COMP3317 Computer Vision
Department of Computer Science

Assignment 4: *Camera Calibration*

Deadline: **17:00, Apr 19, 2017.**

Background

VG3DBuilder is a computer vision based 3D modeling software developed by the Vision Group of the Department of Computer Science at The University of Hong Kong. It provides basic tools for color-to-gray-scale image conversion, corner detection, camera calibration, and image based 3D modeling.

In this assignment, you are going to implement the following three functions of the VG3DBuilder:

- Camera calibration
- Projection matrix decomposition
- Triangulation

Task

The core program VG3DBuilder, which implements the graphical user interface and the general program logic, has already been compiled into an executable. This executable will make function calls to two dynamic link libraries (DLL), namely corner.dll and camera.dll. corner.dll accomplishes the tasks of color-to-gray-scale image conversion and corner detection, and camera.dll accomplishes the tasks of camera calibration, projection matrix decomposition and triangulation. Your task in this assignment is to complete the implementation of the functions for camera calibration, projection matrix decomposition and triangulation in the dynamic link library.

To guide your coding, a project created under Visual Studio 2010 (32bit) is provided to you. In completing this assignment, you should only modify the file `Assignment4.cpp` which contains the definitions of the following three functions:

- Calibrate() – this function takes a list of 2D and 3D coordinates of a small set of image points marked by the user manually, and a list of 2D coordinates of the detected corners as input, and compute a 3×4 camera projection matrix.
- Decompose() – this function takes a 3×4 camera projection matrix as input, and decompose it into the product of a 3×3 camera calibration matrix **K** and a 3×4 matrix [**R T**] composed of the rigid body motion of the camera.
- Triangulate() – this function takes a list of camera projection matrices and a list of corresponding points as input, and compute 3D coordinates by triangulating the corresponding points.

Please refer to the tutorial notes as well as comments in the source code for the meanings and usage of the parameters of these three functions.

Requirements

Your implementation should:

- [Calibration] Classify the input 3D points into points on the *x*-*z* planes, points on the *y*-*z* plane, or points not on the calibration planes.
- [Calibration] Estimate a plane-to-plane projectivity for each of the two calibration planes using the input 2D/3D point pairs.
- [Calibration] Using the estimated plane-to-plane projectivities, assign 3D coordinates to all the detected corners on the calibration pattern.
- [Calibration] Estimate a 3×4 camera projection matrix from all the detected corners on the calibration pattern using linear least squares.
- [Decomposition] Use QR decomposition to decompose the input camera projection matrix **P** into the product of a 3×3 camera calibration matrix **K** and a 3×4 matrix [**R T**] composed of the rigid body motion of the camera.
- [Decomposition] Normalize the resulting camera calibration matrix **K** into the standard form (refer to slide 23 of lecture notes on camera calibration).
- [Triangulation] Compute 3D coordinates from corresponding points using linear least squares.
- [Triangulation] Handle the situation where one or more of the corresponding points are missing (i.e., represented by a null vector).

Note that marks will be deducted if your code contains severe bugs or results in memory leakage.

You can compare your results with those produced by the sample program for checking the correctness of your program.

Submission

Points to note when submitting your assignment:

- You should hand in only your modified `Assignment4.cpp` but not the whole project.
- You should include a `readme.txt` file describing the features you have implemented, especially when you have turned in a partially finished implementation.
- Pack your source code `Assignment4.cpp` and the `readme.txt` into a zip file, and submit it via the Moodle page.
- No late submission will be accepted.
- Programs that cannot be compiled will not be graded.

