ICBV Exercise 1

1 Instructions

Please read and follow these instructions carefully.

- The assignment contains both programming tasks and a written part. The written part needs to be submitted in a single PDF file and the programming tasks must be answered in the provided Jupyter notebook. You need to submit a single zip file where:
  - The name of the file is "id1_id2_HW1.zip" where id1 and id2 are replaced by the id number of the submitting students.
  - In the zip file put the written answers as a PDF file and the Jupyter notebook along with any additional files used by your code.
- The written assignment must be typed on the computer. Handwritten scanned documents **will not** be accepted.
- Make sure that the Jupyter notebook can be ran. Before submitting, reset the kernel and rerun all cells. In addition, make sure you submit any additional files used by your code (Usually image files).
- Both the PDF containing the written answers and the Jupyter notebook must contain the id numbers and the names of the submitting students.

Question 1 - Parallel and perspective projections

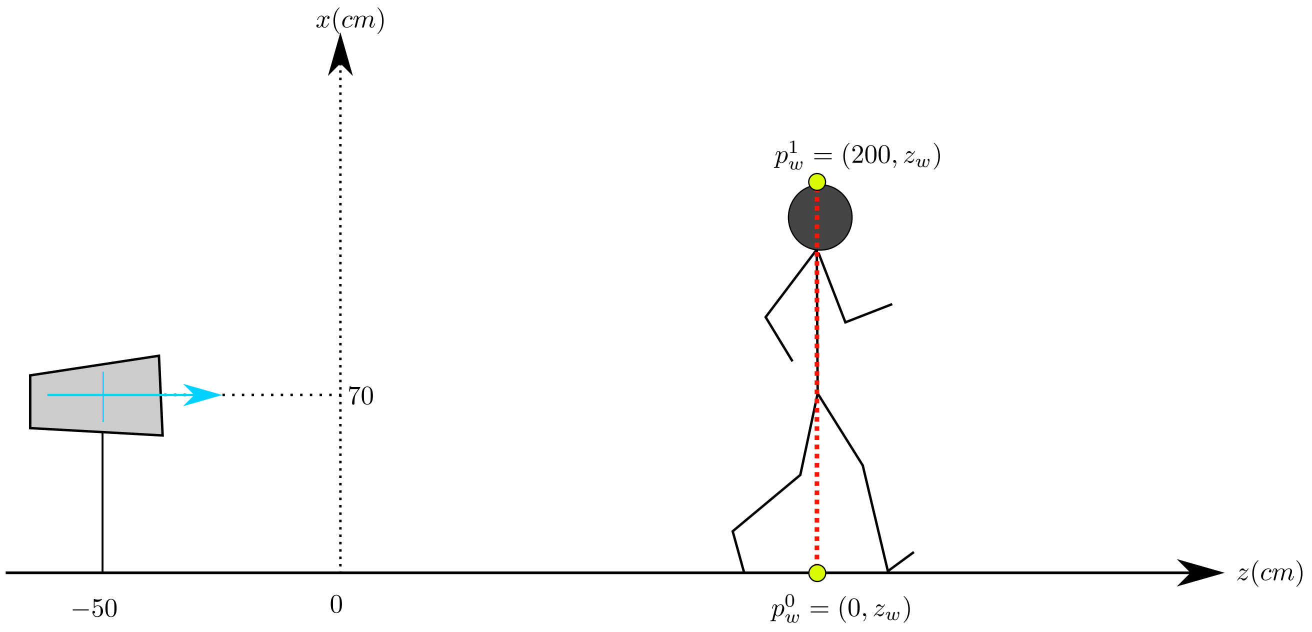


Figure 1

Avi, a student in the class, wanted to examine the effects of perspective projection. For this, he decided to conduct an experiment. He positioned a camera on a stand such that the center of the camera is at $o = (o_x, o_z) = (70cm, -50cm)$. The camera faces towards the positive direction of the z axis and at $0°$ relative to the z axis. Avi recorded himself walking along the z axis. He began at $z_w = 100cm$ until he reached $z_w = 1000cm$. Let $p_w^1 = (x_w^1, z_w^1)$ be the point at the top of Avi's head and $p_w^0 = (x_w^0, z_w^0)$ be the point at the ground below it. Assume the points remain at constant height - x_w^1 is 200 and x_w^0 is 0.

