Principles of Robot Autonomy I
Homework 3
Due Friday, November 1 (11:59pm)

You will submit your homework to Gradescope. Your submission will consist of a single pdf with your answers for written questions and relevant plots from code.

Your submission must be typeset in LaTeX.

Problem 1: Chessboard Camera Extrinsics

Starter code for this question has been made available online through GitHub. To get started, download the code by running `git clone https://github.com/StanfordASL/AA274a-HW3.git` in a terminal window.

NOTE: In this homework, we'll be using a very popular open-source computer vision package named OpenCV. This is reflected in the `requirements.txt` file in the homework repository.

Introduction

Objective: In this problem you will solve one of the fundamental problems in computer vision, computing the position of a camera relative to a known marker arrangement. In this case, the markers will be the corners of the squares on a checkerboard with known dimensions. In this problem you will learn how cameras can be used for localization against features, and how projection to and from the image plane into world coordinates works!

Setup: This entire problem is contained in the `p1_extrinsics.ipynb` Jupyter notebook. We recommend developing in the VM to mitigate any dependency issues, but feel free to develop locally if you are comfortable setting up a python environment and installing the necessary dependencies.

Background:

Camera Extrinsics: The position $\mathbf{t} \in \mathbb{R}^3$ and orientation $\mathbf{R} \in SO(3)$ of a camera with respect to some "world" frame are often jointly referred to as the camera *extrinsic* parameters, or *extrinsics* for short. Once obtained, the extrinsics can be used to transform a 3D point in the world frame $\mathbf{P} \in \mathbb{R}^3$ into a 3D point in the camera frame $\mathbf{P}' \in \mathbb{R}^3$ as follows:

$$\mathbf{P}' = \mathbf{R}\mathbf{P} + \mathbf{t} \tag{1}$$

Notice that in Eq. 1, $\mathbf{R}$ and $\mathbf{t}$ actually represent the position and orientation of the world frame with respect to the camera frame. The goal of this problem is to compute $\mathbf{R}$ and $\mathbf{t}$ of a camera for a world frame that lies on the surface of a chessboard.

Chessboard: Everything in this problem starts with a single image of a checkerboard. Without adding any code you can run through the first two few cells in the Jupyter notebook to see what image we are working with for this problem. We use a checkerboard because we can easily physically measure the side-lengths of the squares (for our problem the squares have a dimension of 0.0205m). Furthermore, the black and white

pattern of the checkerboard has high contrast and it makes it easy to find the features that we are going to use for calibration, square "cross-junctions".
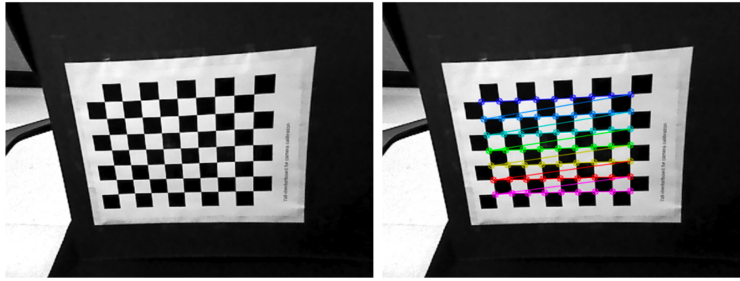

Figure 1: Cross-junction (corner) extraction of chessboards.

Feature Extraction

The first step towards computing the pose of the camera relative to the checkerboard is accurately estimating the position of the cross-junctions in the image frame. That is, in the image of the checkerboard, which pixels correspond to the cross-junctions? Fortunately, for us modern roboticists this question was answered quite a while ago by researchers studying computer vision. The solution is to apply a series of transformations on the image that isolate the high-contrast regions of the squares on the board, and then to use the Harris Corner Detection Algorithm to detect the cross-junctions. This combination of methods is provided in the `cv2.findChessboardCorners(...)` function provided by OpenCV.

