Homework 3: Camera Geometry

Kaveh Fathian, Email: fathian@ariarobotics.com

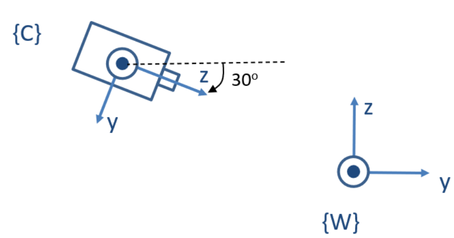
Handout: 2024-10-11
Due: 2024-10-28, at 3:00 PM on Canvas

Instructions:

- Homework is on a "**rolling"** basis and more **questions will be added until 1 week** before the due date. There will be an announcement (on Discord or in class) when new questions are added.
- For all problems in this homework, you can convert your images to **grayscale** for simplicity. So, no need to work with RGB images (unless you want to).
- You can get help from your teammates (or others) for all problems and/or code. However, you will need to code the problems and submit your report **individually** on Canvas. Reports/code that are **identical** will receive a grade of **zero**.
- The **quiz** will strongly resemble homework questions, and if you understood/coded the homework yourself, you will be able to answer the quiz questions immediately. Because the quiz will be closed-notes & no internet access, understanding the homework is crucial!
- **Deliverables**: You will submit a **single PDF** file to Canvas. The PDF must contain your **answers**, your **code** (copy-paste in the document), and any requested **outputs** (like images). For convenience, you may use Jupyter notebook and convert it to a PDF.
- Only use **images provided** in the homework material, as requested by each problem. Using any other image, will result in a **grade of zero**.
- **Grading:** This homework will be scaled to **10pts** of your final grade. Grading **rubric** will be posted on **Canvas** after the assignment due date.

Problem 1: Coordinate transformation:

- 2D transformation: Compute the coordinate of a 2D point p = $(10, 20)^T$ using a transformation of 45 degrees about the x-axis, and a translation of t = $(40, -30)^T$. Answer/explain the following:
    - What is the point representation in <u>homogeneous</u> coordinates?
    - What is the rotation matrix R?
    - What is the translation vector t?
    - What is the full transformation matrix (consisting of R, t) that can be used to transform the <u>homogeneous</u> point coordinate?
    - How do we apply this transformation to the point (in homogeneous coordinate form)?
    - What is the coordinate of the transformed point, in homogeneous coordinates, and in the cartesian coordinates?
- 3D transformation: A camera is located at point (0,-5,3) in the world frame. The camera is tilted down by 30 degrees from the horizontal. We want to find the 4x4 homogeneous transformation ${}^{C}H_{W}$ from the world frame {W} to camera frame {C}. Note that in "the world" Z is up (X-Y ground plane) but in "the camera", Z is out (X-Y image plane).



Answer/explain the following:

- What is ${}^{C}H_{W}$? Explain how you computed it.

  - Using transformation ${}^{C}H_{W}$, transform the point ${}^{W}p = (0,0,1)$ in the world frame to the camera frame. Hint: use the homogeneous coordinates of the point for this transformation.

Notes:

- If you are not familiar with coordinate transformations, please take a look at the notes "Coordinate_Transforms.pdf" in the HW3 folder of course materials.

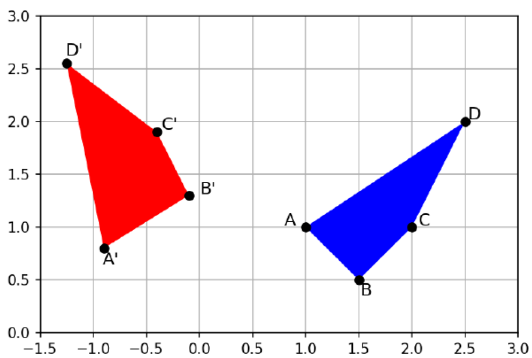
Problem 2: Camera calibration:

- Find the calibration/ intrinsic matrix of a camera (e.g., your cellphone camera). Use the camera calibration board (print PDF file) provided in the HW3 folder.
  - Provide a copy of your code in the report
  - Display the images you took from the calibration board (at different angles/locations)
  - After calibration, print out the camera intrinsic matrix
  - Print out five distortion parameters, and explain what they are for.
  - Print out camera extrinsic matrices for all of your images

Notes:

- For this problem, you can follow the camera calibration instructions at
  - https://docs.opencv.org/4.x/dc/dbb/tutorial_py_calibration.html
  - https://www.geeksforgeeks.org/camera-calibration-with-python-opencv/

Problem 3: Least-squares estimation:
Suppose we have a quadrilateral ABCD and a transformed version A'B'C'D' as seen in the image below.