Appendix I: The Calibration Pattern

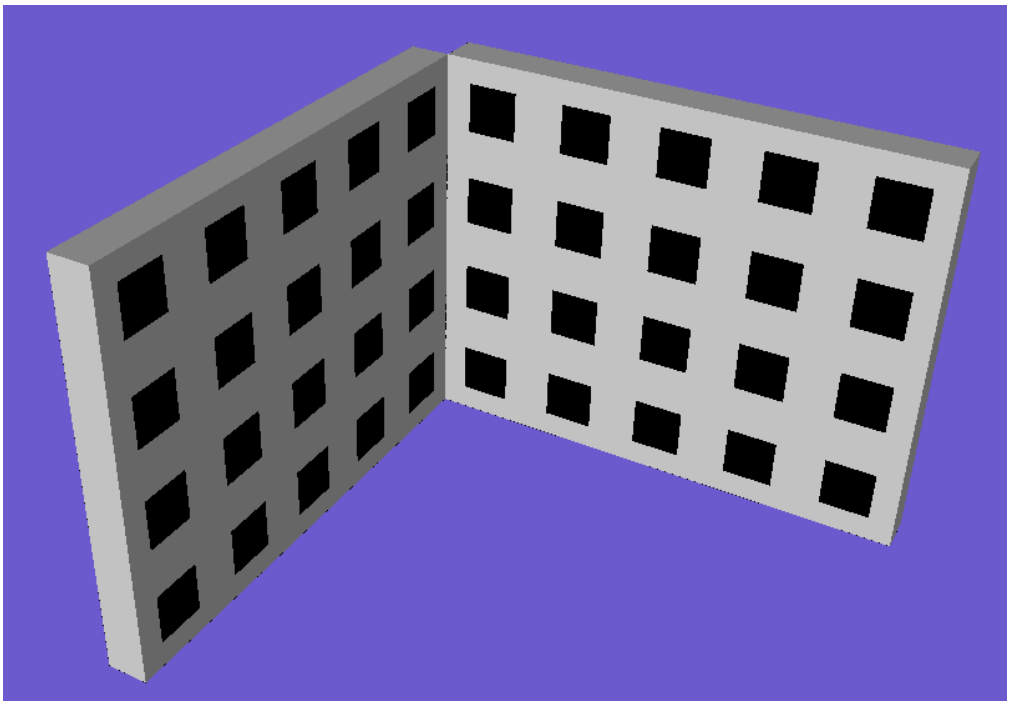

Figure 1: A 3D view of the calibration pattern

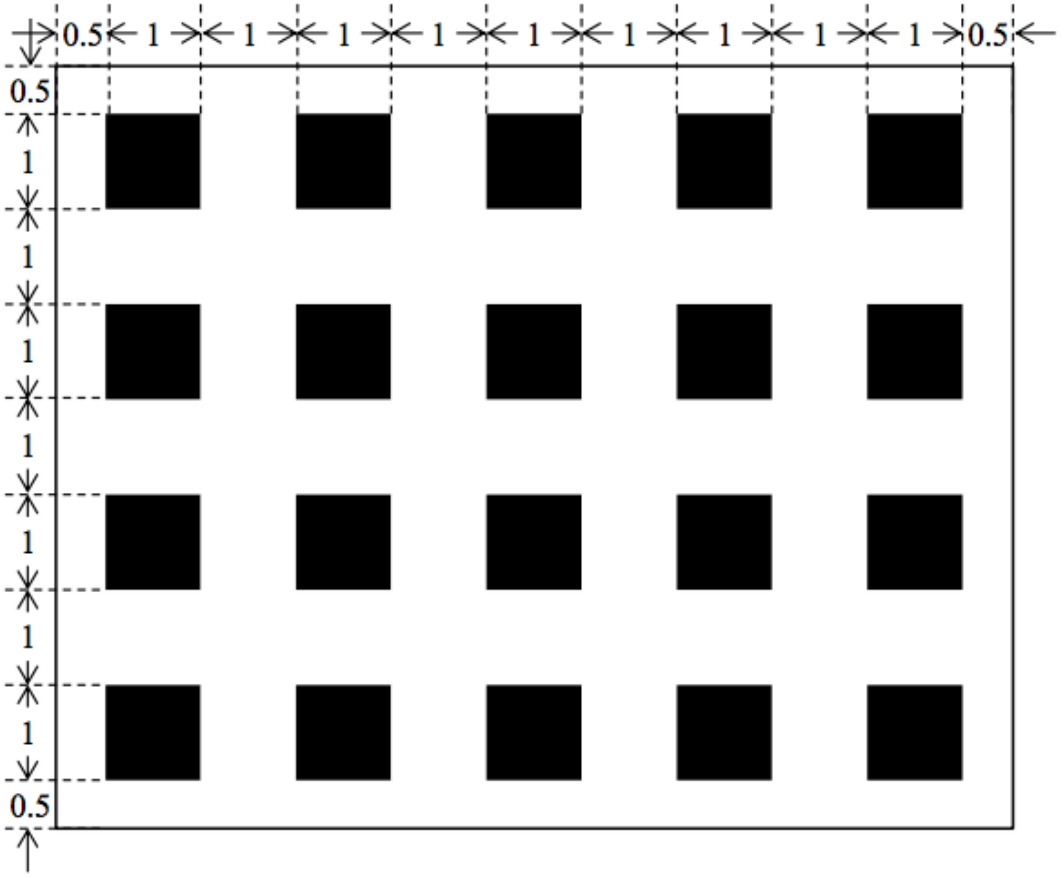

Figure 2: A 2D view of one plane of the calibration pattern (unit = inch)

