CSE 152 Assignment 1
Introduction to Computer Vision
Winter 2019
Due Mon, Feb 04, 9pm PST

Instructions :

- Attempt all questions.
- Please comment all your code adequately.
- Turn-in a PDF for the written report and a zip-folder for the code. The report should be called `LastName_FirstName.pdf` and similarly the zip folder should be `LastName_FirstName.zip`.
- Include all relevant information such as text answers, output images and main code snippets in the PDF.
- Submit your homework by Gradescope. Remember to correctly select pages for each answer on Gradescope to allow proper grading.

1 Camera Matrices and Rigid-Body Transformations

Consider a world coordinate system W, centered at the origin $(0, 0, 0)$, with axes given by unit vectors $\widehat{\mathbf{i}} = (1, 0, 0)^\top$, $\widehat{\mathbf{j}} = (0, 1, 0)^\top$ and $\widehat{\mathbf{k}} = (0, 0, 1)^\top$. We use a notation where boldfaces stand for a vector and a hat above a boldface letter stands for a unit vector.

(a) Consider another coordinate system, with unit vectors along two of the orthogonal axes as: $\widehat{\mathbf{i}}' = (0.9,\ 0.4,\ 0.1\sqrt{3})^\top$ and $\widehat{\mathbf{j}}' = (-0.41833,\ 0.90427,\ 0.08539)^\top$. Find the unit vector, $\widehat{\mathbf{k}}'$, along the third axis orthogonal to both $\widehat{\mathbf{i}}'$ and $\widehat{\mathbf{j}}'$. Is there a unique such unit vector? If not, choose the one that makes an acute angle with $\widehat{\mathbf{k}}$. **[2 points]**

(b) Find the rotation matrix that rotates any vector in the $(\widehat{\mathbf{i}}, \widehat{\mathbf{j}}, \widehat{\mathbf{k}})$ coordinate system to the $(\widehat{\mathbf{i}}', \widehat{\mathbf{j}}', \widehat{\mathbf{k}}')$ coordinate system. **[2 points]**

(c) What is the *extrinsic parameter matrix* for a camera at a displacement $(-1, -2, -3)^\top$ from the origin of W and oriented such that its principal axis coincides with $\widehat{\mathbf{k}}'$, the x-axis of its image plane coincides with $\widehat{\mathbf{i}}'$ and the y-axis of the image plane coincides with $\widehat{\mathbf{j}}'$? **[3 points]**

(d) What is the *intrinsic parameter matrix* for this camera, if its focal length in the x-direction is 1050 pixels, aspect ratio is 1, pixels deviate from rectangular by 0 degrees and principal point is offset from the center $(0, 0)^\top$ of the image plane to the location $(10, -5)^\top$? **[3 points]**

(e) Write down the projection matrix for the camera described by the configuration in parts **(c)** and **(d)**. **[3 points]**

(f) Consider a plane, orthogonal to $\widehat{\mathbf{k}}$, at a displacement of 2 units from the origin of W along the $\widehat{\mathbf{k}}$ direction. Consider a circle with radius 1, centered at $(0, 0, 2)^\top$ in the coordinate system W. We wish to find the image of this circle, as seen by the camera we constructed in part **(e)**. The following questions need programming (use Python) and the code for each part should be turned in along with any figures and answers to specific questions. Explain your variable names (with comments). Feel free to supply any additional description or explanation to go with your code.

- **(i)** Compute 10000 well-distributed points on the unit circle. One way to do this is to sample the angular range 0 to 360 degrees into 10000 equal parts and convert the resulting points from polar coordinates (radius is 1) to Cartesian coordinates. Display the circle, make sure that the axes of the display figure are equal. **[2 points]**
- **(ii)** Add the z coordinate to these points, which is 2 for all of them. Make all the points homogeneous by adding a fourth coordinate equal to 1. **[1 point]**
- **(iii)** Compute the projection of these homogeneous points using the camera matrix computed in part **(e)**. Convert the homogeneous projected points to 2D Cartesian points by dividing out (and subsequently discarding) the third coordinate of each point. **[2 points]**
- **(iv)** Plot the projected 2D points, again ensure that the axes of your plot are equal. What is the shape of the image of a circle? **[2 points]**