(i) 🖥 Use the `cv2.findChessboardCorners(...)` along with the corner parameters specified in the code to extract the cross-junction pixel locations, and plot them on top of the image using the `ax.plot(...)` function. Include this plot in your write-up.
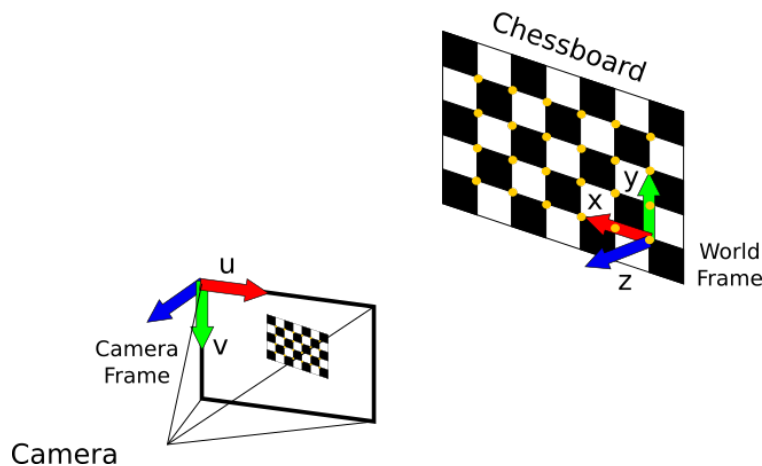
Extrinsic Calibration

Figure 2: Chessboard problem setup.

In this part of the problem you will use a known camera intrinsics matrix $\mathbf{K}$, the known distance between cross-junctions (0.0205 m), and the cross-junction image coordinates to compute the relative position and rotation of the camera to the chessboard.

To solve for the position and rotation of the camera relative to the chessboard you will make use of a method that is similar to the solution that was presented for the PnP problem discussed in Lecture 8. The main difference here is that all of the points exist on the surface of the chessboard, causing all of their z-components

to be zero in the world frame, see Figure 2. Inspecting the equation for transforming world coordinates into homogeneous pixel coordinates we can see that this results in a slightly different set of equations:

$$
\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} \propto \mathbf{K}[\mathbf{R} \ \mathbf{t}] \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} = \mathbf{K}[\mathbf{r_0} \ \mathbf{r_1} \ \mathbf{r_2} \ \mathbf{t}] \begin{bmatrix} X \\ Y \\ 0 \\ 1 \end{bmatrix} = \mathbf{K}[\mathbf{r_0} \ \mathbf{r_1} \ \mathbf{t}] \begin{bmatrix} X \\ Y \\ 1 \end{bmatrix} = \mathbf{H} \begin{bmatrix} X \\ Y \\ 1 \end{bmatrix} \tag{2}
$$

where $\mathbf{r_0}$ is the first column of $\mathbf{R}$ and so on. We know the location of pixels in both image coordinates (due to cross-junction detection), and in world coordinates (from known chessboard geometry). Therefore, equation 2 forms a system of equations for each 2D-3D correspondence. For cross-junction i the resulting set of equations is

$$
\begin{bmatrix} -X_i & -Y_i & -1 & 0 & 0 & 0 & u_iX_i & uY_i & u_i \\ 0 & 0 & 0 & -X_i & -Y_i & -1 & v_iX_i & v_iY_i & v_i \end{bmatrix} \begin{bmatrix} \mathbf{h_0}^\top \\ \mathbf{h_1}^\top \\ \mathbf{h_2}^\top \end{bmatrix} = \mathbf{M}_i \begin{bmatrix} \mathbf{h_0}^\top \\ \mathbf{h_1}^\top \\ \mathbf{h_2}^\top \end{bmatrix} = 0, \tag{3}
$$

where $\mathbf{h_0} \in \mathbb{R}^{3\times1}$ is the first row of $\mathbf{H} \in \mathbb{R}^{3\times3}$. Forming $\mathbf{M_i} \in \mathbb{R}^{2\times9}$ for each of the points and stacking them vertically forms a new matrix $\mathbf{M}$ whose null-space corresponds to the flattened elements of $\mathbf{H}$. The null-space of a matrix can be found by computing the Singular Value Decomposition, and choosing the column of the $\mathbf{V}$ matrix with the smallest singular value.

