EECS 442 Computer Vision: HW2

Term: Fall 2017
Instructor: Jason J. Corso, EECS, University of Michigan
Due Date: 10/05 23:59 Eastern Time

Constraints: This assignment may be discussed with other students in the course but must be written independently. Over-the-shoulder Matlab coding is strictly prohibited. **Web/Google-searching for background material is not permitted. Everything you need to solve these equations is presented in the course notes and background materials, which have been provided already.**

Goals: Deepen understanding of domain operations, geometric invariance and local feature point case study.

Data: You need to download the supplemental data to complete this assignment. All paths in the assignment assume the data is off the local directory. The download has data and code; it is named `hw2.zip`.

Problem 1 (16): Domain Operations

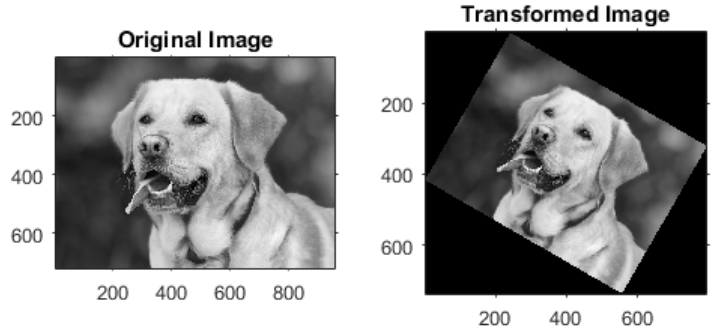
Recall domain operations and geometric transformations discussed in the lecture.

1. (2) Why do we use homogeneous coordinates?
2. (4) In the 2D rigid transformation matrix
$$\begin{pmatrix} \mathbf{R} & \mathbf{t} \\ \mathbf{0} & 1 \end{pmatrix},$$
rotation is centered at the origin. Sometimes we need to rotate a point around another point instead of the origin. Assume that there are two points $o = (3, 5)$ and $p = (4, 2)$ in the 2D plane. Write down the transformation matrix T that rotates p around o by 45^o clockwise.

   Hint: T could be written as the multiplication of several matrices. You can just leave it in that form.
3. (10) Implement a similarity transform in Matlab:
$$T = \begin{pmatrix} \frac{\sqrt{3}}{3} & -\frac{1}{3} & 30 \\ \frac{1}{3} & \frac{\sqrt{3}}{3} & 40 \\ 0 & 0 & 1 \end{pmatrix}.$$
   It's recommended to read through `2D Geometric Transformation Recipe.pdf` before you start coding, especially if you are not very clear how you're going to do this. That doc includes the instructions and tips of implementation, which may save you the time to finish this task.

   Choose any image you like as the input image. For simplicity, turn the image into **gray-scale** if it is not. Report the original and the output image; include your verbatim in your pdf submission with brief comments. Submit your code together with your report.

   Below is an example output.



Problem 2 (14): 3D Structure Tensor

The Harris operator we discussed at length in lecture computes and analyzes the eigenvalues of the 2D gradient structure tensor (you will implement it below in the third problem). Let λ_1 and λ_2 denote the larger and smaller eigenvalues, respectively. We set two criteria for selection feature points (the value of λ_2 and the ratio $\frac{\lambda_1\lambda_2}{\lambda_1+\lambda_2}$) based on the analysis of the eigenvalues (e.g., two small eigenvalues indicate a mostly absent gradient, one large and one small eigenvalue indicate an edge, and two large eigenvalues indicate a corner).

Now, consider a video parameterized over $(x, y, t) \in \mathbb{R}^3$. The eigenvalues of the 3D gradient structure tensor will similarly stratify the pixels in the video into different types. Let $\{\lambda_1, \lambda_2, \lambda_3\}$ denote the three eigenvalues of this 3D structure tensor (sorted in decreasing order).

Complete the following questions.

1. (10) Derive the form of the 3D structure tensor from the SSD error function (for window W):

$$E(u, v, w) = \sum_{u,v,w \in W} [\mathbf{I}(x + u, y + v, t + w) - \mathbf{I}(x, y, t)]^2$$

   Propose a criterion to extract "3D corners."

2. (4) Answer the question "What is a 3D corner?". What is an example of a physical phenomenon that may give rise to three large eigenvalues? Remember this is space-time and not 3D space, such as that you would image with MRI, for example.

Problem 3 (30): Local Image Features and Image Stitching

We motivated and derived an initial local image feature point detector based on an eigendecomposition of the structure tensor. We also demonstrated an application example of image stitching. In this question, you will explore these topics further.

