3D Surround Vision Calibration Tool

User Guide

TABLE OF CONTENTS

1 Introduction

1.1 Overview

The 3D Surround Vision Calibration Tool is a graphical user interface (GUI)-based PC program addressing the camera calibration needs for TI's Surround Vision (Surround View, 3D Perception) demos.

Section 1.2 lists the system requirements to run the tool.
Section 2 provides an introduction to the main screen and the general workflow.
Sections 3 to 6 go into more detail on the individual parts of the tool:

- specification of "Input Frames",
- specification of "Distortion Centers"
- specification of "Lens Look-up Tables"
- estimation of "Camera Poses".

The tool can be used in different ways, so Section 7 provides instructions for a few tasks that are commonly needed when calibrating TI's Surround Vision demos.

1.2 System Requirements and Installation

System Requirements:

- 64 bit Microsoft Windows OS
- Matlab Compiler Runtime (MCR) version 8.4 . MCR can be downloaded for free from the Mathworks website. You do NOT have to install Matlab.

Installation:
The tool is located in the subfolder ./exe_out/. You should copy this entire folder to a location of your choice inside your user space to minimize risk of file access permission issues.

2 Main Screen

Double-click on file "main.exe" inside the "./exe_out/" folder to start the program. The welcome screen in Figure 1 appears. It lets you choose a working directory. It is recommended that you choose a directory created specifically for the calibration of one setup. All the output files generated by the tool will be stored in this folder. Use the "edit" button to choose the working directory.

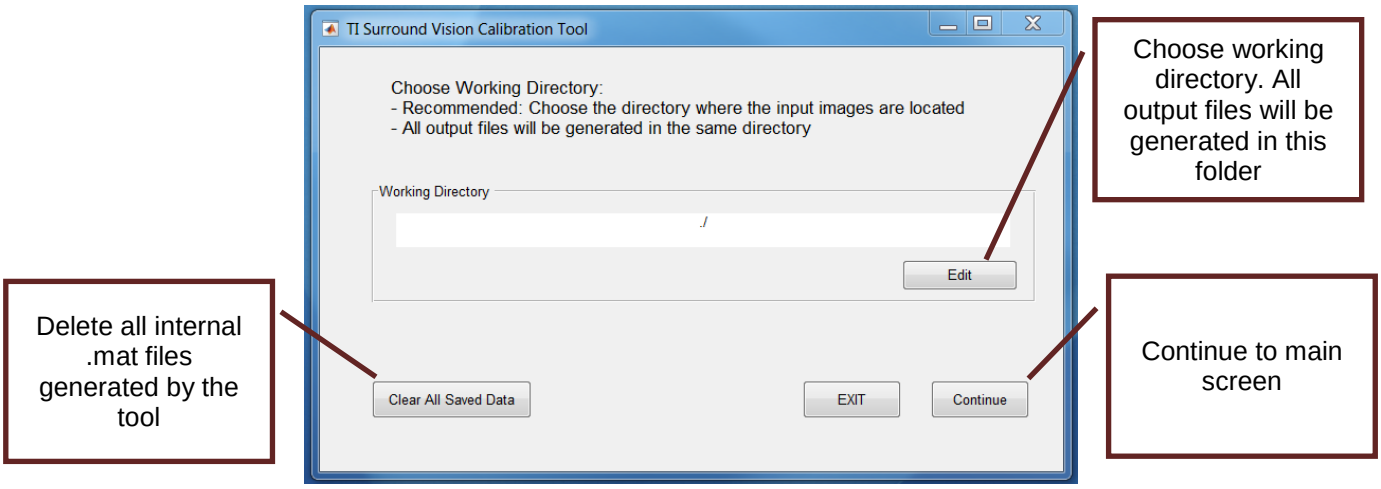

Figure 1: Welcome Screen

The GUI is able to remember data you entered previously. To do so, it stores .mat files (inputImagePrms.mat, distCenters.mat, lut_ldc.mat, chartPrms.mat) in the working directory. Make sure you don't use these file names for other purposes. The button "Clear All Saved Data" deletes all these files and lets you start fresh again. Press "Continue" to reach the main screen (Figure 2).

