University at Buffalo
Department of Computer Science and Engineering
CSE 473/573 - Computer Vision and Image Processing

Fall 2025

Project #1

Due Date: 10/7/2025, 11:59PM

1 Rotation Matrix (40 points)

Figure 1 illustrates the transformation from coordinate xyz to coordinate-XYZ in the following these three steps: step 1) rotate around x axis with α (coordinate-xyz to coordinate-x y'z'); step 2) rotate around $y\prime$ axis with β (coordinate-x y'z' to coordinate-x"y'z"); step 3) rotate around $z\prime\prime$ axis with γ (coordinate-x"y'z" to coordinate-XYZ). α, β, and γ are all given in angles (not radians), and keep angles within range $0° < \alpha, \beta, \gamma < 90°$ To avoid Gimbal lock .

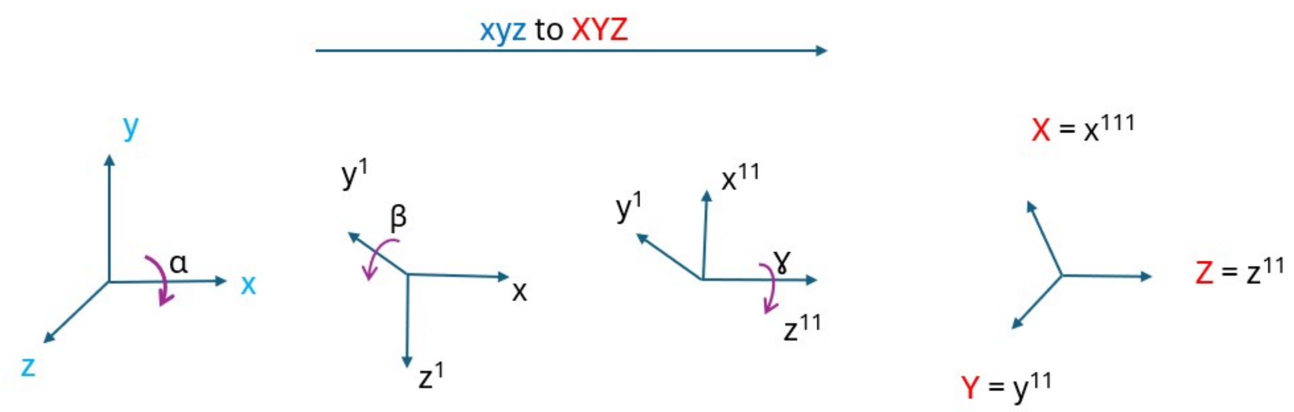

Figure 1: Frame Rotation sequence.

- Design a program to get the rotation matrix from xyz to XYZ. (20 points)
- Design a program to get the rotation matrix from XYZ to xyz. (20 points)

2 Camera Calibration (60 points)

Preliminary.1.

The projection from world coordinate to image plane can be indicated by intrinsic parameters (Camera) and extrinsic parameters (World). From world coordinate to camera coordinate, the extrinsic parameters can be used as