Section A

Let x_i^0 and x_i^1 be the perspective projections of $p_w^0 = (0, z_w)$ and $p_w^1 = (200, z_w)$ respectively where $z_w \in [100, 1000]$.

- Express $x_i^0(z_w)$ and $x_i^1(x_w)$.
- Plot the x_i^0 and x_i^1 as a function of z_w in the provided Jupyter notebook

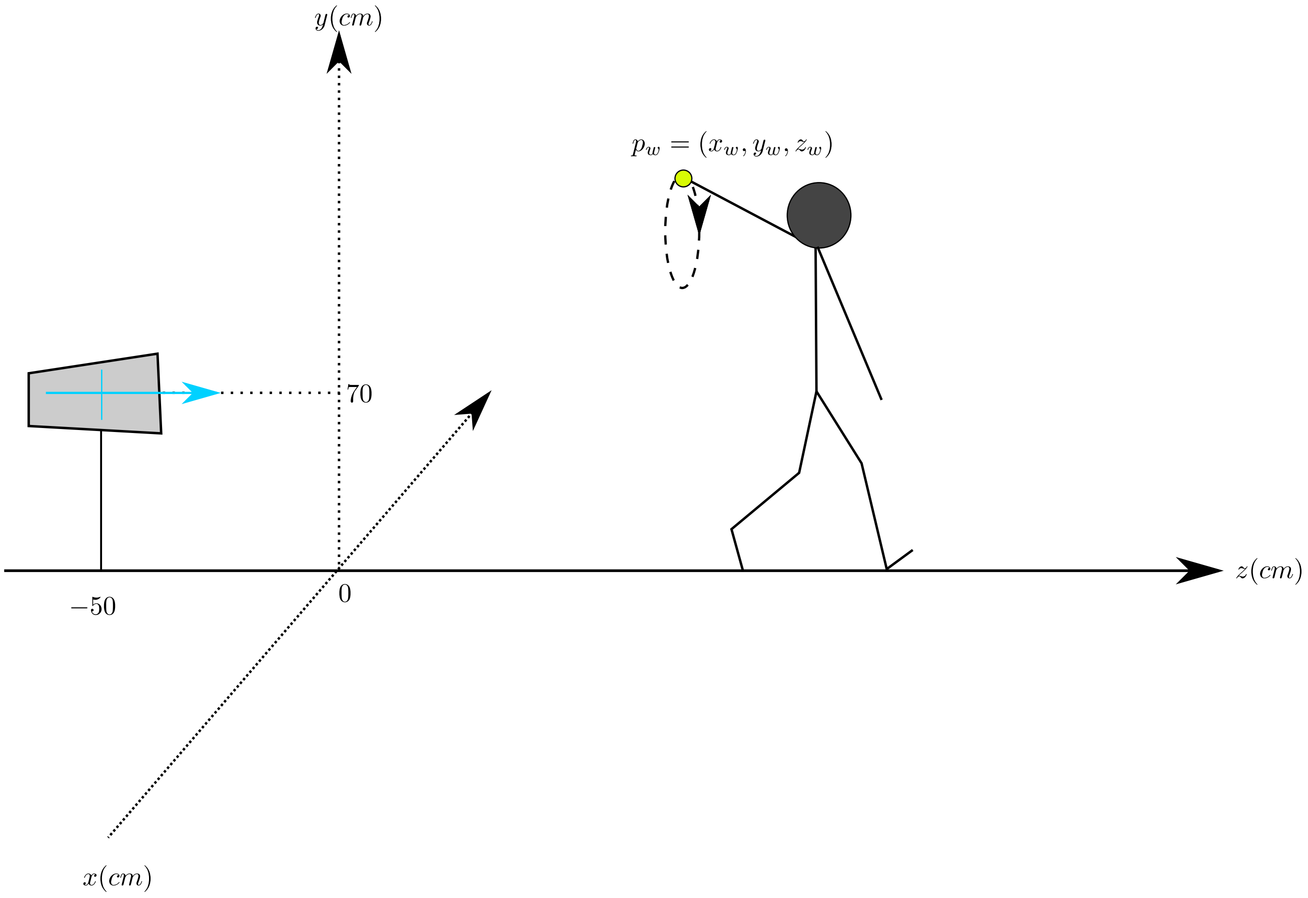
Section B

- Express $x_i^0(z_w)$ and $x_i^1(z_w)$ using a parallel projection where the projection plane is $z = 0$.
- Plot the x_i^0 and x_i^1 as a function of z_w in the provided Jupyter notebook

Section C

Deducing from Sections A and B, when is it preferable to use parallel projection, when an object is close or far from the camera? Explain **shortly**.

Section D



Avi performed another experiment. He moved his hand in circular motion with radius of $10cm$ while walking away from the camera. The palm of his hand denoted as point $p_w = (x_w, y_w, z_w)$ where the starting point is $(0, 200, 100)$. A complete circle is performed every 100cm. The following points are of the first circle:

$$(0, 200, 100), (10, 190, 125), (0, 180, 150), (-10, 190, 175), (0, 200, 200)$$

1. Express x_w, y_w as a function of z_w where $z_w \in [100, 1000]$
2. Express x_i, y_i as a function of z_w
3. Plot x_i, y_i in the provided Jupyter notebook.

Question 2 - Camera calibration

In this question, you will perform a calibration of a camera of your choice (e.g. your mobile phone camera). Your goal is to find the camera matrix using pairs of corresponding world and image points from which the camera parameters and camera matrix can be estimated.

Section A - Chessboard corners

Take 20 pictures of a chessboard and use Opencv's methods cv2.findChessboardCorners and cv2.drawChessboardCorners. Use the template code provided in the Jupyter notebook to view the results over your pictures.
Add the required code in the provided Jupyter notebook.
Hint : Finding the corners needs to be performed over a grayscale image, drawing the corners is over an RGB image.

Section B - Create matching 3d points

As shown in the practical session, create a set of 3d points matching your board dimension and square size.
Add the required code in the provided Jupyter notebook.

