Assignment 1

CS283, Computer Vision
Harvard University

Due Friday, Sep. 15, at 5:00pm

The purpose of this assignment is to introduce you to Matlab as a tool for manipulating images, and to exercise your new knowledge of model-fitting and the two-dimensional projective plane. There is a "Hints and Information" section at the end of this document that is likely to help you a lot. This will be a common feature in future assignments.

The input image files that are required to complete this assignment can be found in the `./data` folder of the ZIP archive that contained this PDF file. In future assignments, there will often also be a `./src` folder that contains associated code.

As described in the "CS283 Assignment Submission Guidelines", you will submit your work using the submission system on the course website. It is important that your submission is formatted in adherence to the guidelines. Submissions that deviate from these guidelines risk not being graded. Remember that the online submission system closes *exactly* at the deadline.

Following Hartley & Zisserman, the notation for Question 5 is such that $\mathbf{x}$ and $\tilde{\mathbf{x}}$ indicate homogeneous and inhomogeneous vectors, respectively.

1. *(10 points)* In the assignment's `./data` folder there is a color image called `baboon.tif`. This image appears frequently in the image processing literature.

   (a) Write a sequence of Matlab commands that loads the image and reports its height and width in pixels. (See help for functions `imread`, `size`, and `fprintf`.)

   (b) Write a sequence of Matlab commands that converts this image to a grayscale image and displays it. In addition, display three other grayscale images that correspond to each of the three separate RGB color components. To do this you will need to understand the way Matlab represents RGB images and how to decompose them. Use the `subplot` command to display the four results in a single row. (See help for functions `imshow` and `rgb2gray`.)

   (c) You can use the `imwrite` command to write an image to a file in various formats with varying levels of compression. Write code that creates a new JPEG version of the image with a quality setting of 95 to the file `baboon_compressed.jpg`, and then reads and displays this new image. Can you tell the difference between the compresses image and the original?

   (d) The compression ratio is the ratio between the size of the original file (in bytes) and the size of the compressed file (in bytes). What is the compression ratio? (Do not submit code for this part of the question.)

   (e) By changing the JPEG quality settings, determine the lowest setting for which the compressed image is indistinguishable from the original. What is the compression ratio? (Do not submit code for this part of the question.)

2. *(10 points)* The `./data` folder contains another popular image, `cameraman.tif`.

   (a) Write code that uses a random number generator to select 25% of the pixels and replace their gray-level values with independent, random integers uniformly distributed between 0 and 255. Display the result. (See functions `rand` and `randperm`.)

   (b) At 25%, is the picture still recognizable? Explore the question for other percentages and determine if there is a well defined threshold or if there is gradual degradation. (Do not submit code for this part of the question.)

3. *(30 points)* Given a set of points $\mathbf{p}_i = (x_i, y_i)$ in the image plane, we wish to find the best line passing through those points.

(a) One way to solve this problem is to find (a, b) that most closely satisfy the equations $y_i = ax_i + b$, in a least-squares sense. Write these equations in the form $\mathbf{Ax} = \mathbf{b}$ where $\mathbf{x} = (a,\ b)^\mathsf{T}$, and give an expression for the least-squares estimate of $\mathbf{x}$. Give a geometrical interpretation of the error being minimized. Does this make sense when fitting a line to points in an image?

(b) Another way to solve the problem is to find $\boldsymbol{\ell} = (a,\ b,\ c)$ (defined up to scale) that most closely satisfies the equations $ax_i + by_i + c = 0$, in a least-squares sense. Write these equations in the form $\mathbf{A}\boldsymbol{\ell} = \mathbf{0}$ where $\boldsymbol{\ell} = (a,\ b,\ c)^\mathsf{T}$ and $\boldsymbol{\ell} \neq \mathbf{0}$. This problem has a trivial solution (zero vector $\mathbf{0}$), which is not of much use. Describe one way to avoid this trivial solution, and describe an algorithm for solving the resulting optimization problem. Provide a geometrical interpretation of the error being minimized. Is this approach more or less useful than the approach of (a)? Why?

(c) Write code that loads the image `dots.tif` (from the `./data` folder, as usual) and : i) detects the red, green, and blue points and obtain their (x, y) image coordinates; ii) uses the approach of part (b) to fit lines to each of these three point sets; and iii) plots these lines, superimposed on the image. (The functions `ind2sub` and `find` may be useful.)

(d) These lines do not intersect at a point, but we can find a point that comes 'closest' to lying at their common intersection. For this, one can use a procedure that is analogous to part (b): formulate and solve a homogeneous linear system of the form $\mathbf{Ax} = \mathbf{0}$, with appropriate constraints on the solution $\mathbf{x}$. Describe one way to do this, and interpret your approach geometrically. Does it make sense? Write code that computes the common intersection point using your method and displays the result, superimposed on the image.