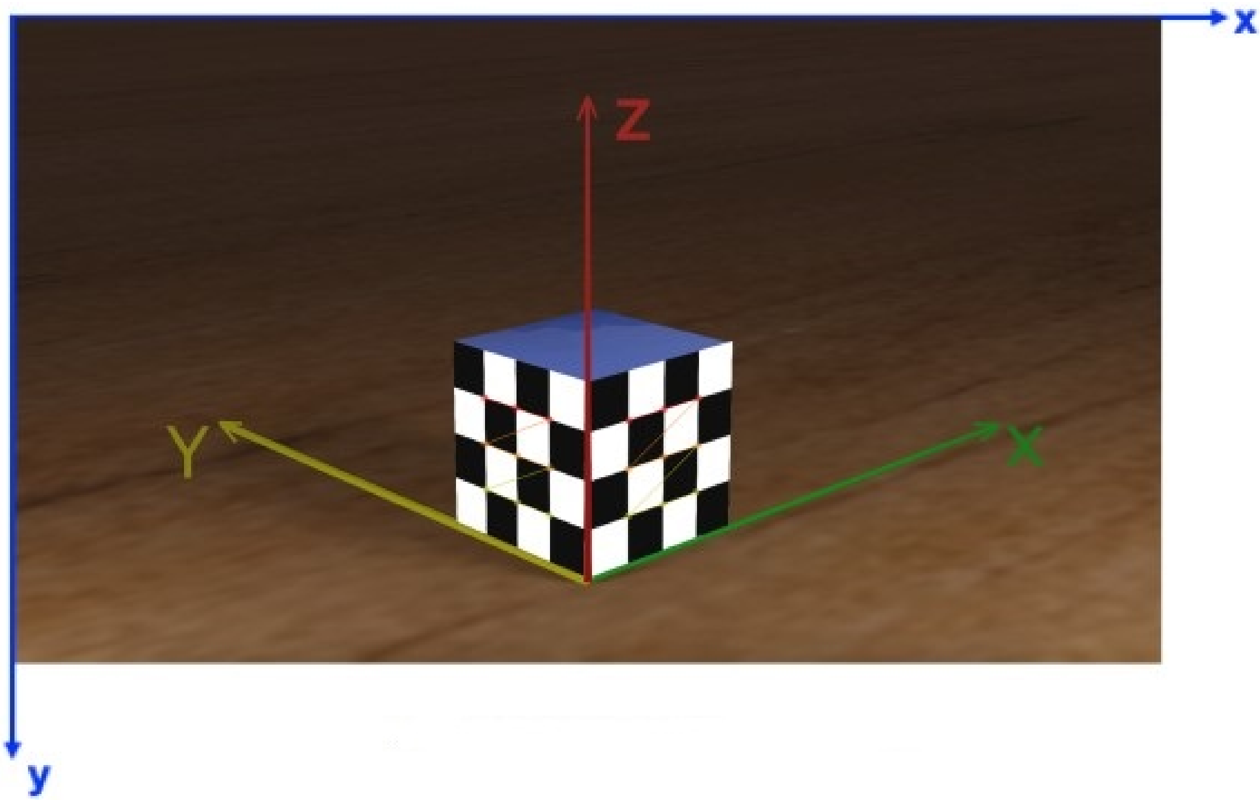

Figure 2: Image of checkered board.

$$
\begin{aligned}
M = M_{in} \cdot M_{ex} &= \begin{bmatrix} f_x & 0 & o_x \\ 0 & f_y & o_y \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} r_{11} & r_{12} & r_{13} & T_x \\ r_{21} & r_{22} & r_{23} & T_y \\ r_{31} & r_{32} & r_{33} & T_z \end{bmatrix} \\
\begin{bmatrix} m_{11} & m_{12} & m_{13} & m_{14} \\ m_{21} & m_{22} & m_{23} & m_{24} \\ m_{31} & m_{32} & m_{33} & m_{34} \end{bmatrix} &= \begin{bmatrix} f_x r_{11} + o_x r_{31} & f_x r_{12} + o_x r_{32} & f_x r_{13} + o_x r_{33} & f_x T_x + o_x T_z \\ f_y r_{21} + o_y r_{31} & f_y r_{22} + o_y r_{32} & f_y r_{23} + o_y r_{33} & f_y T_y + o_y T_z \\ r_{31} & r_{32} & r_{33} & T_z \end{bmatrix}.
\end{aligned} \tag{1}
$$

Here, M is projection matrix. Let's define $\mathbf{m}_1 = (m_{11}, m_{12}, m_{13})^T, \mathbf{m}_2 = (m_{21}, m_{22}, m_{23})^T, \mathbf{m}_3 = (m_{31}, m_{32}, m_{33})^T, \mathbf{m}_4 = (m_{14}, m_{24}, m_{34})^T$. Also define $\mathbf{r}_1 = (r_{11}, r_{12}, r_{13})^T, \mathbf{r}_2 = (r_{21}, r_{22}, r_{23})^T, \mathbf{r}_3 = (r_{31}, r_{32}, r_{33})^T$. Observe that $(\mathbf{r}_1, \mathbf{r}_2, \mathbf{r}_3)$ is the rotation matrix, then

$$
(\mathbf{r}_1, \mathbf{r}_2, \mathbf{r}_3) \begin{pmatrix} \mathbf{r}_1^T \\ \mathbf{r}_2^T \\ \mathbf{r}_3^T \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}.
$$

Then we have $\mathbf{r}_i^T \mathbf{r}_i = 1, \quad \mathbf{r}_i^T \mathbf{r}_j = 0 \ (i \neq j)$.