Section C - Perform the calibration

Perform the camera calibration using Opencv's cv2.calibrateCamera. Add the required code in the provided Jupyter notebook.
The method returns the following variables as output:

- cameraMatrix
- distCoeffs
- rvecs
- tvecs

Explain shortly what each of them mean.

Section D - Camera Matrix

You saw the following definition in class:

$$\underbrace{K \cdot F}_{M}$$

1. What element of the formula does the output parameter "cameraMatrix" refers to?
2. Obtain the extrinsic transformation matrix for each pair in (rvecs, tvecs). Write the code in the Jupyter notebook.
3. Obtain a corresponding matrix M for each of the pairs. Write the code in the Jupyter notebook.

Section E

In this section you will project the world points you defined in Section B on top of one of the images you used for the calibration process. On top of the image you chose:

- Plot the chessboard corners from section A using the opencv drawChessboardCorners method.
- Plot a projection of the world points defined in section B on top of the same image. Perform the projection using:
    1. The camera matrix you got from the calibration process
    2. The Opencv method cv2.projectPoints

Is there a difference between the projection using the camera matrix and the projection using the Opencv method? If so, why? Please explain shortly.

Section F

Choose another image from the images used in the calibration process. Draw a cube on top the upper left black square of the chessboard present in the chosen image.

1. **Hint** - First find out the world points of the cube and project them on top of the image as shown in section E. You can draw the cube skeleton using Matplotlib's line plotting methods or Opencv's cv2.line.

2. Rotate the cube points 30° clockwise around one of the axes and display the result

Perform the drawing in the Jupyter notebook.

Section G

A student at the course didn't have an access to a camera. Instead of taking pictures, the student obtained a collection of chessboard images from multiple sources on the internet and used them instead. Is there a problem with this approach? If so, explain what the student did wrong.