CS4670 Spring 2016

HW2

Due: May 6, 2016, noon
Homeworks to be done alone. Remember the academic integrity policies.
Show your work!
Start early and complete problems as we cover them in class.
Consult piazza for clarifications on the HW2 FAQ.

1 Single View Modeling

(a) **Horizon Line**

Figure 1 shows a square $PQRS$ defined by 2-D points on the image plane $P = (0, 0)$, $Q = (1, -0.2)$, $R = (0, -1)$, and $S = (-1, -0.3)$ viewed under perspective projection. In 3-D, the square is lying on the ground plane (where $z = 0$), and the line PQ is parallel to the y-axis.

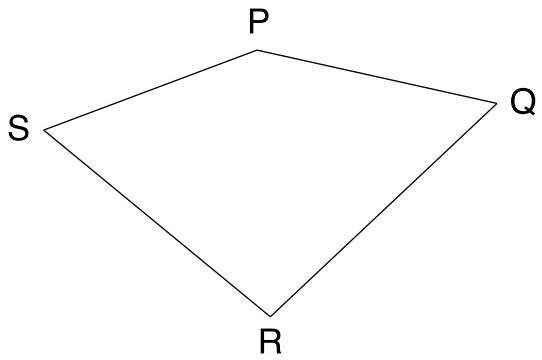


Figure 1: Square $PQRS$

(i) What are the two vanishing points v_x and v_y? Show your work.

(ii) What is the vanishing line (horizon line) defined by these two vanishing points? Show your work.

(b) **Measuring Height.** Figure 2 shows a camera image of a square $PQRS$ from the previous part, with a vertical pillar rb standing on the ground plane. The 2-D coordinates in the image are: $r = (0.2, 0.4), b = (0.2, -0.6)$.

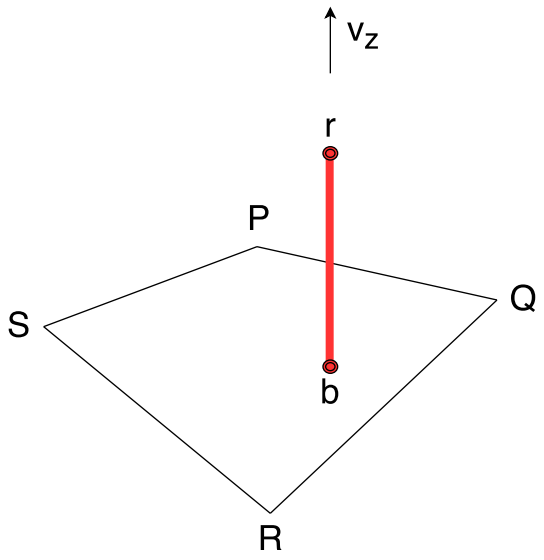


Figure 2: Square $PQRS$ with pillar rb

The points r and b define the vertical pillar with point b lying on the ground plane. (The 3-D z-coordinate $b_z = 0$). Let $(r_x, r_y, r_z) = (0.76, 0.89, 5)$ be the 3-D coordinates of the point r. Given that you know the vanishing point v_z is at infinity, compute the height of the camera.

(i) Obtain a point t that is co-linear with r and b that has the same height as the camera. Show your derivations.

(ii) Use the image cross ratio to obtain H, the height of the camera. Show your derivation, equations, and work.

(c) **Finding Position**

The point $g = (-0.2, 0.35)$ is now introduced to the image. The point g has the same z coordinate as the point r.

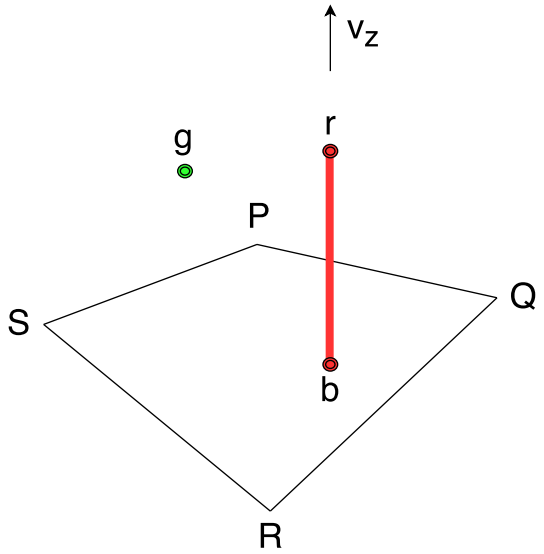


Figure 3: Square $PQRS$ with pillar rb and point g

(i) Obtain an equation for the homogeneous image coordinates of the point f lying directly under g on the ground plane.

(ii) Compute the homography H that transforms 3-D points lying on the ground plane into homogeneous image coordinates. Assume that the 3-D coordinates of the points are the following (they are all on the ground plane): $P_{3D} = (0, 0, 0)$, $Q_{3D} = (0, 1, 0)$, $R_{3D} = (1, 1, 0)$, $S_{3D} = (1, 0, 0)$.

(iii) Given the 3-D coordinates of $r = (r_x, r_y, r_z) = (0.76, 0.89, 5)$, compute the 3-D coordinates of g.

