Homework 3 Written Questions

CSCI 1430

Template Instructions

This document is a template with specific answer regions and a fixed number of pages. Given large class sizes and limited TA time, the template helps the course staff to grade efficiently and still focus on the content of your submissions. Please help us in this task:

- Make this document anonymous.
- Questions are in the orange boxes. Provide answers in the green boxes.
- Use the footer to check for correct page alignment.
- **Do NOT remove the answer box.**
- **Do NOT change the size of the answer box.**
- **Extra pages are not permitted unless otherwise specified.**
- **Template edits or page misalignment will lead to a 10 point deduction.**

Gradescope Submission

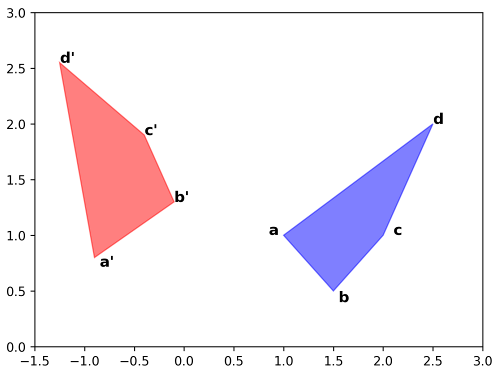
- Compile this document to a PDF and submit it to Gradescope.
- Pages will be automatically assigned to the right questions on Gradescope.

This Homework

- 6 questions **[10 + 8 + 4 + 10 + 3 + 12 = 47 points + 2 bonus points]**.
- Include code, images, and equations where appropriate.

Q1 — [10 points]

Suppose we have a quadrilateral $\mathbf{abcd}$ and a transformed version $\mathbf{a'b'c'd'}$:



$\mathbf{a} = (1, 1)$	$\mathbf{a'} = (-0.9, 0.8)$
$\mathbf{b} = (1.5, 0.5)$	$\mathbf{b'} = (-0.1, 1.3)$
$\mathbf{c} = (2, 1)$	$\mathbf{c'} = (-0.4, 1.9)$
$\mathbf{d} = (2.5, 2)$	$\mathbf{d'} = (-1.25, 2.55)$

Let's assume that each point in $\mathbf{abcd}$ was mapped to its corresponding point in $\mathbf{a'b'c'd'}$ by a 2×2 transformation matrix $\mathbf{M}$.

e.g., if $\mathbf{p} = \begin{bmatrix} x \\ y \end{bmatrix}$, $\mathbf{p'} = \begin{bmatrix} x' \\ y' \end{bmatrix}$, and $\mathbf{M} = \begin{bmatrix} m_{1,1} & m_{1,2} \\ m_{2,1} & m_{2,2} \end{bmatrix}$,

then $\begin{bmatrix} m_{1,1} & m_{1,2} \\ m_{2,1} & m_{2,2} \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} x' \\ y' \end{bmatrix}$

We would like to estimate $\mathbf{M}$ using least squares for linear regression.

Q1.1 — [1 point]

Rewrite the equation $\mathbf{Mp} = \mathbf{p'}$ into a pair of linear equations by expanding the matrix multiplication.

TODO: Replace each of the '__' below with x, y, x', y', or 0.

$$\begin{cases} \_\_m_{1,1} + \_\_m_{1,2} + \_\_m_{2,1} + \_\_m_{2,2} = \_\_ \\ \_\_m_{1,1} + \_\_m_{1,2} + \_\_m_{2,1} + \_\_m_{2,2} = \_\_ \end{cases}$$

Q1.2 — [2 points]

To estimate $\mathbf{M}$, as it has four coefficients, we need four equations. As our quadrilaterals have four points, we could create eight such equations (two for each point) that use the transformation $\mathbf{M}$.

From these equations for each $\mathbf{p}, \mathbf{p}'$ correspondence, we can construct a matrix $\mathbf{A}$ and column vector $\mathbf{b}$ with a vector of coefficients of $\mathbf{M}$—let's call that $\mathbf{x}$—that must satisfy:

$$\mathbf{Ax} = \mathbf{b} \quad \text{where} \quad \mathbf{A} \times \begin{bmatrix} m_{1,1} \\ m_{1,2} \\ m_{2,1} \\ m_{2,2} \end{bmatrix} = \mathbf{b}$$

Note: As systems of linear equations are typically written in the form $\mathbf{Ax} = \mathbf{b}$, we've overloaded the symbol $\mathbf{b}$ here. So, be careful—$\mathbf{b}$ here is the vector of target x', y' values across all equations, and not the point in the quadrilateral.

