Homework 09: Tow-View Geometry

Kaveh Fathian, Email: fathian@ariarobotics.com

Handout: 2025-10-29
Due: 2025-11-05, 11:59pm, on Canvas

General Instructions:

- You should solve the homework and submit your report **individually**. Identical submissions will receive a grade of zero.
- Getting help from others or checking your answers with other students (not the TAs) is okay and encouraged.
- Ask any questions on **Ed Discussion** (instead of emailing).
- **Before** the homework due date, TAs are strictly prohibited from **pre-grading** your homework. Do not expect the TAs to help you verify if your answers are correct or give you the problem solution.
- **After** the homework due date, if you do not know how to solve a problem, reach out to the TAs. They will walk you through the solution and help you understand it. Note that homework solutions will **not** be posted because some problems will be used in next year's class.
- **Exams** may contain questions related to homework, so make sure you learn how to solve the homework problems correctly.
- The deliverables are outlined for each problem, and you should carefully **follow the instructions**. Failing to follow instructions will result in **points being subtracted**.
- You will submit a **single PDF** file to Canvas as your homework report. The PDF must contain your **answers** and any requested **outputs** (e.g., printouts, snapshots of code, or GUIs). If requested, follow the instructions specified by the problem to provide your **code** (e.g., in a compressed .zip or .tar file) in addition to the PDF file.
- **Grading:** Each homework in this class will contribute **5pts** to your final grade (there will be 12 homework assignments, each 5pts, leading to 60pts for all assignments). A detailed grading **rubric** will be posted on **Canvas** after the homework due date. Any bonus points will be added to your overall course bonus points, which will be added to your final grade.
- **Late submission:** Late or missed submission will not be accepted and will receive a grade a zero. Any excused absence must be documented and disclosed to the instructor (extensions will be granted on a case-by-case basis). Three or more missed homework lead to an INC grade.

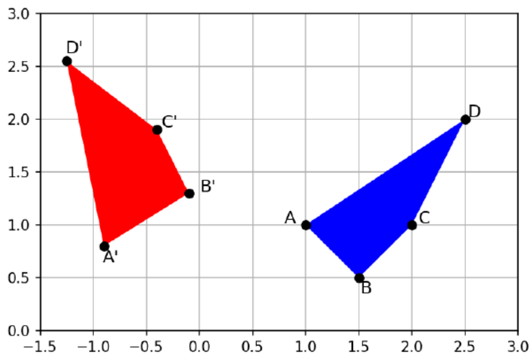
Homework 09: Tow-View Geometry

Kaveh Fathian, Email: fathian@ariarobotics.com

Handout: 2025-10-29

Due: 2025-11-05, 11:59pm, on Canvas

EXERCISE 1 (2.5pts) – Given a quadrilateral $ABCD$, the shape was transformed into another quadrilateral $A'B'C'D'$, as shown in the image below, via a 2x2 transformation matrix M.



$$A = (1, 1) \qquad A' = (-0.9, 0.8)$$
$$B = (1.5, 0.5) \qquad B' = (-0.1, 1.3)$$
$$C = (2, 1) \qquad C' = (-0.4, 1.9)$$
$$D = (2.5, 2) \qquad D' = (-1.25, 2.55)$$

More specifically, if $X = \begin{bmatrix} x \\ y \end{bmatrix}$ is a point on $ABCD$, its corresponding point $X' = \begin{bmatrix} x' \\ y' \end{bmatrix}$ on $A'B'C'D'$ is computed by multiplying the transformation matrix $M = \begin{bmatrix} m_{11} & m_{12} \\ m_{21} & m_{22} \end{bmatrix}$ as

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

We would like to approximate M using least squares.

- **Step 1:** Expand $M\,X = X'$ and rewrite it as a pair of linear equations. Report your answer by filling out each blank space "__" below with x, y, x', y', or 0.

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

- **Step2:** Using coordinates of the points A, B, C, D and their correspondence, you should obtain 8 equations from previous step (2 equations for each point). Stack all equations and represent them as multiplication of an 8x4 matrix Q and an 8x1 column vector b as

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

  Note that Q and b should have numerical values (not symbolic).