$A = (1,1)$ $\quad$ $A' = (-0.9, 0.8)$

$B = (1.5, 0.5)$ $\quad$ $B' = (-0.1, 1.3)$

$C = (2,1)$ $\quad$ $C' = (-0.4, 1.9)$

$D = (2.5, 2)$ $\quad$ $D' = (-1.25, 2.55)$

Let's assume that each point in ABCD was approximately mapped to its corresponding point in A'B'C'D' by a 2x2 transformation matrix M. That is, if $X = \begin{bmatrix} x \\ y \end{bmatrix}$, and $X' = \begin{bmatrix} x' \\ y' \end{bmatrix}$, and $M = \begin{bmatrix} m_{11} & m_{12} \\ m_{21} & m_{22} \end{bmatrix}$, then

Homework 3: Camera Geometry

Kaveh Fathian, Email: fathian@ariarobotics.com

Handout: 2024-10-11
Due: 2024-10-28, at 3:00 PM on Canvas

$$\begin{bmatrix} m_{11} & m_{12} \\ m_{21} & m_{22} \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} \approx \begin{bmatrix} x' \\ y' \end{bmatrix}$$

We would like to approximate M using least squares for linear regression.

- Rewrite the equation $M\,X \approx X'$ into a pair of linear equations by expanding the matrix multiplication. Do this by replacing each of the “__” below with x, y, x', y', or 0, and **print out** the answer.

$$\begin{cases} \_\_\, m_{11} + \_\_\, m_{12} + \_\_\, m_{21} + \_\_\, m_{22} = \_\_ \\ \_\_\, m_{11} + \_\_\, m_{12} + \_\_\, m_{21} + \_\_\, m_{22} = \_\_ \end{cases}$$

- With the quadrilaterals in question, there are 4 points that transform. So we should expect to see 8 such equations (2 for each point) that use the transformation equation M. From stacking these pairs of equations for all point correspondences, we can construct a 8x4 matrix Q and a 8x1 column vector b that satisfy

$$Q \begin{bmatrix} m_{11} \\ m_{12} \\ m_{21} \\ m_{22} \end{bmatrix} = b$$

  Find and **print out** Q and b (which are based on the coordinate values of ABCD and A’B’C’D’).
- Our problem is now over-constrained, so we want to find m_{ij} that minimize the least squared error between, i.e., we want to minimize $\|Qm - b\|^2$, where m is the 4x1 vector, shown above, containing m_{ij} elements. Use `numpy.linalg.lstsq()` to take in the Q matrix and b vector you found above, and return the solution vector m. Reshape the output m from `linalg.lstsq` to get the 2x2 matrix M. **Print out** the matrix $M = \begin{bmatrix} m_{11} & m_{12} \\ m_{21} & m_{22} \end{bmatrix}$.

Problem 4: Panorama:
Panoramic photography is a technique that combines multiple images captured by a rotating camera into a single, wide photo. This is done by combining images based on their matching features through a process called image stitching. Given images named “**filed1.jpg**” through “**filed8.jpg**”, write an algorithm that stiches the images together to create a panorama. You can use the template provided in **“panorama_template.py”** and add your code as needed to complete the assignment. **Print out the entire code** in your report, which must contain the following steps:

- Load **all 8 images** and downsample them by a factor of 5 or more (to speed up the process and help with displaying the final result)
- For each consecutive image pair, detect and match **SIFT features** using **Low’s threshold of 0.7**
- Compute **homography** between the image pair using matched features and **RANSAC** algorithm
- Use computed homographies to warp all images onto one base image perspective using `cv2.warpPerspective()`. Combine the base and warped images and **display the resulting panorama image**.

Homework 3: Camera Geometry

Kaveh Fathian, Email: fathian@ariarobotics.com

Handout: 2024-10-11
Due: 2024-10-28, at 3:00 PM on Canvas

Note: you are **NOT allowed** to used any existing panorama and image stitching libraries, e.g., OpenCV's `stitcher.stitch(images).`

Problem 5: 3D reconstruction:
Given images named "**left.jpg**" and "**right.jpg**", write an algorithm that 3d reconstructs the matched SIFT features points. You can use the code template provided in **"twoview_template.py"** and add your code as needed to complete the assignment. **Print out the entire code** in your report, which must contain the following steps. The camera **intrinsic matrix** is given to you as

```
f, cx, cy = 1000, 1024, 768
K = np.array([[f, 0, cx], [0, f, cy], [0, 0, 1]])
```

- Load the images and extract and match **SIFT** features using **Lowe's threshold of 0.7**
- Find the **fundamental/essential matrix** using **RANSAC,** and **print the matrix**
- **Show** matched **inlier** feature points (based on the RANSAC results) connected by lines across the images

- Recover the relative **rotation** and **translation** between the camera views from the fundamental/essential matrix and **print the results**
- Reconstruct 3D points via triangulation, and **display** the resulting 3D points