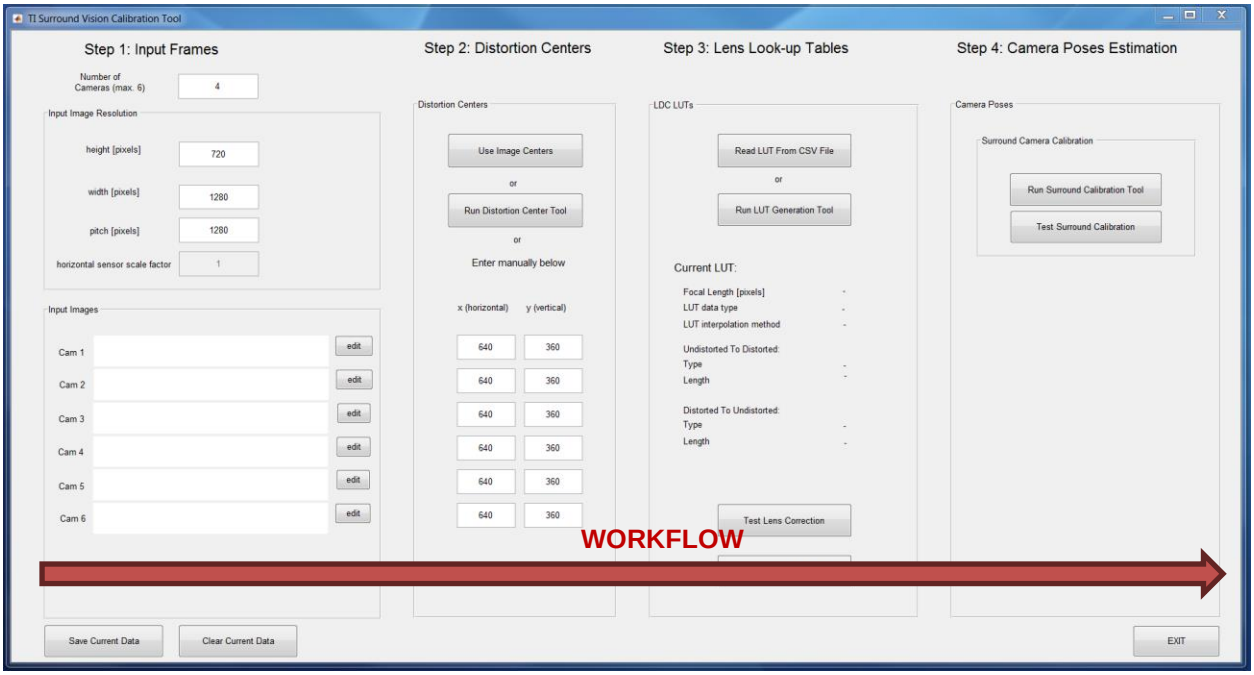

Figure 2: Main Screen

The main screen contains all the functionality of the tool. Some functions will open their own windows, but they will always return to the main screen. The main screen is partitioned into 4 steps (columns). The workflow is from left to right, meaning that a function never depends on data from a column to the right, but may depend on some or all of the data from columns to the left. These data dependencies are detailed in the following sections when the individual functions are introduced.

3 Step 1: Input Images

Please refer to Figure 3. The first column lets you specify the camera input data, such as number of surround cameras (=cameras that are part of the Surround Vision system, do not count other cameras, such as, for example, front camera analytics camera) in your setup, and frame parameters height, width and pitch in pixels. Note that width refers to the actual image content, whereas pitch refers to the width of the frame in memory. In other words, pitch = width + horizontal padding.

