Calibration Toolchain

Sebastian Szczepanski

16. November 2006

1 Motivation

The topic of my team is to calibrate the Olympus SP500-UZ 2D camera and PMDtec PMD[vision] 19k 3D camera intrinsic (remove the barrel distortion) and extrinsic (calculate the world orientation).
For the calibration we use the Zhang 2-dimensional pattern based calibration algorithm as described in [5]. The implementation is specified in section 3. For the algorithm it is required to take minimal 3 pictures of the calibration object (from different view angles) for every intrinsic calibration. As intrinsic parameters change with focus and zoom of a camera it is necessary to calibrate each zoom and focus setting separate. For the Olympus SP500-UZ with 52 zoom and 120 focus setting we have to take 18720 pictures and even more if we want better accuracy.
So it is essential to have a fast and easy way to take several pictures automatically. We wrote a tool that controls the camera via a camera SDK and assists us in this process. The functionality of this tool is described in section 2. The process of calibration itself is the topic of section 3.

2 Taking the pictures

The tool that helps us in the process of taking several calibration image is called takePic. It is a simple win32 command line tool written in c/c++ with the help of bloodshed dev-cpp mingw based IDE package. The program connects to the camera via the Olympus ryeNV camera SDK. The command line takes 3-4 simple parameters. The exposure time in milliseconds, the zoom setting(focal length) in mm*10 and a focus setting or range of focus.

syntax: cam.exe EXPOPSURETIME ZOOM FOCUS [FOCUSEND]

With the help of the focus range parameter (FOCUS - FOCUSEND) it is possible to take several calibration pictures over a full focus range of the camera in a fast and easy way.
For different focus settings with fixed zoom the view area change is only based on the distortion so it is easy to setup the camera system for one focus setting of a fixed zoom and take the calibration images of this fixed zoom for the whole focus range of the camera.
This reduces the setup of the camera to minimal 156 for all zoom and focus setting with a minimal calibration picture count of 3 images. The program takes pictures in the highest resolution of the camera to get the most image information as possible. Because the pattern is good visible it is sufficient to preview the calibration object via the preview LCD of the camera. So a graphical user interface for the tool is not needed.
To decrease the photograph time the camera LCD gets disabled after the the pictures taking process stars. Because of the size of RAW images it is at the moment not feasible to take the calibration images in RAW format. This can change in the future. The calibration images are stored in format to ease the calibration more.

The structure of the format is:
$PROGRAM\_FOLDER/ZOOM/FOCUS/TIMECODE.jpg$
More details on takePic can be found in the takePic doxygen documentation.

3 Calibration

We use the Zhang calibration algorithm (see [5]) to calculate the intrinsic parameters, radial distortion and extrinsic parameters.
In this algorithm the intrinsic is calculated with the help of corresponding feature points from a fixed grid and pictures of the grid from different view angles.
We use a chessboard for the feature point grid (connection points between connected squares). We take calibration images as described in 2 and find the feature points in the pictures (semi-)automatic with the help of a corner detection algorithm.

3.1 calculate intrinsic parameters

To find the chessboard points automatic and calculate the intrinsics from these points we use a modified version of the open source tool CamChecker (see [11]) by Matt Loper from the Brown University (http://www.brown.edu/). We modified the tool to overcome some issues.
1. we also have to calibrate a low resolution(120 x 160 pixel) 3D camera with bad intensity pictures where automatic corner detection is not easy
2. we want to output the calibration data in our own calibration database format
3. we want to save the extrinsic guess found by Zhang algorithm ([5])

To address these problems we added the following modifications to CamChecker:
1. we modified the parameter output
(now all parameters are printed into a file in our own format)
2. we changed the feature point detection tolerance
(we added a parameter to change the tolerance of corresponding square point detection for low resolution pictures)
3.we added a grid for manual setup of the checkerboard points
(for bad intensity pictures)
4. we added the ability to zoom into the picture for better corner point analysis
5. we read out the focal length of a picture to save this data for extrinsic calibration

A detailed description of the way how CamChecker/CamChecker+ detects corner points and calculates the intrinsic data can be found in the CamChecker+ doxygen documentation.

3.2 calculate extrinsic parameters

For the first extrinsic calibration approach we use the extrinsic guess found by CamChecker/CamChecker+ and output the extrinsic data (rotation matrix and translation vector), the intrinsic data and the grid points into a file. This guess is then used to be optimized via minimizing the back projection error.
In the current state of the project we have some errors in this process. the z translation is wrong and x and y translations are slightly wrong. It also happens that the rotation axis are flipped or interchanged (the reason for this error will be described in section 4).

