ENGN2560 Computer Vision

Lab01: Camera Calibration, Camera Projection, and Image Features

The goal of this lab is to:

- Learn basic concepts of processing images in MATLAB.
- Learn how an image is formed by projecting 3D points to a 2D image plane.
- Learn how to calibrate your camera to find intrinsic and extrinsic matricies.
- Explore image features.

Part 0: Image Processings in MATLAB

Labs of this course basically require MATLAB for implementation, and an image processing toolbox is mandatory to be installed. A color image read by MATLAB, *i.e.*, using `imread` function, is a *matrix* of size $N_r \times N_c \times 3$, where N_r is the number of rows, N_c is the number of columns, and each entry is a pixel containing 3 channels, namely, R, G, and B colors. Most of the time the data type of a color image is *uint*8 (unsigned integer 8-bit), which is the data type `imshow` function requires for displaying the image.

A lot of times we convert a color image to a gray image through `rgb2gray` function, and the value of each pixel is now the *brightness* or *intensity* of that pixel. In addition, we also convert the data type of an image I from `uint8` to either single precision or double precision of floating point numbers using `single(I)` or `double(I)`. When accessing the value of a pixel on an image I, say at the 10^{th} row and 15^{th} column, use $I(10, 15)$ to get the returned value. For the image coordinate, however, the x-axis is the column of the image while the y-axis is the row of the image, respectively. Thus, when superimposing a point on an image, say a point located at x=10 and y=15, it should be at the pixel location for which $I(15, 10)$ returns its value.

Part I: A Brief Description of a Pinhole Camera Projection Model

Pinhole Camera Model: Assume that we are using a classic pinhole camera, where an image is undistorted and the camera calibration matrix is

$$K = \begin{bmatrix} \alpha & \sigma & \xi_o^{im} \\ 0 & \beta & \eta_o^{im} \\ 0 & 0 & 1 \end{bmatrix}., \tag{1}$$

where α and β are focal lengths in x and y directions, β is the skew, and $(\xi_o^{im}, \eta_o^{im})$ is the principal point (or optic center) of the camera. Refer to `Camera_Calibration.pdf` file for more information.

Camera Projection: Projecting a 3D scene point to a 2D point In order to know how an image is formed, we must know how a world scene is projected to an image plane. The camera coordinate with respect to a world coordinate is typically represented by a rotation matrix $R \in \mathbb{R}^{3\times3}$ and a translation vector $T \in \mathbb{R}^{3\times1}$. They describe the transformation of a 3D point from the world coordinate to the camera coordinate, *i.e.*, a 3D point Γ_w in a world coordinate is transformed to Γ_c in the camera coordinate by

$$\Gamma_c = R\Gamma_w + T. \tag{2}$$

The 3D point Γ_c under the camera coordinate can be represented as a point with some magnitude in the camera coordinate (a vector) scaled up by some scalar, or the depth ρ from the camera center, Figure 1, *i.e.*,

$$\Gamma_c = \rho\gamma, \tag{3}$$

where $\gamma = [\xi, \eta, 1]^T$ is a point in *meters*. Thus, the third dimension of Γ_c is the depth ρ. γ is said to be represented by a *homogeneous* coordinate, meaning that all 3D points Γ_c are scaled down to the same image plane (red in Figure 1) as 2D points with homogenous depths. The transformation of γ under the camera coordinate system (green in Figure 1) to the point $\gamma_{im} = [\xi_{im}, \eta_{im}, 1]^T$ on an image coordinate system (red in Figure 1) is captured by a *calibration matrix* K (often times it is also called the camera *intrinsic* matrix),

$$\gamma_{im} = K\gamma, \tag{4}$$

where γ_{im} is a 2D point in *pixels*. Combining Equations 3 and 4, projecting a 3D point in a world coordinate to a 2D image plane is

$$\gamma_{im} = \frac{1}{\rho} K \left(R\Gamma_w + T \right). \tag{5}$$

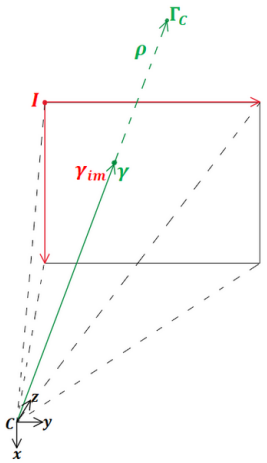


Figure 1: The projection of a 3D point Γ_c (green) in the camera coordinate (black) to a point γ_{im} (red) in pixels in image coordinate can be captured by a camera calibration matrix.

Part II: Problems

Problem 1. Image Formation by Camera Projection Assume that the two cameras have the same resolution of 1200 × 1600 pixels (rows × columns) with the same calibration matrix capturing the same 3D scene, Figure 2. The 3D scene is represented by a dense cloud of 3D points storing $[X, Y, Z]$ coordinates together with their colors stored in (R, G, B), and the relative poses of the two cameras with respect to the world coordinate are represented by their rotation and translation matrices, *i.e.*, R_1, T_1, R_2, and T_2.

