Computer Vision: Spring 2023
Assignment - 1: Camera Calibration

1. Direct Linear Transform (DLT)

Perform DLT based camera calibration on the ***"calib-object.jpg"*** image. Choose at least 20 points on the chessboard (at least 10 on each plane), and find their corresponding points in the image (**Note**: Refer to ***"calib-object.jpg"*** to see the alignment of the world coordinate system and world origin). This will give you a mapping between your world points and image points. Now, using these correspondence between world points and image points, you are required to do the following:

A. Estimate Camera Projection (P) Matrix.
B. Estimate the reprojection error (i.e. the error between where your world points map to in the image according to your estimated Projection Matrix, and where the world points actually map to)
C. Estimate the Camera Intrinsic Matrix (K), Rotation Matrix (R) and Projection Center (C). [Note that this should be trivial once you have estimated the **P** matrix]
D. Find out where your world points map to in the image according to the projection matrix (P) you obtained above i.e., calculate x_i = P@X_i (where "@" represents matrix multiplication). Now, compute a wireframe of the object by drawing straight lines between the computed points x_i and overlay this wireframe on the image (you are allowed to use in-build OpenCV functions for this task. We'll also try to share a boilerplate code to draw the wireframe)

Note that you are NOT allowed to use in-built OpenCV functions for anything other than reading, displaying and saving images in this question for tasks (A), (B) and (C). You can use in-built open-CV functions for task (D)

Do the above tasks ((A), (B), (C) and (D)) for the following settings:

- **(Experiment-1)**: Assume scale of each chessblock = 28mmx28mm and use all the 20 (or more) points which you have marked
- **(Experiment-2)**: Assume scale of each chessblock = 2800mmx2800mm and use all the 20 (or more) points which you have marked
- **(Experiment-3)**: Assume scale of each chessblock = 28mmx56mm and use all the 20 (or more) points which you have marked
- **(Experiment-4)**: Assume scale of each chessblock = 28mmx28mm and use only 10 points out of the 20 (or more) which you have marked (Make sure that all these 10 points do **not** lie on the same plane)

- **(Experiment-5)**: Assume scale of each chessblock = 28mmx28mm and use only 6 points out of the 20 (or more) which you have marked (Make sure that all these 6 points do **not** lie on the same plane)
- **(Experiment-6)**: Assume scale of each chessblock = 28mmx28mm and use only 10 points out of the 20 (or more) which you have marked, and choose points such that all these 10 points lie on the same plane
    - **Q.** Are these results similar to what you got for earlier experiments? If not, why is that the case? Is there any work around?

Note: Make sure to carefully observe the difference in the outputs in different settings. Write in brief (1-2 lines are enough) explaining the difference in results of each experiment.

Hints:

- To identify coordinates of image points, you can open the image in a matplotlib window in your local PC and then hover over the image (the coordinates are visible on the top right corner) in the matplotlib window. Note that this is just one possible method to do it, better methods might exist. Some tools might be available on the internet also
- Since you are required to report the outputs for different settings, make sure you use functions and other good coding practices wherever relevant
- Your results won't be perfect. That's fine. As long as your code is correct and your results aren't horribly wrong (which they shouldn't be if your code is correct), it's fine.

Resources for DLT:

- **High-level overview**
    - Direct Linear Transform - 5 Minutes with Cyrill
- **Detailed Explanation**
    - Watch this video first: Linear Camera Model | Camera Calibration, this explains the Linear Camera Model and explains the related equations
    - Then watch this video: https://www.youtube.com/watch?v=3NcQbZu6xt8. This explains how to solve those equations.

---

2. RANSAC

Implement a RANSAC-based variant of DLT and report results for **Experiment-1** mentioned in the first section.

More Resources:

- https://in.mathworks.com/discovery/ransac.html
- Dealing with Outliers: RANSAC | Image Stitching

---

3. Triangulation

Triangulation is the process of computing the position of a point in a 3D space given its image and the camera matrices for the two views.

Geometric Solution for triangulation:

- Consider a point (say point-1) on image-1 and find a line passing through this point and the center of the camera (same as the center of projection). Let this line be L-1.
- Find the point in image-2 which corresponds to point-1 in image-1. Find a line - L2 which passes through this point and the center of projection of this image.
- Find the point where L1 and L2 intersect. Ideally, L1 and L2 should intersect at the world point which was imaged. However, due to noise in the process, this might not occur. Please read this for a detailed explanation.

The geometric solution does not take into account the uncertainties and is not statistically optimal.

More resources:

- Triangulation for Image Pairs (Cyrill Stachniss) (Till 16:46 timestamp)

Statistically Optimal Solution for triangulation:

- Read: https://www.cs.cmu.edu/~16385/s17/Slides/11.4_Triangulation.pdf

In this section of the assignment, you will assess the quality of your calibration. We would reproject the image of points back to the 3D world and calculate the error.

Task to perform

- Perform DLT on both images of the cube: *"cube-01.jpg" and "cube-02.jpg"* to get projection matrices for the two cameras.
- Consider the image coordinates of a world point used for DLT in both images.
- Triangulate these two image coordinates to find the 3D coordinate of the world point.
- Calculate the euclidean error between the predicted world coordinate and the original world point.

You can do **either statistical optimal or geometric solution** to solve the assignment, but viva questions will be asked from both. **You are NOT allowed to use any OpenCV functions in this task**

---

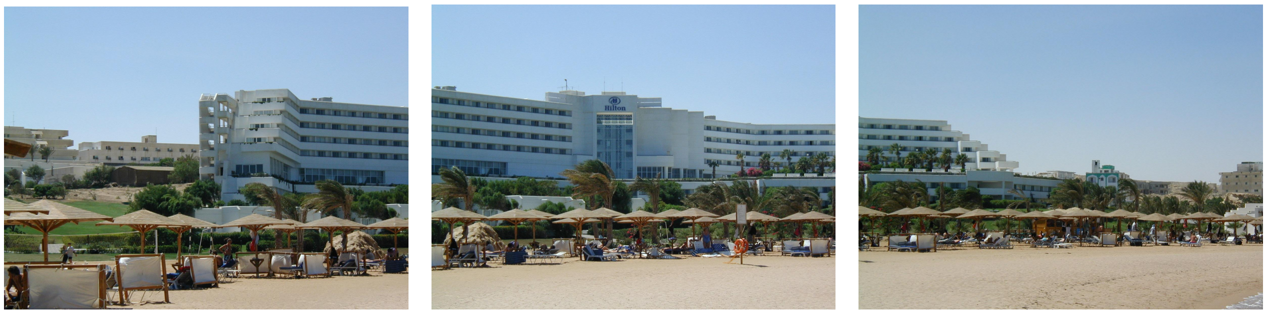
4. Image Stitching

Image stitching is a process of creating a panorama using the combination of multiple partially overlapped images. A basic image-stitching algorithm involves the following steps:

1. Extract features and perform feature matching between two overlapping images.
2. Estimate the homography matrix between them
3. Transform one of the images to the other reference frame using the homography matrix
4. Stitch the two images together
5. Repeat the same procedure to create a single panorama in case there are more than two images.

Given a set of images, you have to produce a single mosaic/panorama using them. You can use any feature detection and description technique (like SIFT, SURF, etc.) to find matches across two partially overlapping images. **You can use OpenCV functions for the individual steps mentioned above, but you are not allowed to use `cv2.Stitcher`**. Report the results from the given set of data.

Individual Images



Stitched Panorama