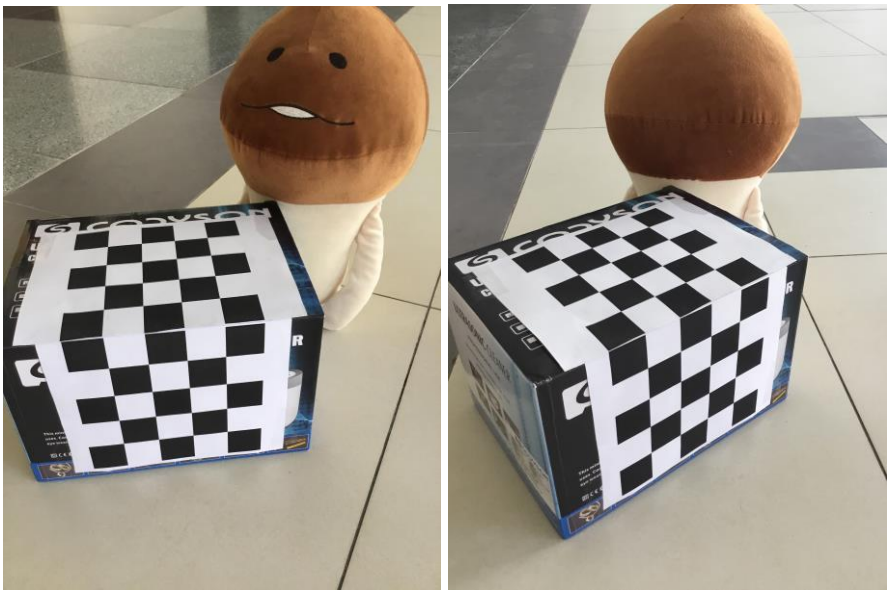
CS655000 Computer Vision Homework 2
Camera Calibration & Homography Transformation

- Due Date: 11/8(Fri.) 23:59
- Use Python to complete the homework

Part 1. Camera Calibration (50%)

Perform camera calibration from a set of 3D points & 2D points correspondences in the chessboard images. You can find these images in the zip file.



You can use “clicker.py” provided by TA or implement by yourself to get the corresponding 2D points and 3D points in “Point3D.txt”, totally 36 pairs, for each image. You should also save your 2D points.

A. Compute the projection matrix from a set of 2D-3D point correspondences by using the least-squares (eigenvector) method for each image.

B. Decompose the two computed projection matrices from **(A)** into the camera intrinsic matrices K, rotation matrices R and translation vectors t by using the Gram-Schmidt process.
Any QR decomposition functions are allowed. The bottom right corner of intrinsic matrix K should be normalized to 1. Also, the focal length in K should be positive.

C. Re-project 2D points on each of the chessboard images by using the computed intrinsic matrix, rotation matrix and translation vector. Show the results (2 images) and compute the point re-projection root-mean-squared errors.

D. Plot camera poses for the computed extrinsic parameters (R, t) and then compute the angle between the two camera pose vectors.

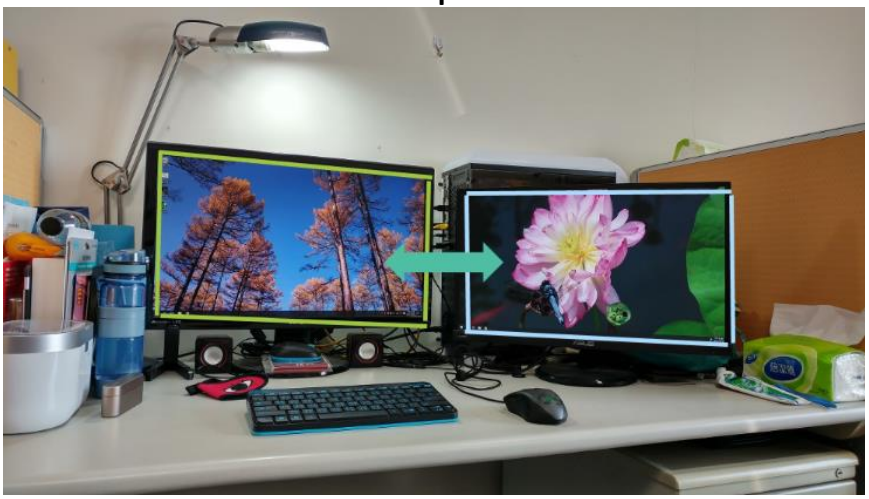
E. **(Bonus) (10%)** Print out two "chessboard.png" in the attached file and paste them on a box. Take two pictures from different angles. For each image, perform the steps above (A ~ D).

F. **(Bonus) (10%)** Instead of mark the 2D points by hand, you can find the 2D points in your images automatically by using corner detection, hough transform, etc.

Part 2. Homography transformation (50%)

You are required to utilize techniques of homography transformation and both backward and forward warping to generate two interesting image pairs.

Example 1



Example 2

A. Shoot three images A, B and C. Image A has to contain two objects. Image B and C should contain one object separately. Like the images shown above. (3 images)

B. Compute homography transformation between the two objects in image A. Use both **backward** and **forward** warping to switch them, like what example 1 shows. (2 images)

C. Compute homography transformation between the object in image B and the object in image C. Use both **backward** and **forward** warping to switch them. Example 2 gives some illustration. (4 images)

D. Discuss the difference between forward and backward warping based on your results.

Reminder

- **You should not use any function (ex. cv2.findHomography, cv2.warpPerspective) which can generate the result directly in each step.**
- Your code should display or output your results so that we can judge if your code works correctly.
- You should provide a **README** file about your execution instructions. **If your code is not executable, you will get zero point**.
- Your report should also contain output images.
- Please compress your **code**, **input images**, **result images**, **report** and **README** in a zip file named HW2_{Student-ID}.zip and upload it to iLMS. The wrong filename will get a 50 points deduction.
- If you encounter any problem, let's **discuss on iLMS forum instead of email**.
- Reference of problem 1