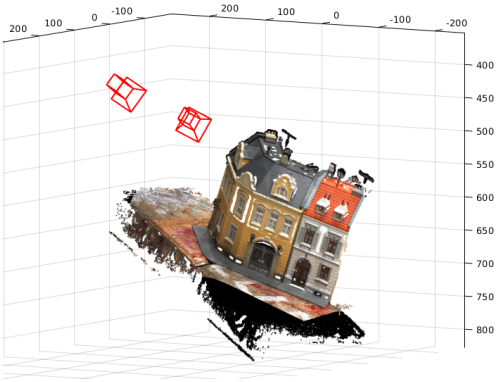


Figure 2: A 3D scene represented by a cloud of points is viewed by two cameras.

Use the provided scene points, camera calibration matrix, and relative pose encoded by `.mat` files, project the 3D points to the two image planes. Implement it in the provided `P1_main.m` file. Steps to follow are:

1. Create matrices representing images with size the same as the image resolution.
2. The scene points are all under the world coordinate. Use Equation 5 to project them to the images. Note that the projected points located outside the image boundaries should not be considered.
3. Direct projections on a blank image plane might give you an image shown in Figure 3(a). To construct a nicer and a rather compact image, Figure 3(b), create a `meshgrid` of the size same as the image and use `griddata` function to interpolate the projected points.

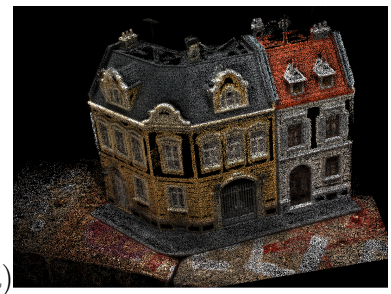
(a)  (b) 

Figure 3: Examples of projecting 3D points directly on a blank image plane (a) and an image with a nicer and compact formation (b).

Problem 2. Calibrate Your Camera In this problem, we aim to estimate your camera intrinsic matrix, Equation 1, and extrinsic matrix with respect to some world coordinate. Carefully read how those matrices can be estimated in the `Camera_Calibration.pdf` file from course canvas site. Implement the code in the provided `P2_main.m` file by following the steps below.

1. Print out the checkerboard picture provided in this lab, Figure 4. You may paste it on a wall or some cardboard to make sure that the the checkerboard remains a plane during the clibration process.

2. Let the top left corner of your checkerboard be the origin of the world coordinate. Measure the width of each square on the checkerboard to get the corner locations (x, y, z), where x and y can be the horizontal and the vertical directions, respectively, and z is any arbitrary value for all points. These locations are the 3D points in the world coordinate. You are allowed to use `generateCheckerboardPoints` function to get those locations.

3. Take one picture of your checkerboard using your camera. Detect corners on the checkerboard using `detectCheckerboardPoints` function or `detectHarrisFeatures` function. When taking the pictures, make sure that the checkerboard is not occluded so that every corner on the checkerboard is detected. The detected corner locations are the 2D points in pixels in image coordinate.

4. Correspond N 3D corner locations measured in step 2 and 2D corner locations detected in step 3.

5. Solve the intrinsic and extrinsic matrices of your camera using those N correspondences.

In your report, you should include pictures of your printed checkerboard and the detected corners, and write down the estimated calibration matrix, rotation matrix, and translation vector of your camera.

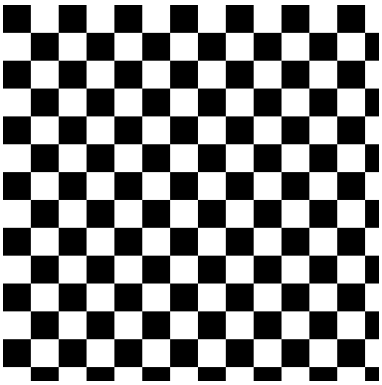


Figure 4: A checker board used to calibrate your camera.

Problem 3. Image Features Image features are interest points on an image. An example is a corner on the checker board in Figure 4. Among many different types of features, SIFT stands out to be the most robust features. In this problem, we are going to find SIFT features from images.

Download and install the VLFeat library from `http://www.vlfeat.org/`. Go through the tutorial of detecting SIFT features available at `http://www.vlfeat.org/overview/sift.html`. Play with the input parameters, *e.g.*, `PeakThresh` and `edgethresh`, and plot SIFT keypoints (features) on two pairs of images in Figure 5. Put your code in the provided `P3_main.m` file. Do you think SIFT features have high repeatability rate, namely, the same feature appears at the same place across the two images? Put your answers in the report.

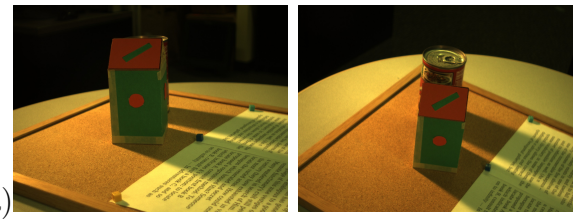
(a)  (b) 

Figure 5: Two pairs of images with (a) view changes, and (b) scale/brightness changes.