Assignments

Assignment 1: Camera geometric calibration

Instructions: Install Python (instructions for Windows) and OpenCV (instructions for Windows). Learn how to use OpenCV's geometric-camera calibration. For instance, read the OpenCV book (Chapter 7), read the OpenCV documentation or go directly to this tutorial. The offline phase is carried out only once for each of three runs. The online phase for each new image/frame from your webcam (or test image). You can also use C++ but realize your work can be used in Assignment 2, which is in Python.

- Offline phase:

1. Print this image (or use it on a tablet). Measure the size of each cell and figure out where you have to use this in your code. Not properly doing this, will not give you the full points
2. While there is an OpenCV function that can find chessboard corners for you, this function sometimes fails. Therefore, you need to implement an interface that asks the user to manually provide the four corner points and that linearly interpolates all chessboard points from those coordinates. Tutorial for getting mouse click coordinates The output of your interface should be similar to that of the OpenCV function that finds the corner points automatically. Of course, a linear interpolation is not ideal if there are significant perspective effects. That's okay, and there is a choice task to address this issue.
3. With a camera or webcam, take 25 training images of the chessboard in different positions and orientations (rotated, tilted) in the image. Of these 25 images, there should be at least 5 images with the chessboard in view but where OpenCV could not detect the corners automatically. These images require manual annotation of the corner points. Take a final test image that has the chessboard tilted significantly, and close to the image border. For this image, the corner points have to be found automatically.
4. Implement the geometric-camera calibration using OpenCV functions (only for this assignment, you are allowed to use code available on the internet). Calibrate your camera (geometrically) using the training images from step 3. Make sure the function you use will **not** assume that the camera center is fixed but is estimated and ensure that the function will allow you to have different focal lengths in horizontal and vertical directions (check the flags). You will do three runs of calibration.
    1. Run 1: use all 25 training images (including the images with manually provided corner points).
    2. Run 2: use only ten images for which corner points were found automatically.
    3. Run 3: use only five out of the ten images in Run 2.
5. In each run, you will calibrate the camera. After calibration, you will need the camera intrinsics (or cameraMatrix) and the camera extrinsics (for Assignment 2).

- Online phase:

1. For each run, take the test image and draw the world 3D axes (XYZ) with the origin at the center of the world coordinates, using the estimated camera parameters. Also draw a cube which is located at the origin of the world coordinates. You can get bonus points

for doing this in real time using your webcam. See the example images below.

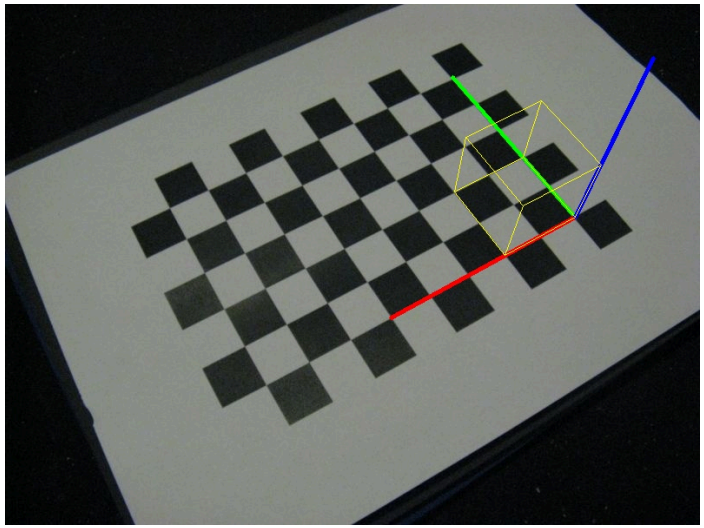
2. Draw a polygon that covers the top side of the cube (check the fillConvexPoly function). The color of the polygon should change with the postion and orientation of the center of the top plane, relative to the camera. Specifically:
    1. The intensity of the color (v-value in HSV color space) should reflect the distance to the camera. Linearly scale the value of the intensity between 255 when the distance is 0, to a value of 0 when the distance is 4 meters or more.
    2. The hue-and saturation components of the color should reflect the relative orientation of the top plane towards the camera. Saturation is encoded by the relative in-plane rotation. So when the chessboard is parallel to the camera, it should 255. When the chessboard is tilted away from the camera plane for 45 degree or more, the saturation value should be 0. Scale the saturation values (255-0) linearly with the angle (0-45).
    3. Hue is encoded by the relative orientation of the