Declare $\mathbf{A}$ and $\mathbf{b}$:

Replace each '__' below with a 0 or a coordinate value from the quadrilaterals.

$$\begin{bmatrix} \_\_ & \_\_ & \_\_ & \_\_ \\ \_\_ & \_\_ & \_\_ & \_\_ \\ \_\_ & \_\_ & \_\_ & \_\_ \\ \_\_ & \_\_ & \_\_ & \_\_ \\ \_\_ & \_\_ & \_\_ & \_\_ \\ \_\_ & \_\_ & \_\_ & \_\_ \\ \_\_ & \_\_ & \_\_ & \_\_ \\ \_\_ & \_\_ & \_\_ & \_\_ \end{bmatrix} \times \begin{bmatrix} m_{1,1} \\ m_{1,2} \\ m_{2,1} \\ m_{2,2} \end{bmatrix} = \begin{bmatrix} \_\_ \\ \_\_ \\ \_\_ \\ \_\_ \\ \_\_ \\ \_\_ \\ \_\_ \\ \_\_ \end{bmatrix}$$

Q1.3 — [1 point]
If we use more than four equations, then our problem is over-constrained. In this case, we want to find values for $m_{i,j}$ that minimize the squared error between the transformed values for $\mathbf{p}'$ and the real known $\mathbf{p}'$ values, i.e., we want to minimize $||\mathbf{Ax} - \mathbf{b}||^2$.

To do this, we can use the singular value decomposition to find the pseudoinverse of $\mathbf{A}$, written as $\mathbf{A}^\dagger$. Then, we multiply it by both sides, giving us:

$$\begin{aligned}\mathbf{A}^\dagger\mathbf{Ax} &= \mathbf{A}^\dagger\mathbf{b}\\ \mathbf{x} &= \mathbf{A}^\dagger\mathbf{b}.\end{aligned}$$

In linear algebra class, you might have learned how to solve linear systems by hand, but thankfully the computer can do this for us even for large systems! `numpy.linalg.lstsq()` takes in our $\mathbf{A}$ matrix and $\mathbf{b}$ vector, and returns $\mathbf{x}$.

Note: We provide a Python script `transformation_viz.py` for you to estimate and visualize the transformation. Complete matrix $\mathbf{A}$ and vector $\mathbf{b}$ to see how the quadrilaterals match up!

State your estimated $\mathbf{M}$ by replacing each of the '__' below:

$$\mathbf{M} = \begin{bmatrix} m_{1,1} & m_{1,2} \\ m_{2,1} & m_{2,2} \end{bmatrix} = \begin{bmatrix} \_\_ & \_\_ \\ \_\_ & \_\_ \end{bmatrix}$$

Q1.4 — [6 points]
Note that the least squares regression function that we use here is an approximation function, meaning that the values we derived for the transformation matrix are not exact. The same is true of our 8 point algorithm method of calculating the fundamental matrix.

Determine what kind of transformation it is by forming a system of linear equations and determining a matrix $\mathbf{M}$ that produces zero residual error (to machine numerical precision). Feel free to try your implementation in `transformation_viz.py` to see how the resulting quadrilateral matches the ground truth.

Write out your system's $\mathbf{A}$ and $\mathbf{b}$ matrices as in (b), state $\mathbf{M}$ and the residual, and state which kind of transformation it is. **[4-6 sentences]**

TODO: your answer for (d) here

Q2: [8 points] In lecture, you've learned that cameras can be represented by intrinsic and extrinsic matrices. These matrices can be used to calculate the projections of points within a 3D world onto 2D image planes. For this, we use *homogeneous coordinates*. The final 3×4 matrix is known as the *camera matrix*.

Recall that the transformation can be represented by the following expression:

$$\begin{bmatrix} f_x & s & 0 \\ 0 & f_y & 0 \\ 0 & 0 & 1 \end{bmatrix} \times \begin{bmatrix} r_{11} & r_{12} & r_{13} & t_x \\ r_{21} & r_{22} & r_{23} & t_y \\ r_{31} & r_{32} & r_{33} & t_z \end{bmatrix} \times \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} = w \begin{bmatrix} u \\ v \\ 1 \end{bmatrix}$$

where f is the focal point, r is the rotation matrix, t is the translation vector, w is some weighing/scaling factor, and (u, v) is the position of the point in the real world (x, y, z) projected on the 2D plane.

