CSE 166: Image Processing, Fall 2017 – Assignment 1

Instructor: Ben Ochoa

Due: Wednesday, October 11, 2017, 11:59 PM

Instructions

- Review the academic integrity and collaboration policies on the course website.
- 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 containing your solutions and results.
- Your report will be a pdf file named `CSE_166_hw1_lastname_studentid.pdf`, where `lastname` is your last name and `studentid` is your student ID number.
- All of your MATLAB source code must be included as a listing in the appendix of your report.
- Submit your report on Gradescope.
- Additionally, you must create a zip file named `CSE_166_hw1_lastname_studentid.zip`, where `lastname` and `studentid` is your last name and student ID number, respectively. This zip file will contain the pdf file and a directory named `code` that contains all of your MATLAB source code.
- Submit your completed assignment by email to `rkollipa@eng.ucsd.edu` **and** `asrikant@ucsd.edu`. The subject of the email message must be `CSE 166 Assignment 1`. Attach the zip file to the message.
- It is highly recommended that you begin working on this assignment early to ensure that you have sufficient time to correctly implement the algorithms and prepare a report.

Problems

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

   Given the 2D transformation matrices

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

   show that $\mathtt{H} = \mathtt{H}_\mathrm{t}^{-1}\mathtt{H}_\mathrm{R}\mathtt{H}_\mathrm{t}$ 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.