2 Epipolar Geometry

In this problem, we will verify the claim in class that for any point in one image of a stereo pair, the corresponding point in the other image lies on the epipolar line. The final output for this problem should be a Python function, contained in a file `testepipolar.py`. Here is what the function should look like:

```
# You may add other libraries or files as well
import matplotlib.pyplot as plt
import numpy as np

def testepipolar(xl, xr, l, r):
    # Compute the fundamental matrix
    F = computeF(xl, xr)

    # Compute the epipolar lines in the left image
    leftlines = ??
    # Compute the epipolar lines in the right image
    rightlines = ??

    # Draw epipolar lines on the left image
    drawLine(leftlines[:,1], l, 'yellow')
    drawLine(leftlines[:,3], l, 'cyan')
    drawLine(leftlines[:,5], l, 'green')
    drawLine(leftlines[:,7], l, 'magenta')
    drawLine(leftlines[:,9], l, 'red')
    plt.show()

    # Draw epipolar lines on the right image
    drawLine(rightlines[:,1], r, 'yellow')
    drawLine(rightlines[:,3], r, 'cyan')
    drawLine(rightlines[:,5], r, 'green')
    drawLine(rightlines[:,7], r, 'magenta')
    drawLine(rightlines[:,9], r, 'red')
    plt.show()
```

The overall code must be contained in a script called `script.py` which might look like:

```
# You may add other libraries or files as well
import matplotlib.pyplot as plt
import numpy as np
from imageio import imread

# Read in the input images
left = 'blocks1.gif'
right = 'blocks2.gif'
l = imread(left)
r = imread(right)

# Pick the corresponding corners
xl = clickPoints(l, 10, 1, 1)
xr = clickPoints(r, 10, 1, 1)

F = testepipolar(xl, xr, l, r)
```

We will now have a look at the individual components of the above code.

(a) Use the provided function in `clickPoints.py` to click 10 points from the image left and the corresponding 10 points in the image right. For the blocks data, left is `blocks1.gif` while for the desk data, left is `desk1.gif`. The other image of the stereo pair is right. Choose the 10 points as shown in Figure 1 for the blocks and as shown in Figure 2 for the desk data. Be very careful to click the corners as accurately as you can.
Turn-in: images with the clicked points, called `lblock-clicks.png`, `rblock-clicks.png`, `ldesk-clicks.png`, `rdesk-clicks.png`. **[2 points]**

(b) Write a function, called `computeF`, to compute the fundamental matrix for an image pair. The function takes as input two $2 \times n$ arrays, containing the points clicked in part (a) in the left and right images, respectively. Here, n is the number of points clicked in each image. Output is the 3×3 fundamental matrix.
Turn-in: the estimated fundamental matrix. **[7 points]**

(c) Using the fundamental matrix, compute the epipolar lines in each image that correspond to points in the other image. Store the homogeneous representations of the epipolar lines in the left image (which correspond to corners in the right image) in the 3×10 matrix *leftlines* and those for the right image in *rightlines*. Use the provided function in `drawLine.py` to draw the epipolar line in each image for the points 1, 3, 5, 7 and 9.
Turn-in: images with the plotted epipolar lines, called `lblock-lines.png`, `rblock-lines.png`, `ldesk-lines.png`, `rdesk-lines.png`. **[4 points]**

(d) Compute the epipoles in the left image and the right image. Call them *el* and *er*, respectively.
Turn-in: the epipoles in the left and right images, expressed in Cartesian coordinates on the image plane (not in homogeneous coordinates). **[2 points]**

(e) Provide a written description of your procedure to compute the fundamental matrix, epipolar lines and epipoles. **[5 points]**

