CSC 249/449 Machine Vision: Homework 1

Term: Spring 2018
Instructor: Dr. Chenliang Xu
Due Date: March 6, 2018 12:30 Eastern Time
Version: 1.1 (February 8, 2018)

Constraints: This assignment may be discussed with other students in the class but must be written independently. Over-the-shoulder Matlab coding is strictly prohibited. Web/Google-searching for background material is permitted, but web/google-searching for specific answers to the questions is prohibited (the lectures and readings provide nearly all of the information needed to answer these questions).

Problem 1 (total 20 points): Vanishing Points and Vanishing Lines.

The first two components are FP Problems 1.2 and 1.3.

Vanishing Lines. (5 points) First prove geometrically that the projection of two parallel lines lying in some plane Φ appear to converge on a horizon line h formed by the intersection of the image plan Π with the plane parallel to Φ and passing through the pinhole.

(10 points) Next prove this result algebraically using the perspective projection formulas. Assume Φ is orthogonal to Π for simplicity.

Vanishing Points (5 points) can be used for a variety of tasks in computer vision, such as camera self-calibration, and single view metrology. Consider the case in an image that you extracted a pair of points p_1 and q_1 as well as a second pair of points p_2 and q_2; these four points are in the image plane. Their counterparts in the world, P_1, Q_1, P_2, Q_2, are coplanar. Furthermore, the line formed by P_1 and Q_1 is parallel to the line formed by P_2 and Q_2. The line formed by p_1 and q_1 may or may not be parallel to the line formed by p_2 and q_2 (note, this does not change the question at all). Derive a closed form equation for the vanishing point induced by these two parallel lines.

Problem 2 (total 20 points): Linear Basis of Images of a Lambertian Surface

Consider the image $\mathbf{I}$ of a Lambertion surface $\mathbf{S}$. The surface is described smoothly by its normal vectors $\mathbf{S}(p)$ for each point p on the surface and $\mathbf{S}(p) \in \mathbb{R}^3$. And, assume we can directly index the image intensity as a point p, i.e., we have:

$$\mathbf{I}(p) = \mathbf{S}(p)^T \mathbf{v} \ , \tag{1}$$

where $\mathbf{v}$ is the direction and intensity of a directional light source (point-light source at infinity) with parallel rays, and we have assumed surface albedo of 1. Finally, assume that we can capture an arbitrary number of images for specific known directional light source, $\mathbf{v}_i$; in other words, $\mathbf{I}_i(p) = \mathbf{S}(p)^T \mathbf{v}_i$.

1. **(5 points)** Formulate (do not solve) the least-squares problem that would let you recover the surface's normal vectors. Be sure to clearly describe the definitions/specification of each term in the least squares set up.
2. **(5 points)** At least how many pairs of images and directional light sources are needed to solve this problem?
3. **(5 points)** Explain how $\mathbf{S}$ is a linear basis for all images of this surface.
4. **(5 points)** What limitations would you have for general surfaces? List at least two.

Problem 3 (total 60 points): Spot-It 1: Matching

This problem will get you working in MATLAB and implementing the first of three computer vision systems that can play the game called Spot-It. This will be an interesting exercise.

Spot-It: the game has a set of playing cards with a small number of pictures on each. The pictures are simple drawings/icons such as a bottle, a whale, etc. The set of pictures are the same cross the whole deck of cards except that they have been translated, rotated and scaled. The cards are organized such that on each pair of cards there is one and only one matching picture identity (differing by translation, scale and rotation).

Data and code: In this version of the assignment, we have scanned in the cards and segmented them for you. In the data downloaded for the assignment, there is a folder named `cards` containing the images and a script `oneload.m` that will load in all of the images into a cell array `cards` (**run this first!**). The function `oneshow.m` will visualize most of the parts of this assignment (depending on the arguments); read the code. Also, grading will be done on Linux MATLAB; use `fullfile` for any local file expansion.

Work: You will implement a basic local feature extraction and description (reduction), and a feature matching across images.

1. **(15 points)** Implement the missing body of code in `harris.m`, which should implement basic 2D harris point extraction. Read the documentation in the code to understand what information is given to your function. You can run `onedetect.m` to execute this code over an image and then visualize the output with `oneshow.m` as follows: `X = onedetect(1); oneshow(1,X);`.
2. **(20 points)** Implement the histograms of oriented gradients descriptor, including the rotational alignment, within function `hog.m`. For reference, we have included the Lower IJCV 2004 paper; the rotational alignment is in Section 5 and the HOG descriptor is in Section 6. However, you should implement it to compute the HOG descriptor separately in each color channel and then concatenate all three color channel descriptors together. You can run `onereduce.m` to detect and describe features on a single image: `F = onereduce(1);`. *Note that the detection and localization in the Lowe paper uses a different measure than the Harris corner we are using. Their measure also adapts to the local scale of the point by analyzing DoG extrema in scale-space, which will perform better.*
3. **(10 points)** Implement feature matching in `onematchsingle.m`, which has one feature vector and one feature matrix (from a different image). The matching function should output the index of the feature match (the column in the matrix) and its cost (use Euclidean distance, closer to 0 distance means lower cost). Note that `onematch.m` will take two feature matches, compute a full set of matches and then rank order them. You can visualize your output with `oneshow(1,X1,2,X2,M)` where `X1` and `X2` are the feature point locations from `onedetect.m` and `M` is the match outputs from `onematch.m`.
4. **(15 points)** Run `oneexample.m` to run through a full example. In a text file named "explain.txt" (less than 2000 characters) explain what you see in the matches of your method. Are they sufficient to play the game? Are they bad? Why might they be missing obvious matches? Note that oneexample creates "lastexample.png", which should be included in the zip and match up with the discussion.

Extra Credit (20 points) The matches using the above algorithm do nothing to enforce the scale invariance that is necessary in this game. For an extra 20 points implement the DoG scale-invariant operator and localization (Sections 3 and 4) in the Lowe paper. These should be implemented in a file named `dog.m` and then you can call the `onedetect.m` with the method as "`dog`" and it will use this detector instead of harris.

Submission Process:

Bring the solutions to problems 1 and 2 on paper to class on the due date. Before the class, zip up your MATLAB code and update to the submission folder in the Blackboard assignments. The zip of your folder should expand to one-level up from the main files: the download of the files expanded to `one.m/xyz.m`, and your implementation should do the same.

Be sure to include all of the files, we will run your code. The MATLAB code needs to run on our machines. Do not include any local paths or local files. If any are used, then be sure to upload them with the other code (none are expected). Do not change output formats/locations of the scripts as these may be auto-graded by our system.

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 complete, then limited or no credit will be given.