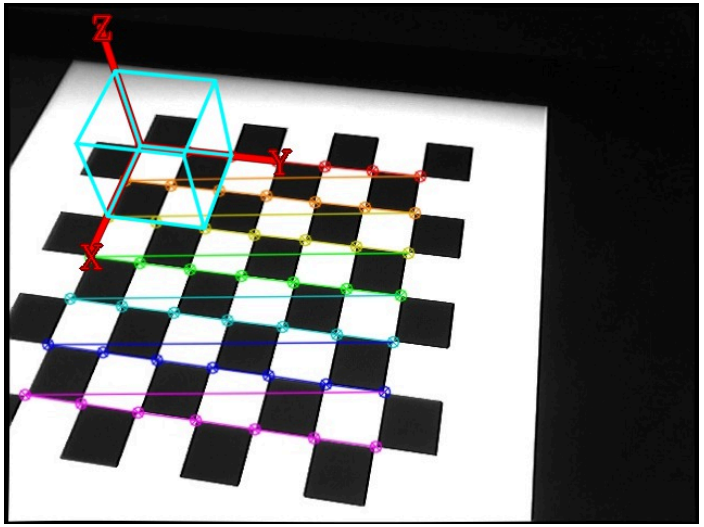
Code: You can develop a script for the online and offline phase independently or combined. In your code, write comments at the beginning of every function (about the purposes of the function). This is compulsory.

Report: Your report (infomcv_assignment_1_report_template.docx) should be around 1 page (up to 2 is allowed) and contain:

1. For each of the three runs, the intrinsic camera matrix and an example screenshot with axes and cube. Provide the explicit form of camera intrinsics matrix K. This can also be done by an OpenCV-function, figure out which one. Also state the resolution of your (training and test) images.
2. A brief explanation of how the estimation of each element in the intrinsics matrix changes for the three runs. What can you say about the quality of each run?
3. A brief mention of the choice tasks that you've done, and how you implemented them. For some tasks, provide the requested information.

Examples of the output:





Grading: The maximum score for this assignment is 100 (grade 10). The assignment counts for 10% of the course grade. You can get 70 regular points and max 30 points for chosen tasks:

- Offline: calibration stage: 15
- Offline: manual corner annotation interface when no corners are found automatically: 15
- Online: stage with cube drawing: 20
- Reporting: 10
- Test images with cube correct and accurate: 10
- CHOICE: real-time performance with webcam in online phase: 10
- CHOICE: iterative detection and rejection of low quality input images in offline phase: 10. Check for a function output that could provide an indication of the quality of the calibration.
- CHOICE: improving the localization of the corner points in your manual interface: 10. Make the estimation of (a) the four corner points or (b) the interpolated points more accurate.
- CHOICE: any animation/shading/etc. that demonstrates that you can project 3D to 2D: 10. There needs to be explicit depth reasoning so lines/vertices further away should not overlap nearer ones.
- CHOICE: implement a function that can provide a confidence for how well each variable has been estimated, perhaps by considering subsets of the input: 10
- CHOICE: implement a way to enhance the input to reduce the number of input images that are not correctly processed by findChessboardCorners, for example by enhancing edges or getting rid of light reflections: 10
- CHOICE: produce a 3D plot with the locations of the camera relative to the chessboard when each of the training images was taken: 10
- CHOICE: use your creativity. Check with Ronald Poppe for eligibility and number of points (via Teams or email).

Submission: Submit a zip through Blackboard with

- Code (+ project files) but no libraries or images (max. submission size is 20MB)
- Report (1 page)

Deadline: Sunday, February 18, 2024, at 23.00. For questions about the assignment, use the INFOMCV 2024 Teams (Assignment channel). If you found that your assignment partner did not work properly, notify Ronald Poppe as soon as possible.

Assignment 2: Voxel-based 3D reconstruction

This document contains a detailed description of Assignment 2, the data and the provided Python code used to visualize a voxel reconstruction. Use the code as a basis of your own program. The main goals of this assignment are (1) to calibrate four cameras (intrinsics and extrinsics), (2) to do background subtraction for each view, and (3) to implement the silhouette-based voxel reconstruction algorithm.

Download data + example code

- Download data (51MB)
- Download example code framework: https://github.com/dmetehan/Computer-Vision-3D-Reconstruction.git