Project 1

Calibrating Camera and Superimposing an Object on the Calibration Pattern

Domagoj Korais and Marco Zullich

Problem Statement

The problem may be split into two logical steps:

1. Camera calibration

Based upon a set of pictures from the same camera depicting a common coplanar calibration pattern, compute the intrinsic and extrinsic parameters of the camera and its pose from the correspondences between 3D coordinates of the pattern’s points of interest and the corresponding 2D coordinates of such points in the image reference frame.

The task is ran twice, first without, then with compensation for radial distortion.

1. Superimposition of an object to the calibration plane

Now that the parameters are known, they can be used to project points and solids of arbitrary 3D coordinate into any of the previous calibration images.

Approach

Task 1 (camera calibration) is carried out using Zhang’s method for homography estimation. The calibration object is a white/black 2D checkerboard with 35 intersections (7 columns x 5 rows) stuck on a paperback cardboard. All checkers within the object are square.

As far as radial distortion is concerned, after the estimation of the homography and calculation of the intrinsic () and extrinsic () parameters, we estimate distortion coefficients up to the 3rd order, we compensate for it refining the 3D-2D correspondences between checkerboard intersections in real world and within the images and re-running Zhang’s procedure iteratively.

After obtaining , we have a rule for transforming any point within the world’s reference frame (whose origin is fixed at the upper-left intersection of the checkerboard) into the 2D coordinates of any one of our calibration pictures, and hence we can superimpose any (virtual) object of our choice to our images, thus carrying out task 2.

Implementation

We opted in favor of implement our program in Python, using mainly two libraries:

* OpenCV
* NumPy

The former in particular uses very efficient methods to quickly carry out some common computer vision and image processing tasks.

The code is contained within the folder "Scripts", which contains two Python libraries:

* CameraCalibration.py is a library that encapsulates a number of methods constructed mainly over built-in OpenCV routines. It's designed to carry out task 1. Though specifically thought for this project, they can be easily reproduced on similar settings.
* drawingHelper.py is a library that contains only one method (draw\_solid) and is designed for task 2.

The libraries are put in display within the IPython notebook "KoraisZullich\_proj2.ipynb", in which the actual tasks of the project are carried out.

### Code description

After importing all the images within the calibration folder, we first need to detect the checkerboard intersections within the picture and define the correspondences between points in world reference frame and image reference frame. This is done via the OpenCV method findChessboardCorners, which returns a flag which indicates whether the specified patter was found, and a list containing the 2D coordinates of the intersections found within the image. 3D coordinates of such intersections are easily obtainable by setting the origin of the world reference frame to the upper-left corner of the checkerboard, taking the X and Y axis along the horizontal and vertical orientation of the checkerboard, such that the Z axis is perpendicular to it: so, all of the intersections have 3D coordinate equal to 0. Then, the unit of measurement is conveniently fixed at the distance from one intersection and the following one along either the X or Y axis. Doing so, we have computed the correspondences between 2D and 3D checkerboard intersections. A refinement to such correspondences is obtained by improving the location of the 2D points by means of OpenCV’s cornerSubPix, which is applicable to corner detectors such as the one used by findChessboardCorners. The method uses considerations on the gradient along the edges near the corners to rectify the location of the corners at subpixel precision.

After finding the correspondeces, the next step is to calibrate the camera, thus getting the matrix containing the intrinsic parameters for the camera and the matrix representing the rigid transformation from the world to the camera reference frame. The first matrix is unique and is obtained from the collection of images previously imported, the transformation matrices are one for each image, since in every picture the checkerboard is positioned differently, hence we need each time a different transformation from world reference frame to image reference frame. To perform the calibration we use calibrateCamera from OpenCV. The method estimates such quantities plus