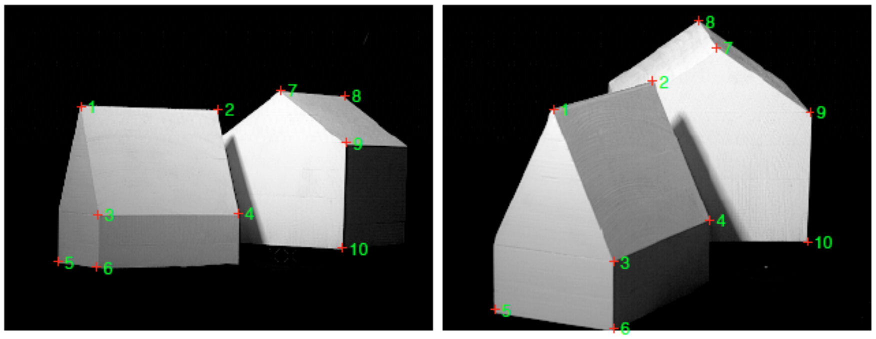

Figure 1: Points to be clicked for blocks data.

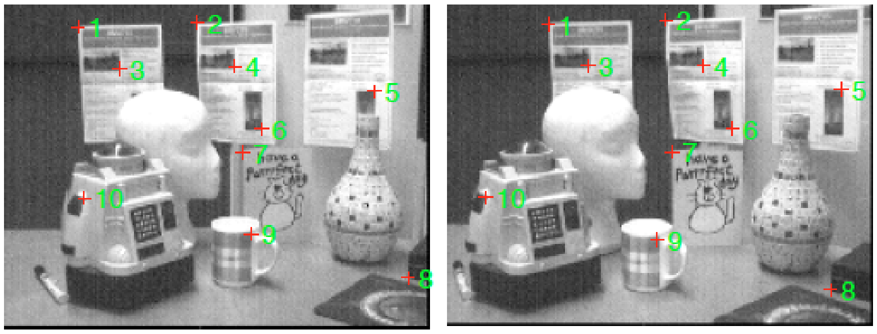

Figure 2: Points to be clicked for desk data.

3 Feature detection, matching and RANSAC

We will now use SIFT to detect and match features, then use RANSAC to eliminate outliers that do not conform to a fundamental matrix model. If you wish to try SIFT matching yourself, you may obtain code for SIFT in the following ways:

- Binaries from original implementation: `http://www.cs.ubc.ca/~lowe/keypoints/`
- VLFeat implementation: `http://www.vlfeat.org/index.html`
- OpenCV: `http://docs.opencv.org/trunk/index.html`

Alternatively, for this assignment, we are providing matched SIFT points in text files that you may simply read as input.

(a) Use SIFT to detect and match features in the two images `road1.png` (left image) and `road2.png` (right image). Visualize the matched features by drawing lines between the left and right images. In Python, you may use the provided `plotmatches.py` code. If you are using the provide SIFT matches, the data in `points1.txt` are the keypoints in the left image and in `points2.txt` are the keypoints in the right image. Each row has the x and y coordinates for a point. Corresponding rows in the two files are the matching points. **[2 points]**

(b) Estimate the fundamental matrix using the SIFT matches. Plot the epipolar lines for 5 randomly selected keypoints. **[3 points]**

(c) Use RANSAC with the 8-point algorithm to remove outliers and re-estimate the fundamental matrix with the inliers. Visualize the inlier matches by drawing lines between the left and right images. Plot the epipolar lines for 5 randomly selected keypoints. **[8 points]**

(d) Conceptually, can you guess approximately where the epipole should lie for the two images above? Explain your reasoning. Do your epipolar lines above match that intuition? **[3 points]**

(e) *Challenge question:* Perhaps you have observed that estimation of fundamental matrices is quite sensitive, for example, to how accurately the corresponding points are localized. Can you observe the data matrix (the "$\mathbf{A}$ matrix") used for fundamental matrix estimation and determine a possible reason for the estimation to be ill-conditioned? Can you suggest a possible remedy? **[4 points]**
Hint: The intrinsic calibration matrix for these images is approximately

$$\mathbf{K} = \begin{bmatrix} 1000 & 0 & 700 \\ 0 & 1000 & 250 \\ 0 & 0 & 1 \end{bmatrix}.$$