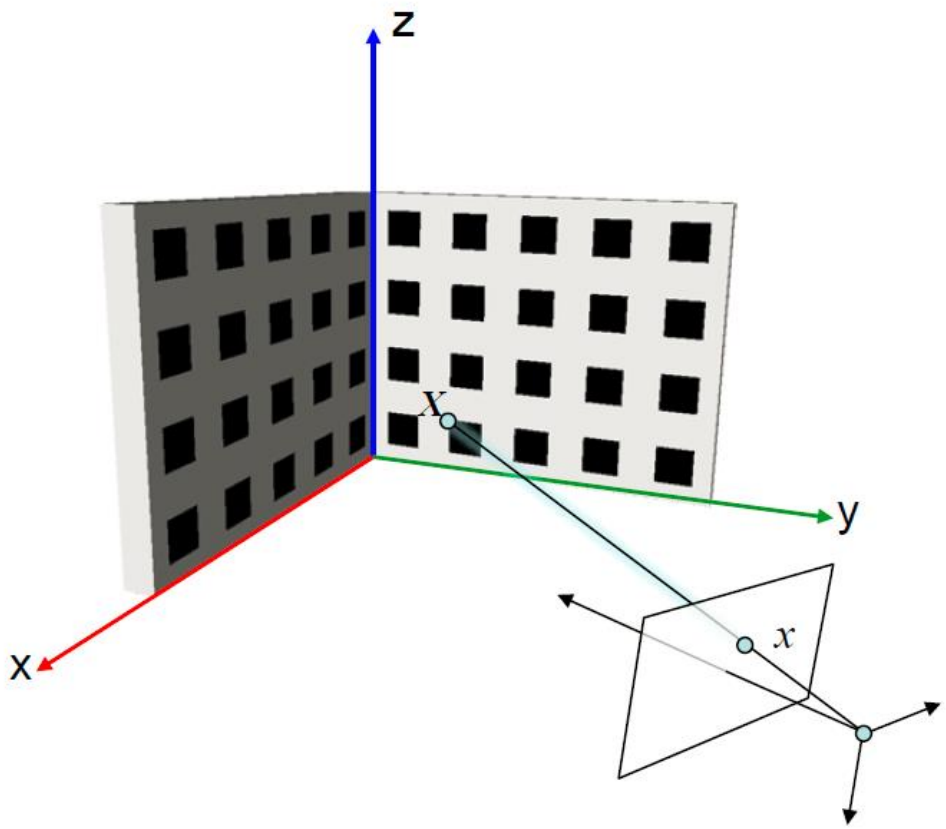

Figure 3: Definition of the world coordinate system

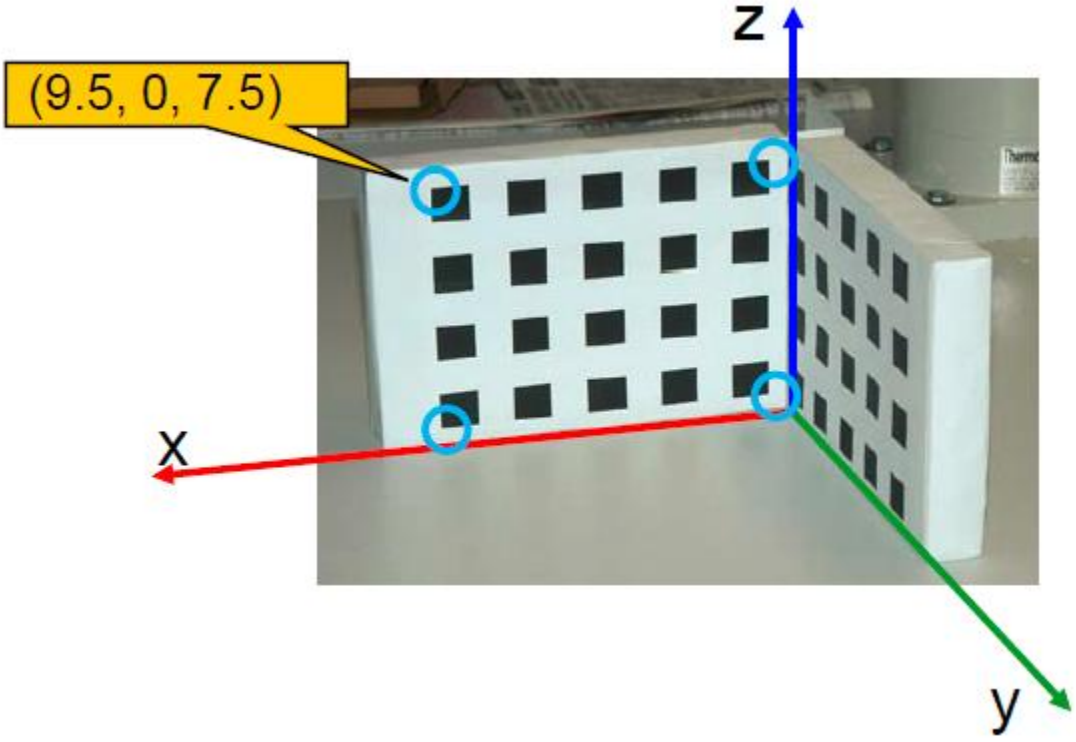

Figure 4: Examples of manually marked 2D/3D points

~ End ~