2 Stereo: Normalized Eight-Point Algorithm

Write Python/numpy code for implementing the **Normalized 8-point algorithm**. Break your code into two functions:

1. `hartley_normalize`, which implements Hartley normalization for the input points. This consists of the following steps:

   (a) Translate the points so that their centroid is at the origin (in non-homogeneous coordinates).

(b) Scale the coordinates of the points so that the average distance from the origin is $\sqrt{2}$.

(c) Compute this transformation for the two input point groups (corresponding to the two images) independently to obtain T and T'.

2. `compute_fundamental_matrix`, which calls `hartley_normalize` and computes the fundamental matrix F given point correspondences $\{p_i \leftrightarrow p_i'\}$. As you learned in class, $p_i'^T F p_i = 0$ should hold.

(a) Using the `hartley_normalize` function, you can obtain T and T' and normalize the input points to get $\hat{p_i} = Tp_i$ and $\hat{p_i}' = T'p_i'$.

(b) Build a linear equation system to solve for $\hat{F}$, where each equation is given by $\hat{p_i}'^T \hat{F} \hat{p_i} = 0$. This will give you the equation $Af = 0$, where f is a vector with 9 elements made up of the entries of $\hat{F}$.

(c) To solve $Af = 0$, you should take the right singular vector corresponding to the smallest singular value of A.

(d) To enforce that $\hat{F}$ is rank-2, compute the SVD of $\hat{F} = USV^T$. Then keep the two largest singular values and set the smallest one to 0, to get S'. $\hat{F}' = US'V^T$ is guaranteed to be maximum rank-2 and minimizes the Frobenius norm of $\hat{F} - \hat{F}'$, i.e. the two matrices are as close as possible.

(e) To arrive at the final fundamental matrix for the original points, you have to reverse the normalization in the following way: $F = T'^T \hat{F}' T$.

You have to fill out the functions in `eightpoint_student.py`. We provided 8 point correspondences and some initial tests in `eightpoint_tests.py`. You can run the tests with the command `nosetests`, similarly to pa4. If you don't have `nosetests`, please run `sudo apt-get install python-nose`.

You can use the following Numpy functions: `dot`, `max`, `mean`, `min`, `reshape`, `sqrt`, `sum`, `svd`, `transpose`, vectorization, all constructors (like np.zeros) and operators. No other functions are allowed. Please include `eightpoint_student.py` with your submission. Do not modify the signature of `compute_fundamental_matrix` or `hartley_normalize`!

3 Optical Illusion

In this problem you have to create an optical illusion. The scene configuration is as follows. There is a camera (center of projection) at $(0, 0.8, 0)$, the floor with normal vector $(0, 1, 0)$, going though the origin, a billboard in vertical position on the floor with points

$A = (-0.3, 0, 1), B = (0.3, 0, 1), C = (-0.3, 0.4, 1), D = (0.3, 0.4, 1)$. Your task is to compute the image which should be drawn on the floor to provide the same picture as the vertical billboard from the viewpoint of the camera. There are examples for optical illusion art on the internet (link). See an example below (4b and 4c):

(a) Use this image as input

(b) Image splatted on the billboard

(c) Transformed image, which should be put on the floor.

1. Compute the projection of the corners of the billboard to the floor using the camera as the center of projection. Write down the coordinates of the projected points in your answer.

2. Now imagine that we tip down the billboard on the floor, but we keep the two points which were already on the floor at the same position. Write down the 2-D coordinates of the points of the tipped down billboard and the projected billboard in the plane of the floor. The floor plane coordinate system's x axis is the same as the original 3-D coordinate system and the y axis is the same as the original 3-D coordinate system's z axis, but it is 0 at points A, B.
   Compute the homography to transform the points of the billboard to the projected billboard points in the plane of the floor using the 4-point algorithm discussed in lecture (link to lecture). You can use your implementation from pa3. Write down the normalized homography in your answer (the bottom-right element of the matrix should be 1).

3. Use the computed homography to transform the provided image (4a) and print it. You first need to transform from image coordinates to billboard coordinates (you computed these coordinates in part 2), let's call this transformation T. Then apply your computed homography and finally transform back to image coordinates using T^{-1}. You can use OpenCV's perspectiveTransform function to compute the corners

of the transformed image and warpPerspective for warping the image itself.

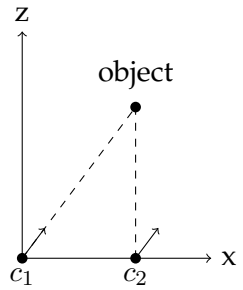
4. If you print the transformed photo, put the paper on a table and shut one of your eyes and move to the right position, you should see the optical illusion! You can also use your phone to take a picture of the paper from the right position to check. Include the original and the transformed image to showcase the illusion with your submission.

4 Stereo: Triangulation

There are two cameras taking pictures of an object, as depicted in the figure below. We want to compute the location of the object.

