to determine the width and height of the image). The script must call the function `transformImageLinear` to apply the transformation. Use `imwrite` to write each output image in MATLAB. Include in your report the output image for each rotation angle. Comment on the qualitative differences between these results and those obtained using nearest neighbor interpolation.