4. *(20 points)* In the presence of outliers, we require more robust techniques for model fitting. RANSAC is one such method that is both useful and conceptually simple.

(a) Write code that loads the image `dots_outliers.tif` and uses the approach of Question 3(b) to fit and draw a "best" line. Note that, in this case, the `.tif` image is black-and-white, with the points being white (you may need to zoom-in to see the points clearly.)

(b) Write code that uses RANSAC to improve the fit and draw a better line. We suggest that the number of iterations be 100 and the threshold used to determine the inlying set for each iteration be a distance of 20 pixels. Your code snippet should begin by assigning these values to global variables:

```
num_iter = 100;          % number of RANSAC iterations
inlier_threshold = 20;   % threshold for inlier set of each line
<<more code here>>
```

Display your results with the lines from both (a) and (b).

5. *(10 points)* Consider a right triangle with vertices $\tilde{\mathbf{x}}_1 = (0,\ 0)$, $\tilde{\mathbf{x}}_2 = (m,\ 0)$ and $\tilde{\mathbf{x}}_3 = (0,\ m)$, and suppose this triangle is warped by an affine transformation such that

$$\begin{pmatrix} x' \\ y' \\ 1 \end{pmatrix} = \begin{bmatrix} a_{11} & a_{12} & t_x \\ a_{21} & a_{22} & t_y \\ 0 & 0 & 1 \end{bmatrix} \begin{pmatrix} x \\ y \\ 1 \end{pmatrix},$$

or $\mathbf{x}' = \mathbf{Ax}$. Derive an expression for the area of the warped triangle defined by $\tilde{\mathbf{x}}'_1$, $\tilde{\mathbf{x}}'_2$ and $\tilde{\mathbf{x}}'_3$. Use this expression to prove that if two right triangles (with $m = m_1$ and $m = m_2$) are warped by the same affine transformation, the ratio of their areas is preserved.

Hints and Information

- To get help on a particular function in Matlab, type `help <function>`. To get a list of functions related to a particular keyword, use the `lookfor` function. We will be making extensive use of the Image Processing Toolbox, and a list of the functions in that toolbox is generated by typing `help images`. When showing results in your live script, save space and improve readability by using `subplot` whenever possible. As an example, the following Matlab script loads three images and displays them in a single figure.

```
% read three images from current directory
im1=imread('image1.tif');
im2=imread('image2.tif');
im3=imread('image3.tif');

% display images in a figure, side-by-side
figure;           % create a new figure
subplot(1,3,1);   % break the figure up into a 1x3 grid of axes
                  % and make the top-left axis active
imshow(im1);      % display an image
title('Image 1'); % add a title

% repeat for images 2 and 3
subplot(1,3,2); imshow(im2); title('Image 2');
subplot(1,3,3); imshow(im3); title('Image 3');
```

- A linear least-squares problem is one in which we want to determine the vector $\mathbf{x}$ that best satisfies a set of inconsistent linear constraints $\mathbf{Ax}=\mathbf{b}$ in the sense of minimum square error. That is, we wish to solve
$$\mathbf{x}^* = \arg\min_{\mathbf{x}} ||\mathbf{Ax}-\mathbf{b}||^2 = \arg\min_{\mathbf{x}} (\mathbf{Ax}-\mathbf{b})^\top(\mathbf{Ax}-\mathbf{b}).$$
The solution is found in closed-form by differentiating the objective function with respect to $\mathbf{x}$ and equating the result to zero. This yields
$$\mathbf{x}^* = (\mathbf{A}^\top\mathbf{A})^{-1}\mathbf{A}^\top\mathbf{b}.$$
In the above expression, the inverse is used only for notational purpose. The explicit calculation of $(\mathbf{A}^\top\mathbf{A})^{-1}$ is *very bad practice*, because finding the inverse is both very expensive and numerically unstable.[1] Instead, it is better to use a method such as QR factorization to solve the (consistent) linear system $(\mathbf{A}^\top\mathbf{A})\mathbf{x} = \mathbf{A}^\top\mathbf{b}$, sometimes called the *normal equation* of the original linear system. Matlab has its own operator for solving linear least squares problems in such a way:

```
x = A\b;
```

- Based on the "CS283 Assignment Submission Guidelines", your submission should have the following file structure:

```
.zip
 ├─ .mlx ..................................................................Main live script file.
 ├─ src/ .................External m-files that are required by the live script (none for Assignment 1).
 └─ data/ .......Image and other data files for the live script, such as cameraman.tif and baboon.tif.
```

[1] http://blogs.mathworks.com/loren/2007/05/16/purpose-of-inv/