

# Spring 2020 - CS 477/577 - Introduction to Computer Vision

## Assignment Four

**Due: 11:59pm (\*) Thursday, February 13.**

(\*) There is grace until 8am the next morning, as the instructor will not grade assignment before then. However, once the instructor starts grading assignments, no more assignments will be accepted.

**Weight: Approximately 5 points**

**This assignment must be done individually**

---

### General instructions

You can use any language you like for this assignment, but unless you feel strongly about it, you might consider continuing with Matlab. This assignment does not have a lot of programming.

You need to create a PDF document that tells the story of the assignment, copying into it output, code snippets, and images that are displayed when the program runs. Even if the question does not remind you to put the resulting image into the PDF, if it is flagged with (\$), you should do so. I should not need to run the program to verify that you attempted the question. See

<http://kobus.ca/teaching/assignment-instructions.pdf>

for more details about doing a good write-up. While it takes work, it is well worth getting better and more efficient at this. A substantive part of each assignment grade is reserved for exposition.

### Learning goals


- Understand image coordinate systems
- Learn about collecting data that can be used to calibrate a camera
- Learn how a camera matrix maps points from 3D to 2D
- Understand how to quantitatively evaluate camera matrices
- Determining a mapping between image coordinate systems (grads)
- Practice with translation matrices (grads)

### Assignment specification

This assignment has four parts, three of which are required for undergrads.

## Part A

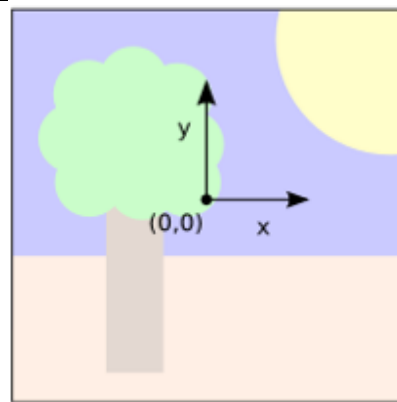
We will need coordinates of points that we choose to click. You can do this with Matlab in several ways.

For example, with `imshow()`, you can use the data cursor tool  at the middle of the row of icons above the image but below the top menu. In this question we will first make sure that we understand this “clicking” coordinate system.

Your first task for this part is provide a figure illustrating how clicked points refer to image positions. This figure, together with its caption and additional explanation should be sufficient to instruct someone on how to relate image location with these coordinates (\$). To help get you started, the canonical coordinate system, which you are already familiar with, is illustrated in Figure 1 as an example, but your answer for the clicking coordinate system should be less terse. There is room for different answers depending on your tool for clicking on points to get coordinates. The important part is to be consistent throughout this assignment and the next.

**Figure 1.** Standard camera coordinate system (Figure credit: Kyle Simek a previous TA). This is different than how image coordinate systems typically work, so we need to convert between them.

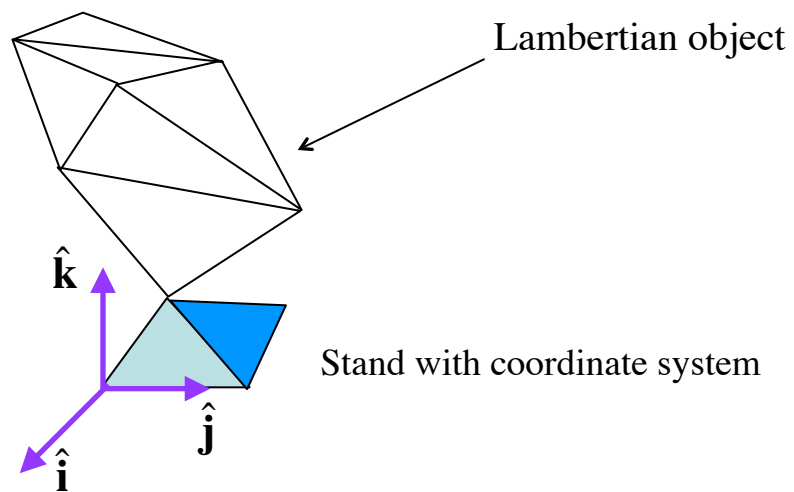
*This figure is provided as an example of illustrating the relationship between a coordinate system and an image. For the first deliverable, you are to provide a figure that illustrates the relationship of the “clicking” coordinate system and an image.*



**XY Coordinates:**

- \* origin at center of image
- \* y-axis points up

The next task is to relate these coordinates to the indices that refer to these locations by array access in the software system you are using. Are they the same (\$)? If not, explain how to convert between these two representations (\$). In particular, if you wanted to change the location of a point that is reported as (100,200) when clicking to be red (i.e., (R=255, G=0, B=0)), what is the statement for your software system (\$)? Check that this works, by providing an image with the pixel clearly illustrated. If you are having trouble seeing the red pixel, you might write a function that draws a colored box around a point of a specified size, which will likely be a useful function moving forward. Embellish your figure with axis with text labeling the index (first or second) and arrows illustrating the direction of increasing indices.