(a) **[2 points]** For each of the following, you are given the camera specifications and a sample 3D point from the real world.

Fill in the camera's intrinsic and extrinsic matrices; then, perform the multiplications and perspective division (unhomogenize) to find the 2D coordinate of the projected point on the image.

(i) A camera with a focal length of 1 in both the x and y directions, a translation of 5 along the x-axis, and no skew or rotation.

TODO: Fill in the __ entries

$$M_{\text{intrinsic}} \times M_{\text{extrinsic}} \times \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

$$= \begin{bmatrix} \_\_ & \_\_ & 0 \\ 0 & \_\_ & 0 \\ 0 & 0 & 1 \end{bmatrix} \times \begin{bmatrix} \_\_ & \_\_ & \_\_ & \_\_ \\ \_\_ & \_\_ & \_\_ & \_\_ \\ \_\_ & \_\_ & \_\_ & \_\_ \end{bmatrix} \times \begin{bmatrix} 30 \\ -20 \\ 10 \\ 1 \end{bmatrix}$$

$$= \begin{bmatrix} \_\_ \\ \_\_ \\ \_\_ \end{bmatrix}$$

$$= \_\_ \times \begin{bmatrix} \_\_ \\ \_\_ \\ 1 \end{bmatrix}$$

(ii) A camera with focal length of 2 in both the x and y directions, a translation of 5 along the x-axis, and no skew or rotation.

TODO: Fill in the __ entries

$$
\begin{aligned}
&= \begin{bmatrix} \_\_ & \_\_ & 0 \\ 0 & \_\_ & 0 \\ 0 & 0 & 1 \end{bmatrix} \times \begin{bmatrix} \_\_ & \_\_ & \_\_ & \_\_ \\ \_\_ & \_\_ & \_\_ & \_\_ \\ \_\_ & \_\_ & \_\_ & \_\_ \end{bmatrix} \times \begin{bmatrix} 30 \\ \text{-}20 \\ 10 \\ 1 \end{bmatrix} \\
&= \begin{bmatrix} \_\_ \\ \_\_ \\ \_\_ \end{bmatrix} \\
&= \_\_ \times \begin{bmatrix} \_\_ \\ \_\_ \\ 1 \end{bmatrix}
\end{aligned}
$$

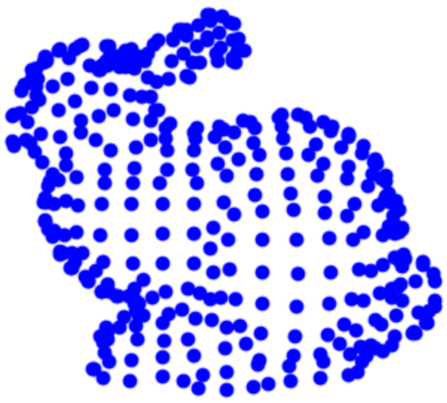
(b) **[2 points]**

Compare the two image coordinates you've calculated in parts a and b. Explain how each parameter affects the final image coordinate. **[2-3 sentences]**

TODO: Your answer to (b) here.

(c) **[1 + 3 points]** In the questions folder, we've provided stencil code for a camera simulation in `camera_simulation.py`. Given a camera matrix, the simulator visualizes an image that a camera would produce.