4 known issues

One problem we found so far is the fact that the rotation axis of the extrinsic parameters are flipped or interchanged. This is because of the point symmetry of the calibration pattern. In an image of a 45° rotated grid you are unable to figure out if one edge is up/down or left/right. The solution to this problem would be a pattern with an obvious orientation. This would be no problem to the underlying Zhang algorithm because this algorithm only needs corresponding points. But it needs improvements to the pattern detection and analyzing code. This issue is solved in GML C++ Camera Calibration Toolbox (see A and [7]) by using a non-uniform chessboard/checkerboard (odd x even resolution).
Another problem is the calibration of the 3D camera. The PMDtec PMD[vision] 19k has only one zoom and focus setting so we only need 3 calibration pictures. But analyzing the intensity pictures of the camera is not easy. The chip is only low resolution and very sensitive on reflections. Also the fixed zoom and focus forces us to have a certain distance to the calibration object. Both distance and reflection force us to use a very good and big pattern. The absence of the PMD camera refused us to test the big pattern for calibrating the 3D camera.

A related work

There are several other projects implementing the Zhang or other algorithms for camera calibration.
One is the openCV (open computer vision library by Intel) which is c/c++ library with several functions for checkerboard corner detection and intrinsic and extrinsic calibration based on the Zhang algorithm ([5]). It can be found under [6].
The GML C++ Camera Calibration Toolbox (by Alexander Velizhev from
Moscow State University http://www.msu.ru/en/) is a program that uses openCV and has a easy to use GUI. You can find it here [7].
Another project that implements camera calibration is DLR Camera Calibration Toolbox by the DLR (Deutsches Zentrum für Luft- und Raumfahrt http://www.dlr.de/). It is a very complex tool set (corner finder and calibrator) that can output distortion up to 7th order. The program is written in the mathematical programming language IDL. To use the program you have to install the IDL virtual machine fist. You can find more information's on the used calibration algorithms here [9] and [10] and more on the program here [8].

B notes on CCD diameters

The Olympus SP500-UZ has a 1/2.5inch diameter CCD (according to the documentation).
These 1/2.5inch CCDs have following real dimensions:

CCD	x-y-dimension	diagonale
1/2.5inch	5.76 x 4.29 mm	7.2 mm

See reference [1],[2] and [3] for more details on CCD dimensions.

You can calculate the angle/field of view of a camera with the following formula:

$fov = 2 \cdot arctan(d/(2 \cdot f))$ with

$f = focal\_length$	(for the Olympus SP500-UZ lens it is 6.3mm till 63mm)
$d = dimension$	(horizontal , vertical or diagonal dimension in mm)
fov	the horizontal, vertical or diagonal angle (corresponding to d) of f.

See [4] for more informations about angle of view of cameras.

For our 6.3mm focal length (minimal zoom of the standard SP500-UZ lens) we get a diagonal angle of view of 66°. This corresponds to the focal length of 33mm of a 35mm classical miniature cameras (this relation is also mentioned in the Olympus SP500-UZ manual).

References

[1] Wikipedia article on Normal Lenses
http://en.wikipedia.org/wiki/Normal_lens 02/10/2007

[2] Wikipedia article on Digital photography
http://en.wikipedia.org/wiki/Digital_photography 02/10/2007

[3] Vincent Bockaert. Sensor Sizes. *Digital Photography Preview online glossary*
http://www.dpreview.com/learn/?/Glossary/Camera_System/sensor_sizes_01.htm 02/10/2007

[4] Wikipedia article on Angle of view
http://en.wikipedia.org/wiki/Angle_of_view 02/10/2007

[5] Z. Zhang. A Flexible New Technique for Camera Calibration.
http://research.microsoft.com/∼zhang/Calib/

[6] OpenCV
http://www.intel.com/technology/computing/opencv/index.htm

[7] The GML C++ Camera Calibration Toolbox
http://research.graphicon.ru/calibration/gml-c-camera-calibration-toolbox-5.html

[8] DLR Camera Calibration Toolbox
http://www.dlr.de/rm/desktopdefault.aspx/tabid-1524/

[9] K. H. Strobl and G. Hirzinger. Optimal Hand-Eye Calibration. In Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2006), Beijing, China, pp. 4647-4653, October 2006.

[10] J. Weng, P. Cohen, and M. Herniou. Camera calibration with distortion models and accuracy evaluation. In IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI), 14(10): 965-980, 1992.

[11] Matt Loper. CamChecker
http://matt.loper.org/CamChecker/CamChecker_docs/html/index.html