Homework 1, Computer Vision Fall 2018

Prof. Dr. Francesco Maurelli, Jacobs University Bremen

Handed out 19.09.2018, due 26.09.2018 23:59:59

Please use the moodle system to upload your homework (`moodle.jacobs-university.de`). The system will shut down at the deadline and homeworks will not be accepted at a later time. No other form of submission is allowed.
Please upload one single zip file, with a main pdf file which can guide the instructors through your work and the various Matlab files and images generated. Be as clear as possible.
Use of LaTeXis recommended, though not compulsory. Scans of hand-written papers are accepted as long as the writing is clearly understandable. Add your source code and result images when requested and properly reference them in the pdf file.

1 Image Manipulation

Remember, you can use the MATLAB `doc` *function to learn how to use a command (e.g.* `doc rgb2gray`) *. Also, the problems below can be solved without writing a single for loop, if you use matrix math in your MATLAB code (a.k.a. "vectorization").*

(a) The provided `imageManip.m` script reads in the provided `u2dark.png` photo and converts it to grayscale using rgb2gray. Use the script and your own code to calculate the following statistics: What is the average pixel value of the resulting grayscale image? What are the min and max values? There are several ways to calculate these quantities in MATLAB. Review the MATLAB tutorial if you need help.

(b) We would like to bring the image to a more typical average brightness. Add an offset and apply a scaling factor to all pixels, so that the minimum pixel value becomes 0 and the max pixel value becomes 255. (Cameras often do a similar function automatically.) Include the original image in your report, as well as the MATLAB code you used to produce it.

(c) Next, we would like to double the contrast of the pixels in the middle brightness range. Specifically, take your result from part (b) and replace each pixel's intensity i with a new intensity $i^{'}$, where

$$i^{'} = 2 * (i - 128) + 128$$

Threshold $i^{'}$ so that $0 \leq i0 \leq 255$ (you can use the `uint8` function). Include your MATLAB code and the resulting contrast-boosted image in your report. Compare the image to part (b). What was the downside of increasing contrast in this way, and why did it happen?

2 Edge Detection

An "edge" is a place where image intensity changes abruptly. Edges can indicate the borders of objects.

(a) The intensity changes associated with vertical edges can be detected by calculating:

$$horizontalgradientatapixel \approx (thepixel) - (pixelonitsleft)$$

for every pixel in the image. Open `edgedetector.m` and edit the `DetectVerticalEdges()` function to do this. Run the `edgedetector()` function to display your calculated gradients as an image. Compare the original image to the image of gradients. Verify that the vertical edges were detected and are visible as very bright or very dark areas in the gradient image. However, the gradient image also shows some tiny bright/dark spots that indicate "edges" in the water. What caused these tiny "edges?" Include your explanation and your gradient image in your report.

(b) We will now assess the effects of "blurring" the gradient image. The simplest blurring operation is the box blur. In a box blur of width n, each blurred pixel is computed as follows:

$$blur(x, y) = \frac{1}{n^2} \sum_{i=0}^{n-1} \sum_{j=0}^{n-1} img(x + i; y + j)$$

In `edgedetector.m`, edit the `BoxBlur()` function to do this, and run the `edgedetector()` function to view the results. What was the effect of the blur on the large edges, compared to the tiny "noise" edges? Why? Include your explanation and your blurred edge image in your report.

3 Transformation Matrices

Note: In Problem above, you had to handle the fact that MATLAB images don't use Cartesian coordinates. You don't need to think about that detail for this problem; just use standard Cartesian coordinates.

Recall that transformation matrices can be used to rotate, scale, and/or skew vectors. For example, the rotation matrix

$$A = \begin{bmatrix} \sqrt{2}/2 & -\sqrt{2}/2 \\ \sqrt{2}/2 & \sqrt{2}/2 \end{bmatrix}$$

will rotate a vector 45 degrees counterclockwise (CCW). Points may be thought of as vectors from the origin, so we could equivalently say that it will rotate a point by 45 degrees CCW about the origin. For a vector

$$X_0 = \begin{bmatrix} x_0 \\ y_0 \end{bmatrix}$$

(a) Please describe (in words) the effect of the transformation

$$X_1 = AAX_0$$

In other words, how does X_1 relate to X_0? *Hint: when considering a series of matrix transformations, it is best to think of them as being applied one after the other, moving from right to left.*

(b) Design a series of matrix multiplications which will first scale a vector X_0 by 1.0 in the x direction and 0.5 in the y direction, and then rotate the result by 45 degrees counterclockwise, to produce a vector which we will call X_2. Give a formula for X_2 in terms of X_0, the matrix A from above, and any other necessary matrices which you define.

(c) Use matrix multiplication to produce a single 2x2 matrix T which will accomplish the rotating-and- scaling transformation from the point above. In other words, give a 2x2 matrix T such that $X_2 = TX_0$, with X_2 as defined above. You may use Matlab to multiply matrices. The MATLAB script `imageTransform.m` is provided with the homework. Open it and add your transformation matrix where the comments indicate. Run the script to see MATLAB apply your transformation matrix to every pixel in the image. Include the transformed image in your report, as well as your matrix T. *If the output is not what you expect, double-check your work in the previous points.*

(d) *We will use 2D homogeneous coordinates $[x; y; 1]^T$ in this problem.* Build a transformation matrix T to translate a vector by +2 in the x direction (and not perform any rotation). Also build a rotation matrix R to rotate a vector 90 degrees CCW. Include T and R in your report. Given the 2D homogeneous vector

$$p_0 = \begin{bmatrix} 7 \\ 0 \\ 1 \end{bmatrix}$$

use MATLAB to apply the transformation $p_1 = TRp_0$ and $p_2 = RTp_0$. Give the values of p_1 and p_2 and explain why they are the same or why they differ from each other.

4 SVD for Image Compression

Singular Value Decomposition (SVD) can be effectively used to compress images. Suppose I is the pixel intensity matrix of a large image nxn. The transmission (or storage) of I requires $O(n^2)$ numbers. Instead, one could use I_k, that is, the top k singular values σ_1; σ_2; ... ; σ_k along with the left and right singular vectors u_1; u_2; ... ; u_k and v_1; v_2; ... ; v_k. This would require using $O(kn)$ real numbers instead of $O(n^2)$ real numbers. If k is much smaller than n, this results in substantial savings. In this problem, you will explore SVD compression on the `flower.bmp` image we have provided. In addition to your answers to each question, you should also submit your Matlab code and required plots where necessary. *Hint: You may find the Matlab `svd` command particularly useful for this problem.*

(a) Use MATLAB to read in `flower.bmp` and convert it to grayscale and 'double' format. Apply SVD and give the top 10 singular values. Generate a plot for all singular values versus their rankings (the `diag` command may be helpful to format the values). What do you notice from this plot?

(b) Verify that you can reconstruct and display the image using the three SVD matrices (note that the `svd` command returns V , not V^T). Then, perform compression by using only the top k singular values and their corresponding left / right singular vectors. Let k = 10, 50, and 100. Reconstruct and print the compressed images for the three different values of k. Briefly describe what you observe.

(c) Instead of transmitting the original (grayscale) image, you can perform SVD compression on it and transmit only the top k singular values and the corresponding left / right singular vectors. This should be much smaller than the original image for low values of k. With this specific image, will we still save space by compressing when k = 200? Show why or why not.