From M we have

$$
\begin{aligned}
\mathbf{m}_1^T \mathbf{m}_3 &= r_{31}(f_x r_{11} + o_x r_{31}) + r_{32}(f_x r_{12} + o_x r_{32}) + r_{33}(f_x r_{13} + o_x r_{33}) \\
&= f_x(r_{11} r_{31} + r_{12} r_{32} + r_{13} r_{33}) + o_x(r_{31}^2 + r_{32}^2 + r_{33}^2) \\
&= f_x(\mathbf{r}_1^T \mathbf{r}_3) + o_x(\mathbf{r}_3^T \mathbf{r}_3) \\
&= o_x
\end{aligned} \tag{2}
$$

Similarly, Next, from M we have

$$
\begin{aligned}
\mathbf{m}_1^T \mathbf{m}_1 &= (f_x r_{11} + o_x r_{31})^2 + (f_x r_{12} + o_x r_{32})^2 + (f_x r_{13} + o_x r_{33})^2 \\
&= f_x^2 \cdot \mathbf{r_1}^T \mathbf{r_1} + 2 f_x o_x \cdot \mathbf{r_1}^T \mathbf{r_3} + o_x^2 \cdot \mathbf{r_3}^T \mathbf{r_3} = f_x^2 + o_x^2
\end{aligned} \tag{3}
$$

So $f_x = \sqrt{\mathbf{m}_1^T\mathbf{m}_1 - o_x^2}$. Similarly we have $o_y = \mathbf{m}_2^T\mathbf{m}_3$, $f_y = \sqrt{\mathbf{m}_2^T\mathbf{m}_2 - o_y^2}$. Overall, we come to the conclusion as follows

$$o_x = \mathbf{m}_1^T\mathbf{m}_3 \quad o_y = \mathbf{m}_2^T\mathbf{m}_3 \tag{4}$$

$$f_x = \sqrt{\mathbf{m}_1^T\mathbf{m}_1 - o_x^2} \quad f_y = \sqrt{\mathbf{m}_2^T\mathbf{m}_2 - o_y^2} \tag{5}$$

Preliminary.2.
Let $X_wY_wZ_w$ be the world coordinate and xy be the image coordinate, we have the transformation matrix $M \in \mathbb{R}^{3\times4}$:

$$s\begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} m_{11} & m_{12} & m_{13} & m_{14} \\ m_{21} & m_{22} & m_{23} & m_{24} \\ m_{31} & m_{32} & m_{33} & m_{34} \end{bmatrix} \cdot \begin{bmatrix} X_w \\ Y_w \\ Z_w \\ 1 \end{bmatrix} \tag{6}$$

$$\begin{aligned} sx &= m_{11}X_w + m_{12}Y_w + m_{13}Z_w + m_{14}, \\ sy &= m_{21}X_w + m_{22}Y_w + m_{23}Z_w + m_{24}, \\ s &= m_{31}X_w + m_{32}Y_w + m_{33}Z_w + m_{34}. \end{aligned} \tag{7}$$

We can solve m_{ij} with the equation below:

$$\begin{bmatrix} X_w^1 & Y_w^1 & Z_w^1 & 1 & 0 & 0 & 0 & 0 & -x^1X_w^1 & -x^1Y_w^1 & -x^1Z_w^1 & -x^1 \\ 0 & 0 & 0 & 0 & X_w^1 & Y_w^1 & Z_w^1 & 1 & -y^1X_w^1 & -y^1Y_w^1 & -y^1Z_w^1 & -y^1 \\ & & & & & & \vdots & & & & & \\ X_w^n & Y_w^n & Z_w^n & 1 & 0 & 0 & 0 & 0 & -x^nX_w^n & -x^nY_w^n & -x^nZ_w^n & -x^n \\ 0 & 0 & 0 & 0 & X_w^n & Y_w^n & Z_w^n & 1 & -y^nX_w^n & -y^nY_w^n & -y^nZ_w^n & -y^n \end{bmatrix} \cdot \begin{bmatrix} m_{11} \\ m_{12} \\ m_{13} \\ m_{14} \\ m_{21} \\ m_{22} \\ m_{23} \\ m_{24} \\ m_{31} \\ m_{32} \\ m_{33} \\ m_{34} \end{bmatrix} = \mathbf{0}, \tag{8}$$

where the first matrix is with size $2n \times 12$ (n is the number of available points).
Preliminary.3.
Solve the homogeneous linear equation $\mathbf{A}\mathbf{x} = 0$, where $\mathbf{x}$ is the vector of N unknowns, and $\mathbf{A}$ is the matrix of $M \times N$ coefficients. A quick observation is that there are infinite solutions for $\mathbf{A}\mathbf{x} = 0$, since we can randomly scale x with a scalar λ such that $\mathbf{A}(\lambda\mathbf{x}) = 0$. Therefore, we assume $\|\mathbf{x}\| = 1$. Solving the equation can be converted to

$$\min \|\mathbf{A}\mathbf{x}\| \tag{9}$$

The minimization problem can be solved with Singular Value Decomposition (SVD). Assume that $\mathbf{A}$ can be decomposed to $\mathbf{U}\boldsymbol{\Sigma}\mathbf{V}^T$, we have

$$\min \|\mathbf{A}\mathbf{x}\| = \|\mathbf{U}\boldsymbol{\Sigma}\mathbf{V}^T\mathbf{x}\| = \|\boldsymbol{\Sigma}\mathbf{V}^T\mathbf{x}\|. \tag{10}$$

