Homework 3 Questions

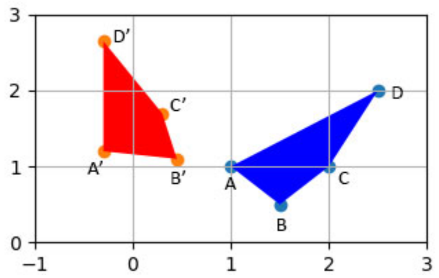
Document Instructions

- 5 questions **[10 + 8 + 4 + 13 + 3 = 40 points + 2 bonus points]**.
- Fill all your answers within the answer boxes, and **please do NOT remove the answer box outlines**.
- Questions are highlighted in the **orange boxes**, bonus questions are highlighted in **blue boxes**, answers should be recorded in the **green boxes**.
- Include code, images, and equations where appropriate.
- To identify all places where your responses are expected, search for 'TODO'.
- The answer box sizes have been set by the staff beforehand and will truncate your text if it goes beyond the limit. Please make sure your responses fit in the appropriate spaces. **Extra pages are not permitted unless otherwise specified.**
- Make sure your submission has the right number of pages to validate page alignment sanity (check the footer).
- Please make this document anonymous.

Gradescope Instructions

- When you are finished, compile this document to a PDF and submit it directly to Gradescope.
- The pages will be automatically assigned to the right questions on Gradescope *assuming you do not add any unnecessary pages*. **Inconsistently assigned pages will lead to a deduction of 2 points per misaligned page (capped at a maximum 6 point deduction).**

Q1: [10 points] Suppose we have a quadrilateral $ABCD$ and a transformed version $A'B'C'D'$ as seen in the image below.



$$\begin{aligned} A &= (1,1) & A' &= (-0.3, 1.3) \\ B &= (1.5, 0.5) & B' &= (0.5, 1.1) \\ C &= (2,1) & C' &= (0.3, 1.8) \\ D &= (2.5, 2) & D' &= (-0.3, 2.6) \end{aligned} \tag{1}$$

Let's assume that each point in $ABCD$ was approximately mapped to its corresponding point in $A'B'C'D'$ by a 2×2 transformation matrix $\mathcal{M}$.

e.g. if $X = \begin{pmatrix} x \\ y \end{pmatrix}$ and $X' = \begin{pmatrix} x' \\ y' \end{pmatrix}$, and $\mathcal{M} = \begin{pmatrix} m_{1,1} & m_{1,2} \\ m_{2,1} & m_{2,2} \end{pmatrix}$

then $\begin{pmatrix} m_{1,1} & m_{1,2} \\ m_{2,1} & m_{2,2} \end{pmatrix} \times \begin{pmatrix} x \\ y \end{pmatrix} \approx \begin{pmatrix} x' \\ y' \end{pmatrix}$

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

(a) **[1 point]**

Rewrite the equation $\mathcal{M} \times X \approx X'$ into a pair of linear equations by expanding the matrix multiplication.

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

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

(b) **[2 points]** With the quadrilaterals in question, there are 4 points that transform so we should expect to see 8 such equations (2 for each point) that use the transformation equation $\mathcal{M}$.

From these pairs of equations for each x-x' correspondence, we can construct a matrix $\mathcal{Q}$ and column vector b that satisfy

$$\mathcal{Q} \times \begin{pmatrix} m_{1,1} \\ m_{1,2} \\ m_{2,1} \\ m_{2,2} \end{pmatrix} = b$$

Note: Systems of linear equations are typically written in the form $\mathcal{A} \times x = b$, but since we have already defined $\mathcal{A}$ and x, we're writing it as $\mathcal{Q} \times m = b$

Find $\mathcal{Q}$ and b:

Replace each of the '__' below with a 0 or a coordinate value from $ABCD$ and $A'B'C'D'$.

TODO: your answer for (b) here

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

(c) **[1 point]** Our problem is now over-constrained, so we want to find values for $m_{i,j}$ that minimize the squared error between the approximated values for X' and the real X' values, i.e., we want to minimize $||\mathcal{Q} \times m - b||$.

To do this we use singular value decomposition to find the pseudoinverse of $\mathcal{Q}$, written as $\mathcal{Q}^\dagger$. We then multiply it by both sides, giving us:

$$\begin{aligned} \mathcal{Q}^\dagger \mathcal{Q} m &= \mathcal{Q}^\dagger b \\ m &\approx \mathcal{Q}^\dagger b. \end{aligned}$$

Thankfully, the computer can do all of this for us! `numpy.linalg.lstsq()` takes in our $\mathcal{Q}$ matrix and b vector, and returns approximations for m. Plug the values you wrote in part (c) into that function and write the returned $\mathcal{M}$ matrix here.