1. (10) Implement the local structure tensor construction and eigendecomposition as described in class. This method is called Harris, but implement the simplified corner response measure based on the minimum eigenvalue of the structure tensor, as discussed in class. The source file to which you should add code is `harris.m`. It includes a comment block indicating where you should fill in your code. The function should return `C` where every pixel contains the corner strength. This is a "corner response image."

   To test your code, you should run `run_3_1.m`, which will load a simple checkerboard image and execute your `harris` function on it. This will generate two images in figures and save them to disk. `response_checkerboard.png` is the checkerboard response image (directly from harris) and `detect_checkerboard.png` is the actual corner detections. The detect script will run a non-maximal suppression routine and extract corner locations from the response image.

   Include two images as well as your code verbatim in the pdf report. Submit the `harris.m` together with the report.

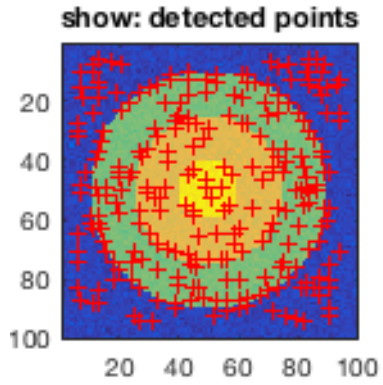
   Note that a `j_harris.p` file has been included. This is my working solution code. You can us it for testing and evaluations. You may not use it for generating the images for the assignment. We will know because we will run your code.

2. (5) Run the `run_3_2.m` script; it will load an image of concentric rings and generate a corner response image `response_rings.png`. Please discuss the response image explaining why it looks like it does. Specifically describe why there are so many responses yet there are no "corners" in the image, and why, even though, there are many responses, no full ring has a high corner response over the entire ring. Include the image and your response in the writeup.

   You may use the `j_harris.p` in place of your `harris.m` to answer this question, but you are encouraged to use yours.

3. (5) Run the `run_3_3.m` script. It will load the same concentric rings and generate the image that is on the right. It is a mess. There are very many false positive corners. Figure out a way, **any way**, that will improve this result to get corners only on the right boundaries at most. You can add any code you want to `run_3_3.m`; you may change the code that is there. You may not call Image Processing Toolbox functions for detecting corners; you must use our detectors, but you can change/add to the other parts of the code. Remove the false positives so that you only have detected corners on the ring boundaries. Include your new `run_3_3.m` (as '.m' and text in the report) and the best `detect_rings.png` that you can generate.

   You may use the `j_harris.p` in place of your `harris.m` to answer this question, but you are encouraged to use yours.



4. (10) **Image Stitching**. This question will show you an end-to-end use of these computer vision tools by automatically stitching together the left two images below to generate the stitched one on the right. No human intervention is needed. To make this possible a lot of existing code has been provided to you; your `harris.m` is all we additionally need to accomplish this task. See for yourself: run `run_red.m`. Amazing, right?

   The code for doing this has the three basic pieces: reduction (extraction of corners and representation with HOG features), matching (finding the best `K` correspondences across the images), and estimation (computation of the homography that aligns the two image). You can walk through the code to get a better sense of these steps; ask questions if you have them.

   Run `run_red_varyk.m`, which will run the whole process three times but vary the number of correspondences that are selected. In each run, it will generate some images `red_showcorrespondences_K.png` and `red_stitched1_K.png` where `K` is the number of correspondences (4, 10 or 20). The first image shows the extracted correspondences and the second shows the stitching. To fit the homography, at least 4 correspondences are needed, which is why we use 4. But, 4 seems to yield an inaccurate stitching. 10 is much better. However, when we go to 20, it fails completely.

   Explain (1) why 4 correspondences is worse than 10; (2) why 20 correspondences, which we may expect to be better, completely fails.

   Include the six images and your explanations in the report.

   You should use your `harris.m` to answer this question.

Image 1 Image 2 Automatically Stitched Image

Submission Process: NOTE: this may change before the submission. I am considering an alternative process. Submit a single pdf with your answers to these problems, include the code verbatim as text in the pdf (it is only a few lines of Matlab or Python). Include all plots and discussion in the pdf. Submit code together with the pdf if specified in the problems.

Grading and Evaluation: The credit for each problem in this set is given in parentheses at the stated question (sub-question fraction of points is also given at the sub-questions). Partial credit will be given for both paper and Matlab questions. For Matlab questions, if the code does not run, then limited or no credit will be given.