Please implement `calculate_camera_matrix()` by calculating the camera matrix using the parameters given in the code (see stencil for more detail). When successful, you will see a bunny rendered as dots (see below). Paste your code for this function and attach a screenshot of the working demo once you finish. Play around with the sliders to see how different parameters affect the projection!



```python
import numpy as np
import matplotlib.pyplot as plt
from mpl_toolkits import mplot3d
from matplotlib.widgets import Slider, Button

# Initial random matrices
initial_intrinsic_matrix_to_replace = np.random.rand(3,3)
initial_extrinsic_matrix_to_replace = np.random.rand(3,4)
initial_camera_matrix_to_replace = np.random.rand(3,4)

# Setting up the point cloud
file_data_path= "./images/bunny.xyz"
point_cloud = np.loadtxt(file_data_path, skiprows=0,
                         max_rows=1000000)
# center it
point_cloud -= np.mean(point_cloud,axis=0)
# homogenize
point_cloud = np.concatenate((point_cloud, np.ones((
                         point_cloud.shape[0], 1))),
                         axis=1)
# move it in front of the camera
point_cloud += np.array([0,0,-0.15,0])

def calculate_camera_matrix(tx, ty, tz, alpha, beta, gamma,
                            fx, fy, skew, u, v):
    """
    This function should calculate the camera matrix using
                            the given
    intrinsic and extrinsic camera parameters.
    We recommend starting with calculating the intrinsic
                            matrix (refer to lecture
                            8).
    Then calculate the rotational 3x3 matrix by calculating
                             each axis separately and
    multiply them together.
    Finally multiply the intrinsic and extrinsic matrices
                            to obtain the camera
                            matrix.
    :params tx, ty, tz: Camera translation from origin
    :param alpha, beta, gamma: rotation about the x, y, and
                             z axes respectively
    :param fx, fy: focal length of camera
    :param skew: camera's skew
    :param u, v: image center coordinates
    :return: [3 x 4] NumPy array of the camera matrix, [3 x
                             4] NumPy array of the
                            intrinsic matrix, [3 x 4]
                             NumPy array of the
                            extrinsic matrix
    """
    ########################
    # TODO: Your code here #
    # Hint: Calculate the rotation matrices for the x, y,
                            and z axes separately.
    # Then multiply them to get the rotational part of the
                            extrinsic matrix.
    ########################
    return (initial_camera_matrix_to_replace,
    initial_intrinsic_matrix_to_replace,
    initial_extrinsic_matrix_to_replace)

def find_coords(camera_matrix):
    """
    This function calculates the coordinates given the
                            student's calculated
                            camera matrix.
    Normalizes the coordinates.
    Already implemented.
    """
    coords = np.matmul(camera_matrix, point_cloud.T)
    return coords / coords[2]
```

Only Q2.3 should be on this page

```
################################################
# YOU MAY USE THIS ADDITIONAL PAGE

# WARNING: IF YOU DON'T END UP USING THIS PAGE
# KEEP THESE COMMENTS TO MAINTAIN PAGE ALIGNMENT
################################################
```

Q3: **[4 points]** Given a stereo pair of cameras:

(a) **[2 points]**

Briefly describe triangulation. Describe the inputs and outputs of the process. You may wish to use a diagram. **[3-4 sentences]**

TODO: Your answer to (a) here.

(b) **[2 points]**

Why is it not possible to find an absolute depth for each point when we don't have calibration information for our cameras? Note that absolute depth refers to depth with respect to the camera as opposed to relative depth, which is with respect to another object in the scene. **[3 - 4 sentences]**

TODO: Your answer to (b) here.

Q4: [10 points] Given the algorithms that we've learned in computer vision, we know that whether we can find/calculate the essential matrix, the fundamental matrix, or both depends on the setup of the cameras and images. You are given three datasets of an object of unknown geometry:

(i) A video circling the object;

(ii) A stereo pair of calibrated cameras capturing two images of the object; and

(iii) Two images of the same object on the internet (e.g. Colosseum) at different camera poses but with unknown intrinsics.

(a) **[3 $\times$ 1 points]**

For each of the above setups, what calculations can we perform? Provide brief explanations to support your choice(s). **[1 - 2 sentences]**

(i) Setup 1

☐ Essential Matrix
☐ Fundamental Matrix
☐ Both

(ii) Setup 2

☐ Essential Matrix
☐ Fundamental Matrix
☐ Both

(iii) Setup 3

☐ Essential Matrix
☐ Fundamental Matrix
☐ Both

(b) **[3 $\times$ 1 points]**

State an advantage and disadvantage of using each setup for depth reconstruction. **[2 - 3 sentences]**

(i) Setup 1

TODO: Your answer to (b) (i) here.

(ii) Setup 2

TODO: Your answer to (b) (ii) here.

(iii) Setup 3

TODO: Your answer to (b) (iii) here.

(c) **[4 points]** The differences between the collection methods for these three datasets are crucial in terms of what calculations are possible - and therein which applications they are most useful in.

From a non-technical standpoint, can you think of a scenario why you may prefer one of these data collection setups to another? Why is it important to know what data collection methods have been used to build a particular dataset? **[5-7 sentences]**

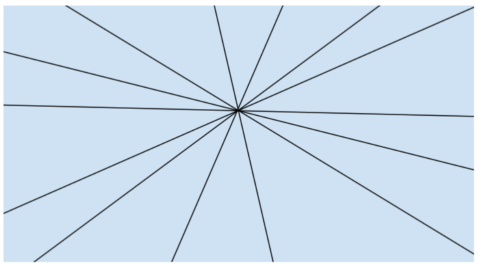
TODO: Your answer to (d) here.