After computing $\mathbf{H}$, using the known intrinsics matrix for the camera $\mathbf{K}$, the pose of the camera relative to the world frame can be computed as

$$
\mathbf{K}^{-1}\mathbf{H} = \lambda[\mathbf{r_0} \ \mathbf{r_1} \ \mathbf{t}]. \tag{4}
$$

The scale factor, λ, arises from the use of homogeneous coordinates. One way to compute it is to choose $\lambda = ||(\mathbf{K}^{-1}\mathbf{H})_{(:,0)}||_2$, or equal to the norm of the first column of the left hand side matrix in Equation 4. This corresponds to a scaling such that the first column of the resulting rotation matrix has a norm equal to 1. Finally, the third column of the rotation matrix must be orthoganol to the other two columns, and therefore $\mathbf{r_2} = \mathbf{r_0} \times \mathbf{r_1}$.

To summarize,

$$
\mathbf{r_0} = \frac{1}{\lambda}(\mathbf{K}^{-1}\mathbf{H})_{(:,0)} \quad \mathbf{r_1} = \frac{1}{\lambda}(\mathbf{K}^{-1}\mathbf{H})_{(:,1)} \quad \mathbf{r_2} = \mathbf{r_0} \times \mathbf{r_1} \quad \mathbf{t} = \frac{1}{\lambda}(\mathbf{K}^{-1}\mathbf{H})_{(:,2)}. \tag{5}
$$

(ii) 🖊 Use the method outlined above, and the corner locations from (i) to compute the rotation, $\mathbf{R}$, and translation, $\mathbf{t}$, of the camera with respect to the chessboard in the image, and include the values that you compute in your write up with elements up to two significant figures.

World to Camera Projection

Using the camera intrinsic and extrinsic parameters, arbitrary points can be projected onto the camera frame from the world frame.

(iii) 🖥 Complete the function `transform_world_to_camera`, and use it to project the world-frame chessboard corner locations onto the image along with the actual corner locations. Include the resulting plot in your write-up. Hint: you might find Equation 2 useful for this part.

Finally, run the last cell in the notebook to project a secret point cloud in world frame onto your chessboard. This is an example of how the methods in this problem can be used for augmented reality applications (no need to submit anything for this).

Problem 2: Object Detection using PyTorch

Objective: Detect objects in a given image using a pre-trained Convolutional Neural Network (CNN) model from PyTorch.

Setup: The code for this problem is provided through this Google Colab notebook:
https://colab.research.google.com/drive/1QPjoZa7-2l8AsdXcBTsPjzQmhLuNp_hf?usp=sharing
You will not have edit access to this notebook. To create a copy in your own Google Drive, select `File > Save a Copy in Drive`. Accept the prompt to open your copy in a new tab. You will have edit access to this copy. If you need to find where in your drive the copy is, select `File > Locate in Drive`.

In this assignment, you are tasked with running a CNN model on an image to determine which objects are in it. Fortunately, the model is pre-trained and most of the setup is done for you as well.

You should follow along in the notebook `HW3_P2_Object_Detection.ipynb`, which has 3 distinct parts: Environment setup, Object detection, and Visualizing bounding boxes. To begin, take a look at the first code block to understand how the environment was setup. In parts 2 and 3, you get to apply PyTorch functions to integrate and run an "off-the-shelf" CNN model and then plot the boxes detected by the model.

When you are done with the notebook, return here and complete the following short answer questions.

(i) 🖥️ Include a screenshot of the cars image with bounding boxes with `score_threshold` of 0.6.

(ii) ✏️ What happens to the bounding boxes if you increase `score_threshold` to 1? Why do you think this is the case? Feel free to try running your code on the other provided images to inform your response.

(iii) ✏️ What happens to the bounding boxes if you decrease `score_threshold` to 0.01? Why do you think this is the case? Feel free to try running your code on the other provided images to inform your response.