Note that $\|\mathbf{V}^T\mathbf{x}\| = \|\mathbf{x}\| = 1$, then let $\mathbf{y} = \mathbf{V}^T\mathbf{x}$, so we have

$$\begin{aligned}\min\|\mathbf{A}\mathbf{x}\| &= \|\mathbf{\Sigma}\mathbf{y}\| \\ &= \left\| \begin{bmatrix} \sigma_1 & & \\ & \ddots & \\ & & \sigma_n \\ 0 & \cdots & 0 \\ \vdots & \vdots & \vdots \\ 0 & \cdots & 0 \end{bmatrix} \begin{bmatrix} y_1 \\ \vdots \\ y_n \end{bmatrix} \right\|,\end{aligned} \tag{11}$$

where $\sigma_1 \geq \cdots \geq \sigma_n \geq 0$. Recall that $\|\mathbf{y}\| = 1$, we can set

$$\begin{bmatrix} y_1 \\ \vdots \\ y_n \end{bmatrix} = \begin{bmatrix} 0 \\ \vdots \\ 1 \end{bmatrix}. \tag{12}$$

So $\mathbf{x}$ should be the last row of $\mathbf{V}^T$.

Preliminary.4.

You need to do normalize (or scaling) the projection matrix (after SVD). For more detailed procedure check out the section 7.1, page 179: "Over-determined solution" in the book "Multiple View Geometry in Computer Vision" for details.

Question

Figure 2 shows an image of the checkerboard, where XYZ is the world coordinate and xy is marked as the image coordinate. The edge length of each grid on the checkerboard is $10mm$ in reality. Suppose one pixel of the image is equivalent to $1mm$. You can calculate the projection matrix from world coordinate to image coordinate based on the 18 corners (marked points) on the checkerboard (9 corners in each side of the checkerboard do not take the ones in the axis). From the projection matrix you can get the intrinsic matrix which is indicated as $\begin{bmatrix} f_x & 0 & o_x \\ 0 & f_y & o_y \\ 0 & 0 & 1 \end{bmatrix}$ (f_x and f_y are not necessarily be equal).

- Design a function to get the image coordinates of the 18 corners from the image. You can decide the order of output points for yourself. (20 point)
- Manually (or design a program) to get the world coordinate of the 18 corners. Note that the output order should be the SAME with the last question. (10 point)
- Design a function to get the intrinsic parameters f_x, f_y, o_x, o_y from the image coordinates and world coordinates acquired above. (20 points)
- Design a function to get the extrinsic parameters R, T from the image coordinates and world coordinates acquired above. (10 points)

Instructions (Please read this very carefully!):

- There is a restricted use of **OpenCV** wherein we allow only for the already imported functions in the given code template to be used for this project.
- All the data should be in tensor form and should be operated based on Pytorch except in the **find_corner_img_coord** function section (in this section you are able to use opencv to detect corners, resulting in numpy arrays, but you have to convert numpy arrays back to torch.Tensor form).

- Please implement all your code in file **"geometry.py"**. Please do NOT make any changes to any file except **"geometry.py"**.
- To submit your code and result, Please run **"pack_submission.sh"** to pack your code and result into a zip file. You can find the command line in **"README.md"** Note that when packing your submission, the script would run your code before packing 1) The resulting zip file is only file you need to submit at UBlearns.2) Please also push the latest code to your classroom git repository , (This is only for progress tracking and can influence your project grade based of the .zip submission on the UBLearns)
- The packed submission file should be named **"submission_<** *Your_UBIT_name* **>.zip"**, and it should contain 3 files, named **"result_task1.json"**, **"result_task2.json"**, and **"geometry.py"**. If not, there is something wrong with your code/filename, please go back and check.
- You are ONLY allowed to use the library that are already imported in the script.
- We grade this project based on the results we get from running your code. If you do not give correct final results, you are very likely to get NO partial points for that step, even if it may because you did the former parts wrong.
- Late submissions are only accepted within the allowed late days.
- Anyone whose raise "RuntimeError", your grade will be 0 for that task.
- Anyone whose code is evaluated as plagiarised, your grade will be 0 for this project and will be reported for Academic Integrity Violation.