- Camera 1 has focal length 3 and Camera 2 has focal length 4. Assume no other intrinsics are in play.
- Camera 1 has center position $c_1 = (0, 0, 0)$ and Camera 2 has center position $c_2 = (5, 0, 0)$. The arrows leaving c_1 and c_2 in the figure show the directions in which the cameras are pointed.
- Both Camera 1 and Camera 2 have 4×4 rotation matrix R, given below. Note that R takes orientations in world coordinates to orientations in camera coordinates. Remember that, in camera coordinates, the $-z$ axis is the direction in which the camera points.

$$R = \begin{pmatrix} -0.6 & 0 & 0.8 & 0 \\ 0 & 1 & 0 & 0 \\ -0.8 & 0 & -0.6 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$



1. The image coordinates of the object on the image from Camera 1 are $(0, 0)$. Using these image coordinates, find the vector pointing from the center position of Camera 1 to the point on the image plane corresponding to the object, with respect to the camera's 3-D coordinate system.
2. Find the vector pointing from the world origin $(0, 0, 0)$ to the point on the image plane corresponding of the object, with respect to the world's coordinate system.

3. Using the vector obtained in (part 2) and Camera 1's center position, find the equation of the line through Camera 1's center and the object, with respect to the world's coordinate system. Provide an equation in parametric form that's defined for non-homogeneous 3-D points.
4. The image coordinates of the object on the image from Camera 2 are $(3, 0)$. Using the steps in parts (part 1) through (part 3), find the equation of the line through Camera 2's center and the object.
5. These lines intersect at the location of the object. Compute the object's location.

5 Photometric stereo

In this exercise, you will solve the photometric stereo problem with unknown directional lighting. You can assume that all light sources have the same intensity. There are 6 light sources and 3 points whose normal vector you have to determine. You can use the Lambertian shading model and assume that all three points have positive albedo k_{d1}, k_{d2}, k_{d3}. The image measurement matrix M is the following:

$$M = \begin{bmatrix} 0.10000 & 0.00000 & 0.00000 \\ 0.00000 & 0.20000 & 0.00000 \\ 0.00000 & 0.00000 & 0.30000 \\ 0.05774 & 0.11547 & 0.17321 \\ 0.08165 & 0.08165 & 0.12247 \\ 0.05345 & 0.16036 & 0.08018 \end{bmatrix}$$

In each part of the exercise, show your reasoning! You are encouraged to use numpy/scipy to compute SVD and solve linear equations. Please include your code in your submitted document. For more detailed explanation, see the original paper: link.

(a) Formulate the problem you must solve in photometric stereo. Clearly specify all matrices and their dimensions and the corresponding relevant equation(s).

(b) Compute the singular value decomposition of $M = U\Sigma V^T$ and show the estimated pseudo light source $\hat{L} = U\sqrt{\Sigma}$ and surface $\hat{S} = \sqrt{\Sigma}V^T$ matrices, where $M = \hat{L}\hat{S}$.

(c) Our final goal is to resolve the ambiguity by finding A in the equation $M = \hat{L}AA^{-1}\hat{S}$. To apply the constraints for the light sources, you have to solve the system of equations $l_i^T B l_i = 1$ for all l_i light source vectors, where $B = AA^T$ is a symmetric matrix as explained in the lecture. Write down this system as a linear matrix equation where

the unknown coefficients of B are stacked in the parameter vector p_B. The equation will take the shape: $X_{design}p_B = 1$. Write down X_{design}.

(d) Solve the previous equation (you can use np.linalg.lstsq here) for B.

(e) Compute A, where A is the invertible matrix we want to find in $M = \hat{L}AA^{-1}\hat{S}$, given $B = AA^T$. You can use SVD again as suggested in the linked paper.

(f) Write down the final light-source and surface matrices.

(g) Write down the normal vectors and albedo of the three points.

(h) Is it possible that your solution is not the same as the original light and normal matrices which gave the measurement matrix M? If yes, why?

6 Overfitting: True/False

We are performing object classification and are training a deep convolutional neural network (CNN). The network has convolutional layers followed by fully connected layers, with ReLU in-between every layer. The last layer is softmax, and the loss is cross-entropy. There is no max pooling. We use mini-batch stochastic gradient descent with momentum.

For our CNN, we always train the network until the training loss converges and stops improving (we look at a moving average to avoid any noise in the training loss). Once we are done, we test on the validation set and discover that our CNN has overfit the training set (the training accuracy is much higher than the validation accuracy).

This will help reduce the amount of overfitting:

1. **Removing** the biases from every layer
2. **Clipping** the weights in a layer to a maximum L2 norm (each iteration)
3. **Adding** dropout near the beginning of the network
4. **Decreasing** the weight on L2 regularization
5. **Increasing** the number of filters in a convolutional layer
6. Performing data augmentation by randomly **cropping** sub-regions of the input image each pass through the dataset
7. **Decreasing** the number of channels (hidden units) in a fully connected layer
8. **Adding** another convolutional layer in the middle

9. Replacing a **convolutional** layer with a **locally connected** layer (same size and number of filters)
10. Replacing a **fully connected** layer with a 3x3 **convolutional** layer (where the input to the layer has shape 9x9x9, and the number of filters is chosen so that the total number of values in the output remains the same)