## Part B

In this part you will collect data that we will use in the next assignment to calibrate a camera from an image of a coordinate system. In part C below, you will use this data to demonstrate that you understand how the camera model works, and to evaluate two instances of them. For this part, use the following image (also in D2L/Content/Data) to collect at least 15 calibrations points.

[http://kobus.ca/teaching/cs477/data/IMG\\_0862.jpeg](http://kobus.ca/teaching/cs477/data/IMG_0862.jpeg)

To set up a world coordinate system, note that the grid lines are 1 inch apart. Also, to be consistent with the light direction given later, the X-axis should be the axis going from the center leftwards, the Y-axis is going from the center rightwards, and the Z-axis is going up. (It is a right-handed coordinate system). To help keep track of the points, I suggest using an image editing program to roughly label the points before you begin. For example, you could insert circles with labels P1, P2, etc. Put this image, or some other visual indication of exactly what your data is, in your report with a description (\$). You should also hand in two files, world\_coords.txt and image\_coords.txt which the grader can use to run your program to get your results for part C below (\$). Your image coordinates should be in our standard image coordinate system developed in the previous problem.

***Note.** Be sure that your 3D points represent all three spatial dimensions (i.e., in “general position”). Intuitively, if they were all in one plane (or mostly in one plane), we would expect to run into trouble when we use the data to calibrate the camera (next HW), as we are not collecting any information about the effect of moving around perpendicular to that plane.*

## Part C

Now consider the two camera matrices (also in D2L/Content/Data):

[http://kobus.ca/teaching/cs477/data/camera\\_matrix\\_1.txt](http://kobus.ca/teaching/cs477/data/camera_matrix_1.txt)

[http://kobus.ca/teaching/cs477/data/camera\\_matrix\\_2.txt](http://kobus.ca/teaching/cs477/data/camera_matrix_2.txt)

Use your understanding of how such a matrix models the camera by computing where they think each of your 3D points should end up on the image plane. The image coordinate system is what Matlab uses to index the points, which is not necessarily the same as the “click” coordinate system (see part A).

Create a visual representation of these points (perhaps a square of a reasonable size) onto an image that shows your data (as in the previous question). Do this for each of the two camera matrices to provide an image visualizing how each of them performs, using red for your clicked points, green for the first camera matrix, and blue for the second camera matrix (\$). Also provide the RMS error between where they points went versus where they should have gone (\$). This is often referred to as “re-projection error.” Based on these numbers and/or visual inspection, which camera matrix is more accurate (\$)?

## Part D (required for grad students only).

*(These exercises are not meant to be difficult).*

1. What is the inverse operation of a 3D translation in the direction  $(t_x, t_y, t_z)$ ? Either guess at the inverse and show that it is indeed the inverse, or derive the inverse in some other way (\$).
2. Show that translation matrices are commutative (\$).

3. Show that the translation matrix does the correct thing even to points represented by homogeneous coordinates where  $w$  is not necessarily one (\$). This question was inspired by a question in class.

### Part E (required for grad students only).

Here we develop a mapping from the standard XY coordinate system (i.e., Figure 1) to our standard image coordinate system developed in Part A above. Assume that the interval  $[0,0]$  to  $[1,0]$  in XY has length 400 pixels in the image coordinate system, and similarly the interval  $[0,0]$  to  $[0,1]$  also has length 400 pixels in the image coordinate system. Next consider that we may need some rotations/flip to map the XY axes into the image coordinate system. Finally, we want the origin in the XY coordinate system to map to  $(400, 600)$ . Derive a mapping based on various transformations discussed in class to map from the XY coordinate system to the image coordinate system. Provide each of the component matrices and the product of them for the final transformation (\$). Provide mappings for  $(-0.5, -0.5)$ ,  $(-0.5, 0.5)$ , and  $(0,1)$  (\$). Also provide the **inverse** mapping for pixel  $(1,1)$  (\$).

*Note. It is easy to get hung up on the difference between a pure coordinate and a pixel which has bounds (four corners). This does not matter for this purpose, and any choice should be fine as long as you use it consistently within your system. Two obvious choices are to associate integral coordinates with the top left corner of a pixel associating the center of the pixel with integral coordinates, which is potentially more complicated but some also think is more elegant. For the purposes of this assignment, let us adopt the first answer for consistent questions on Piazza and easier grading for the TA.*

### What to Hand In

As usual, the main deliverable will be a PDF document named hw4.pdf that tells the story of your assignment described above. Ideally the grader can focus on that document, simply checking that the code exists, and seems up to the task of producing the figures and results in the document. But you need hand in your code (e.g., hw4.m if you are working in Matlab) and the data files that you collected as well.