# Project 4: Camera Calibration and Stereo Vision

(Assignment is developed by Henry Hu, Grady Williams, and James Hays based on a similar project by Aaron Bobick and modified by Elsayed Hemayed)

## Overview

The goal of this project is to introduce you to camera and scene geometry. Specifically we will estimate the camera projection matrix, which maps 3D world coordinates to image coordinates, will calibrate our own camera and reconstruct a simple object using stereo vision technique. The camera projection matrix is estimated using point correspondences. To estimate the projection matrix—intrinsic and extrinsic camera calibration—the input is corresponding 3d and 2d points. You will start first to estimate the projection matrix using some given ground truth data. Then you will apply the developed technique to calibrate stereo cameras then to find a matching pair in the stereo images and to estimate their corresponding 3D point.

## Details and Starter Code

This project consists of three parts:

1. Estimating the projection matrix,
2. Calibrate your own camera, and
3. Reconstruct a simple object (estimate the 3D coordinates of object interest points).

**Part I: Camera Projection Matrix**

Given known 3D to 2D point correspondences, how can we recover a projection matrix that transforms from world 3D coordinates to 2D image coordinates? Our task is to find the projection matrix M using a system of linear equations and to find the least squares regression solution for these camera matrix parameters, given correspondences between 3D world points and 2D image points.

To help you validate M, we provide evaluation code which computes the total "residual error" between the projected 2d location of each 3d point and the actual location of that point in the 2d image. The residual is the Euclidean distance in the image plane (square root of the sum of squared differences in u and v). This should be very small—in the order of a pixel. We also provide a set of “normalized points” in the files ./pts2d-norm-pic_a.txt and ./pts3d-norm.txt. The starter code will load these 20 corresponding normalized 2D and 3D points. If you solve for M using all the points, you should receive a matrix that is a scaled equivalent of the following matrix:

$$M = \begin{pmatrix} -0.4583 & 0.2947 & 0.0139 & -0.0040 \\ 0.0509 & 0.0546 & 0.5410 & 0.0524 \\ -0.1090 & -0.1784 & 0.0443 & -0.5968 \end{pmatrix}$$

Given this matrix, we can project 3D points in the world onto our camera plane. For example, this matrix will take the normalized 3D point < 1.2323, 1.4421, 0.4506, 1.0 > and project it to 2D image < u, v > of < 0.1419, −0.4518 > (after converting the homogeneous 2D point < us, vs, s > to its nonhomogeneous version by dividing by s).

Once we have an accurate projection matrix M, it is possible to tease it apart into the more familiar and more useful matrix K of intrinsic parameters and matrix [R | T] of extrinsic parameters. For this project we will only ask you to estimate one particular extrinsic parameter: the camera center in world coordinates. Let us define M as being made up of a 3x3 matrix called Q, with 4th column m_4:

$$M = (Q \mid m_4)$$

The center of the camera C could be found by:

$$C = -Q^{-1} m_4$$

If we use the normalized 3D points and M given above, we would receive camera center:

$$C = < -1.5125, -2.3515, 0.2826 >$$

We've also provided a visualization which will show the estimated 3d location of the camera with respect to the normalized 3d point coordinates.

## PART II: Calibrate Your Own Cameras using OpenCV

The purpose of this part is to calibrate your own camera(s) as you did in the lab.

1. Design an appropriate calibration pattern. Put your calibration pattern in front of the camera, adjust your camera focus and zoom as well as the light to get a clear image of the pattern.
2. Using OpenCV `calibrateCamera()` to calibrate your camera. You may use the code provided in https://github.com/spmallick/learnopencv/tree/master/CameraCalibration
3. Validate the computed projection matrix as done in PART I. You may need to review and repeat your process to ensure getting good calibration.
4. Repeat this process for another camera (to form stereo camera).
5. Compute the Essential matrix. You may use OpenCV `stereoCalibrate()` to calibrate the two cameras and compute the Essential Matrix

**Hint**: for good results, put the calibration pattern in front of the stereo camera and capture the pattern image from both cameras then move the pattern and put your object in its place and capture its image from both cameras.

**PART III: Depth Measurement**

The purpose of this part is to use your own two cameras to measure the depth of 3D points. You will need to arrange with your colleague to use each other cameras and find simple objects that are easy to match.

1. Calibrate your cameras as done in PART II.
2. Capture the image of your selected object using your cameras. Do not move your cameras from the calibration position.
3. Perform SIFT detection and matching so you can identify a set of matched points in the two images.
4. Improve your results by checking the epipolar constraint using the Essential matrix computed in PART II.
5. Reconstruct the matched points. You can use the OpenCV `triangulatePoints()`.
6. Draw the reconstructed points in the 3D coordinates and discuss the reconstruction accuracy.

## Evaluation and Visualization

For part I, we have provided expected output (matrix M and camera center C). These are numerical estimates so we won't be checking for exact numbers, just approximately correct locations.

For part II, you will be evaluated based on your estimate of the projection matrix. You can test how good your estimate of the projection matrix is by computing the residual error and by visualizing the projected 2d location of each 3d point vs the actual location of that point in the 2d image.

For Part III, you can visualize the reconstructed set of points and rotate them in space to ensure that they have proper relative location. You can also measure the length between the main corners in the object and compare it against the corresponding lengths of the reconstructed points.

## Data

We provide 2D and 3D ground truth point correspondences for the base image pair (pic_a.jpg and pic_b.jpg). Other images will not have any ground truth correspondences.

## Write up

For this project, and all other projects, you must do a project report in HTML. In the report you will describe your algorithm and any decisions you made to write your algorithm a particular way. Discuss any extra credit you did and show what contribution it had on the results (e.g., performance with and without each extra credit component).

## Rubric

- +2 pt: Correctly setting up the system of equations for the least squares regression for the projection matrix. And correctly solving for the projection matrix and correctly solving for the camera center, and (R, T).
- +3 pt: Correctly calibrate your stereo camera and compute the essential matrix.
- +3 pt: Correctly reconstruct set of points of an object of your choice captured by your own cameras.
- +2 pts: Write up.
- +2 pts: Extra Credit
- -0.5*n pts: Lose 0.5 points for every time (after the first) you do not follow the instructions for the hand in format.

## Extra Credit

- 0.5 pts: Use another feature detector and descriptor and compare its accuracy and computation time against SIFT.
- 1 pts: Compute the camera orientation for both cameras in PART II and use the cameras center and orientation to draw a schematic diagram to show both cameras w.r.t. the object.
- 2 pts: Use three cameras instead of two.

## **Deliverables**

Upload a Zip file named as "CIE552_Spring2022_Proj4_YourName.zip" to Classroom. The file should include your code, new data (if any), ReadMe file, results webpage folder, and report.
Hint: For its large size, do not include the original data in your submission.