- **Step 3:** The problem is now formulated as an over-constrained system of linear equations, so m_{ij}'s can be found via least squares. Use `numpy.linalg.lstsq()` to take in the Q matrix and

Homework 09: Tow-View Geometry

Kaveh Fathian, Email: fathian@ariarobotics.com

Handout: 2025-10-29

Due: 2025-11-05, 11:59pm, on Canvas

b vector you computed above, and return the solution vector $m = \begin{bmatrix} m_{11} \\ m_{12} \\ m_{21} \\ m_{22} \end{bmatrix}$. Reshape the output vector m into the 2x2 matrix M.

Deliverables:
- Printout of your entire code.
- Answers for blank spaces in Step 1.
- Numerical values of matrix Q and vector b in Step 2.
- Matrix $M = \begin{bmatrix} m_{11} & m_{12} \\ m_{21} & m_{22} \end{bmatrix}$ in Step 3.

EXERCISE 2 (2.5pts) – Given the algorithm shown below, provide a **brief** explanation (in one or two sentences) for the function of the algorithm's line/segment as requested:
- What is matrix K is line 35?
- What type of image keypoint/descriptor is extracted in line 37?
- What is the Lowe's threshold used for matching keypoints in line 39?
- What are variables F and inlier_mask in line 44?
- What is E in line 46?
- What is the objective of line 47?
- What is the objective of lines 49-56, and what is output X?

Deliverables:
- Answer to all the questions above. Answer should be brief in one or two sentences.

Homework 09: Tow-View Geometry

Kaveh Fathian, Email: fathian@ariarobotics.com

Handout: 2025-10-29

Due: 2025-11-05, 11:59pm, on Canvas

```python
import numpy as np
import cv2 as cv

def get_keypoint(left_img, right_img):
    l_img = cv.cvtColor(left_img, cv.COLOR_BGR2GRAY)
    r_img = cv.cvtColor(right_img, cv.COLOR_BGR2GRAY)

    sift = cv.SIFT_create()
    key_points1, descriptor1 = sift.detectAndCompute(l_img, None)
    key_points2, descriptor2 = sift.detectAndCompute(r_img, None)

    return key_points1, descriptor1, key_points2, descriptor2


def match_keypoints(descriptor1, descriptor2):
    FLANN_INDEX_KDTREE = 1
    index_params = dict(algorithm=FLANN_INDEX_KDTREE, trees=5)
    search_params = dict(checks=50)

    flann = cv.FlannBasedMatcher(index_params, search_params)
    matches = flann.knnMatch(descriptor1, descriptor2, k=2)

    good_matches = []
    for m, n in matches:
        if m.distance < 0.6 * n.distance:
            good_matches.append(m)

    return good_matches


img1 = cv.imread('data/left.png')
img2 = cv.imread('data/right.png')

f, cx, cy = 48.5, 48.5, 48.5
K = np.array([[f, 0, cx], [0, f, cy], [0, 0, 1]])

key_points1, descriptor1, key_points2, descriptor2 = get_keypoint(img1, img2)

good_matches = match_keypoints(descriptor1, descriptor2)

pts1 = np.float32([key_points1[m.queryIdx].pt for m in good_matches]).reshape(-1, 1, 2)
pts2 = np.float32([key_points2[m.trainIdx].pt for m in good_matches]).reshape(-1, 1, 2)

F, inlier_mask = cv.findFundamentalMat(pts1, pts2, cv.FM_RANSAC, 2, 0.99, maxIters=1000)

E = K.T @ F @ K
positive_num, R, t, positive_mask = cv.recoverPose(E, pts1, pts2, K, mask=inlier_mask)

P0 = K @ np.eye(3, 4, dtype=np.float32)
Rt = np.hstack((R, t))
P1 = K @ Rt
pts1_inlier = pts1[inlier_mask.ravel() == 1]
pts2_inlier = pts2[inlier_mask.ravel() == 1]
X = cv.triangulatePoints(P0, P1, pts1_inlier, pts2_inlier)
X /= X[3]
X = X.T
```