Homework 3 Written Questions

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

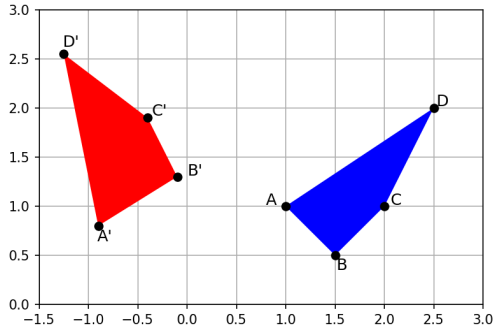
- Compile this document to a PDF and submit it to Gradescope.
- Pages will be automatically assigned to the right questions on Gradescope.

This Homework

- 4 questions **[10 + 8 + 6 + 12 = 36 points]**.
- Include code, images, and equations where appropriate.

Q1: [10 points] In 2D, suppose we have two quadrilaterals $ABCD$ and $A'B'C'D'$ as seen in the image below. They look like they might be related by a rotation and a non-uniform scale transformation.

Given the corresponding points, we will attempt to find $\boldsymbol{M}$ using linear least squares.



$$\begin{aligned} A &= (1, 1) & A' &= (-0.9, 0.8) \\ B &= (1.5, 0.5) & B' &= (-0.1, 1.3) \\ C &= (2, 1) & C' &= (-0.4, 1.9) \\ D &= (2.5, 2) & D' &= (-1.25, 2.55) \end{aligned} \tag{1}$$

Let's try and map each point in $ABCD$ to its corresponding point in $A'B'C'D'$ by a 2×2 transformation matrix $\boldsymbol{M}$, as we know this can represent rotation and scale.

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

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

(a) **[2 point]**

Rewrite $\boldsymbol{Mx} = \boldsymbol{x'}$ as a pair of linear equations by expanding the matrix multiplication. Define these algebraically with x, y, x', y', or 0.

TODO: Replace each of the '__' below.

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

(b) **[2 points]** Systems of linear equations are denoted in the form $\boldsymbol{Ax} = \boldsymbol{b}$, where the vector $\boldsymbol{x}$ holds our entries of $\boldsymbol{M}$. To estimate the four parameters of $\boldsymbol{M}$, we will need four equations in our system such that it is *determined*.

But, as quadrilaterals have four points, we could form a system of eight equations (two for each point) that is said to be *overdetermined*. In this case, it is possible to find values for $\boldsymbol{M}$ that minimize the distance (or residual error) between the approximated values for X' and the real X' values, i.e., that minimize $||\boldsymbol{Ax} - \boldsymbol{b}||_2$.

Form eight equations from the four x–x', y–y' correspondences, and construct a matrix $\boldsymbol{A}$ and column vector $\boldsymbol{b}$ that satisfy

$$\boldsymbol{A} \times \begin{pmatrix} m_{1,1} \\ m_{1,2} \\ m_{2,1} \\ m_{2,2} \end{pmatrix} = \boldsymbol{b}$$

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

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) **[2 point]** To find a solution to our system, we need to invert $\boldsymbol{A}$ even though $\boldsymbol{A}$ is non-square. We can use the pseudoinverse of $\boldsymbol{A}$, written as $\boldsymbol{A}^+$. Left multiplying it by both sides gives us:

$$\begin{aligned} \boldsymbol{A}^+\boldsymbol{A}\boldsymbol{x} &= \boldsymbol{A}^+\boldsymbol{b} \\ \boldsymbol{x} &= \boldsymbol{A}^+\boldsymbol{b}. \end{aligned}$$

As long as $\boldsymbol{A}$ has a pseudoinverse, this solution minimizes $||\boldsymbol{A}\boldsymbol{x} - \boldsymbol{b}||_2$. This is the closed-form least squares solution, where $\boldsymbol{x} = (\boldsymbol{A}^\top\boldsymbol{A})^{-1}\boldsymbol{A}^\top\boldsymbol{b}$ and where $\boldsymbol{A}^+ = (\boldsymbol{A}^\top\boldsymbol{A})^{-1}\boldsymbol{A}^\top$.

We can compute the pseudoinverse from the singular value decomposition. In python, `numpy.linalg.lstsq()` will handle this for us. It takes as input $\boldsymbol{A}$ and $\boldsymbol{b}$, and returns a solution for $\boldsymbol{x}$ along with the residual error. Plug the values you wrote in part (c) into that function and write the returned $\boldsymbol{M}$ matrix here with the residual error.

Note: You may need to reshape your output from `linalg.lstsq`.

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

Residual error: xxx

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

(d) **[4 point]** If the residual is zero (or zero to machine numerical precision), then we can confirm our initial hypothesis that the transformation is a rotation and a scale. If it is not, then we need a transformation with more degrees of freedom.

Determine what kind of transformation it is by forming a system of linear equations and determining a matrix $\boldsymbol{M}$ that produces zero residual error (to machine numerical precision).
Write out your system's $\boldsymbol{A}$ and $\boldsymbol{b}$ matrices as in (b), state $\boldsymbol{M}$ and the residual as in (c), and state which kind of transformation it is.

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

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

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

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

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

(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) **[4 points]** In the questions folder, we've provided stencil code for a camera simulation in `camera_simulation.py`. Given a camera matrix and a set of world points, 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!

TODO_demo_screenshot.png

```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]









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

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

Q3: **[6 points]** Given a stereo pair of cameras:

(a) **[3 points]**

Briefly describe triangulation. Describe the inputs and outputs of the process. You may wish to use a diagram. **[4–6 sentences]**

TODO: Your answer to (a) here.

(b) **[3 points]**

Suppose we wished to find in real-world units the depth for each scene point via triangulation—we call this 'metric depth' or 'absolute depth', rather than relative depth. What information would we need to know about our camera system? *Think about the whole process, including calibration. What units are we in?* **[3–4 sentences]**

TODO: Your answer to (b) here.

Q4a: [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.

Q4b: [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? [6–8 sentences]

TODO: Your answer here.

Feedback? (Optional)

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