Q5: **[3 points]** In two-view camera geometry, what do the following epipolar lines say about the cameras' relative positions? **[1 - 2 sentences each]**

Tip: The Spring '22 course staff created an interactive demo to explore the different scenarios and get a better feel for epipolar geometry.

(a) Please draw or describe the relative positions of the two camera planes. Check slides from the lecture on stereo geometry for details of carera planes. **[1 point]**

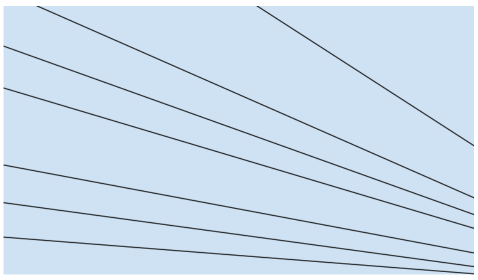
Radiate out of a point on the image plane.



TODO: Your answer to (a) here.

(b) Also draw or describe the relative positions of the two camera planes. Check slides from the lecture on stereo geometry for details of carera planes.**[1 point]**

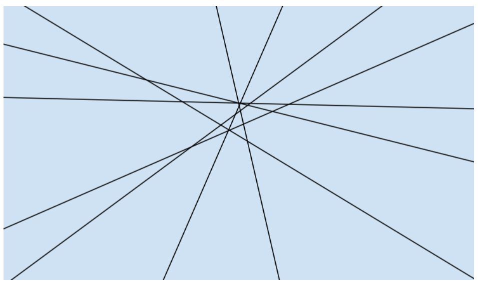
Converge to a point outside of the image plane.



TODO: Your answer to (b) here.

(c) **[1 point]**

Notice the misalignment of the epipolar lines in the image below? What went wrong in the calculation of the fundamental matrix and how can we fix it?
Hint: Check slides from the lecture on stereo geometry.



TODO: Your answer to (c) here.

Q6a: **[6 points]** Cameras are used in surveillance systems. One argument in favor of surveillance systems is to deter and solve crime to improve safety. Another is that if you're not doing anything wrong, you don't have anything to worry about. One argument against surveillance systems is that they compromise people's privacy even when no wrongdoing is taking place. Another is that they increase stress and anxiety.

Computer vision allows the *automation* of surveillance. For instance, it lets us find the mathematical relationship between multiple cameras to track objects and people in 3D spaces, or it can reduce the burden upon a human operator who need only respond to detected events rather than actively monitor many cameras. Such functionality makes it easier to scale a surveillance operation.

On Brown's campus, the number of surveillance cameras has been increasing: compare this 2008 Brown Daily Herald article with this 2020 Brown Daily Herald article. While some, like those in Hegeman Hall, were installed only temporarily (2021 Brown Daily Herald article), there are now 800 surveillance cameras on campus.

Suppose Brown both did and did not use computer vision automation. How comfortable are you with Brown's surveillance apparatus in each case? In what circumstances do you believe that the potential benefits of surveillance *automation* outweigh the potential concerns, and why? [8–10 sentences]

TODO: Your answer here.

Q6b: [6 points] Unmanned aerial vehicles—sometimes called drones—often carry cameras. Their cameras can be used for navigation via manually remote control, or for use within sophisticated computer vision strategies like camera pose estimation and depth estimation to enable assisted or autonomous flying in complex environments.

For your CSCI 1430 final project, you are developing a drone for life-saving organ delivery. You create a successful computer vision algorithm that allows your drone to navigate autonomously. You are approached by several organizations that want to pay you generously for access to your project, but you are also considering open sourcing your algorithm with a permissive software license.

Please list three organizations that might be interested in acquiring your project for their own purposes. If each of these organizations used your project, who could benefit and how? Who could be harmed and how? Would an open-source license affect this? [6–9 sentences]

TODO: Your answer here.

Feedback? (Optional)

We appreciate your feedback on how to improve the course. You can provide anonymous feedback through this form, that can be accessed using your Brown account (your identity will not be collected). If you have urgent non-anonymous comments/questions, please email the instructor.