Note: You may need to reshape your output from `linalg.lstsq` to get the right dimensions.

Replace each of the '__' below with the value of $m_{i,j}$:

TODO: your answer for (c) here

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

(d) **[3 x 2 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.

Suppose there is some CV algorithm that uses an approximation such as the ones we learned about. Evaluate from the perspective of three stakeholders of your choice (e.g. the developer of the technology, the CEO of the company that develops the technology, a third party who uses your CV algorithm) how important transparency is surrounding margins of error. **[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:

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

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
>
> $$\begin{aligned} & M_{\text{intrinsic}} \times M_{\text{extrinsic}} \times \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix} \\ = & \begin{pmatrix} \_\_ & \_\_ & 0 \\ 0 & \_\_ & 0 \\ 0 & 0 & 1 \end{pmatrix} \times \begin{pmatrix} \_\_ & \_\_ & \_\_ & \_\_ \\ \_\_ & \_\_ & \_\_ & \_\_ \\ \_\_ & \_\_ & \_\_ & \_\_ \end{pmatrix} \times \begin{pmatrix} 30 \\ -20 \\ 10 \\ 1 \end{pmatrix} \\ = & \begin{pmatrix} \_\_ \\ \_\_ \\ \_\_ \end{pmatrix} \\ = & \_\_ \times \begin{pmatrix} \_\_ \\ \_\_ \\ 1 \end{pmatrix} \end{aligned}$$

(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

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

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

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

(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!


```
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= "./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 A2 (c) should be on this page

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

(a) **[2 points]**

Briefly describe triangulation (using images might be helpful).

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.

TODO: Your answer to (b) here.

Only A3 (a) - (b) should be on this page

Q4: **[13 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 we take of the object at two different camera poses (position and orientation) using the same camera but with different lens zoom settings.

(a) **[3 × 1 points]**

For each of the above setups, what calculations can we perform?

(i) Setup 1

TODO: Select the right option:

Essential Matrix	☐
Fundamental Matrix	☐
Both	☐

(ii) Setup 2

TODO: Select the right option:

Essential Matrix	☐
Fundamental Matrix	☐
Both	☐

(iii) Setup 3

TODO: Select the right option:

Essential Matrix	☐
Fundamental Matrix	☐
Both	☐

(b) **[3 × 1 points]**

State an advantage and disadvantage of using each setup for depth reconstruction

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

Name an application scenario for each of the different setups

(i) Setup 1

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

(ii) Setup 2

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

(iii) Setup 3

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

(d) **[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]**

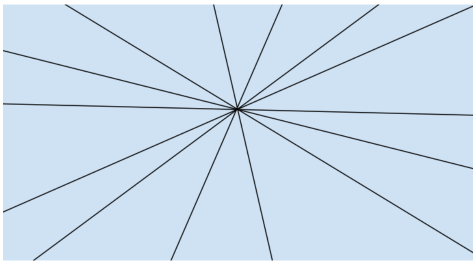
TODO: Your answer to (d) here.

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

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

(a) **[1 point]**

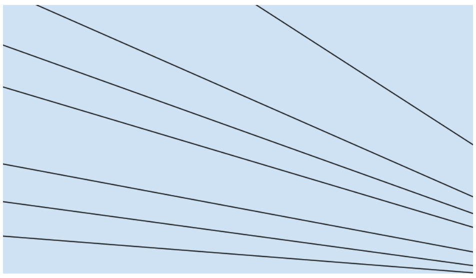
Radiate out of a point on the image plane.



TODO: Your answer to (a) here.

(b) **[1 point]**

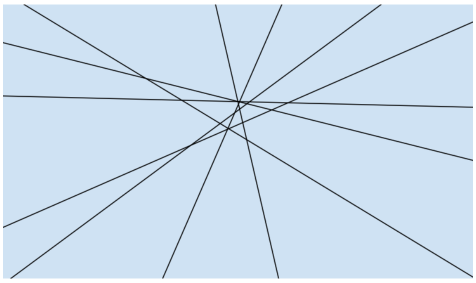
Converge to a point outside of the image plane.



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.

Discussion Attendance:

Extra Credit: [2 points]

Please mark this box only if you've attended the discussion session in person.

☐ I attended the discussion session on DATE

Feedback? (Optional)

Please help us make the course better. If you have any feedback for this assignment, we'd love to hear it!