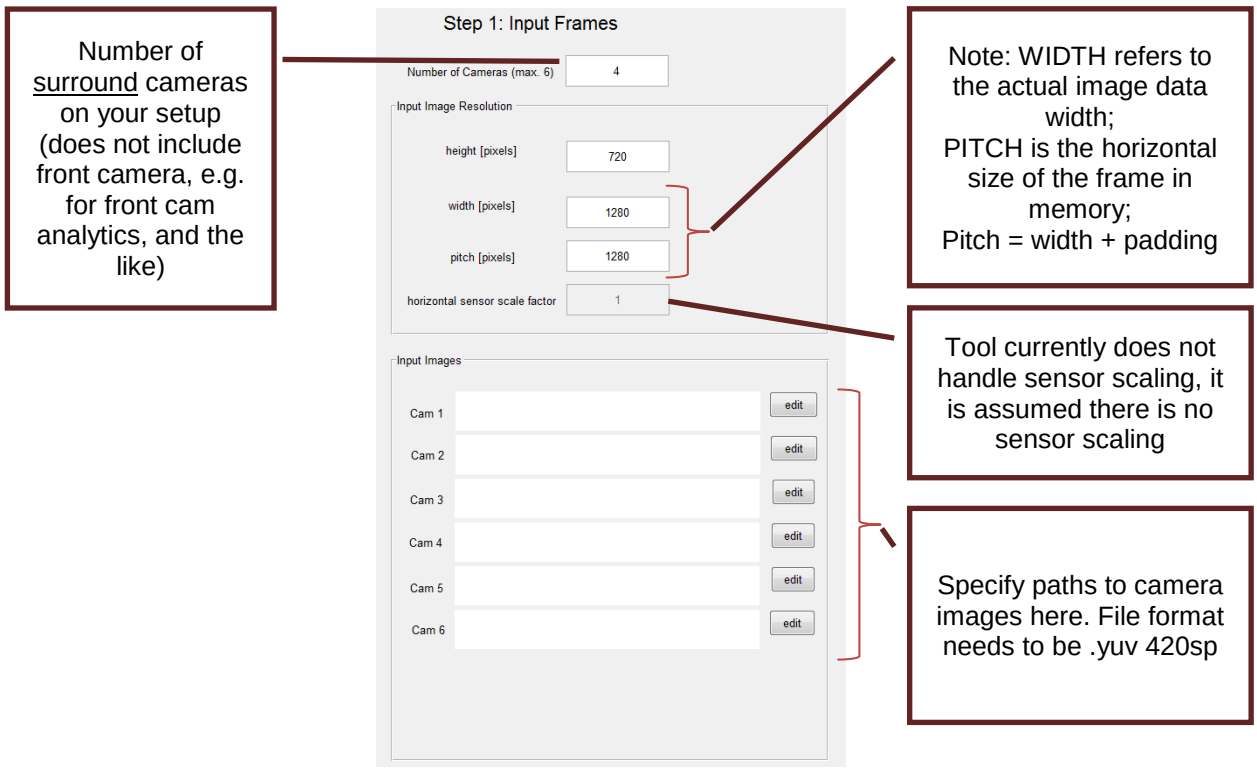

Figure 3: Step 1 - Input Frames

For some of the tool functionality, a set of dumped image frames from the cameras is needed. The tool assumes that the images are available as 420sp .yuv file format. If available, please use the "edit" buttons to specify the .yuv file locations for each camera.

4 Step 2: Distortion Centers

Please refer to Figure 4. Distortion centers are the pixel locations in the input image around which circular lens distortion occurs. For the interested reader, lens distortion is explained in more detail in Section 5.1. For a high-quality camera module, the distortion center coincides well with the image center. The button "Use Image Centers" will fill in the image center based on the image dimensions provided in step 1.

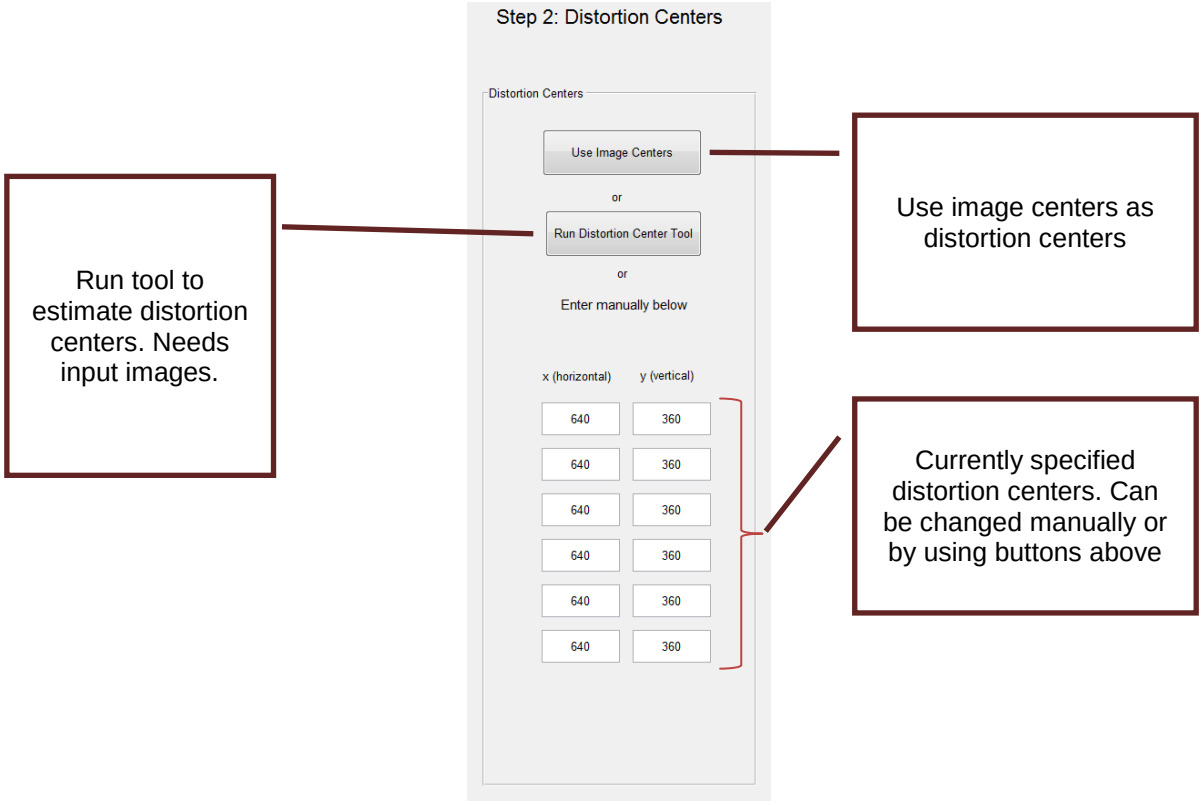

Figure 4: Step 2 - Distortion Centers

In lower-grade camera modules, distortion center might not coincide with the image center. If you know the distortion centers from another source, you can enter them manually in the text fields.

Alternatively, the tool also provides a function to estimate distortion centers from input images. Click "Run Distortion Center Tool" to run this subtool. For each input image, click points on the boundary of the fisheye image as shown in Figure 5. The distortion center tool will fit a circle to these points and use the circle's center as the image's distortion center.

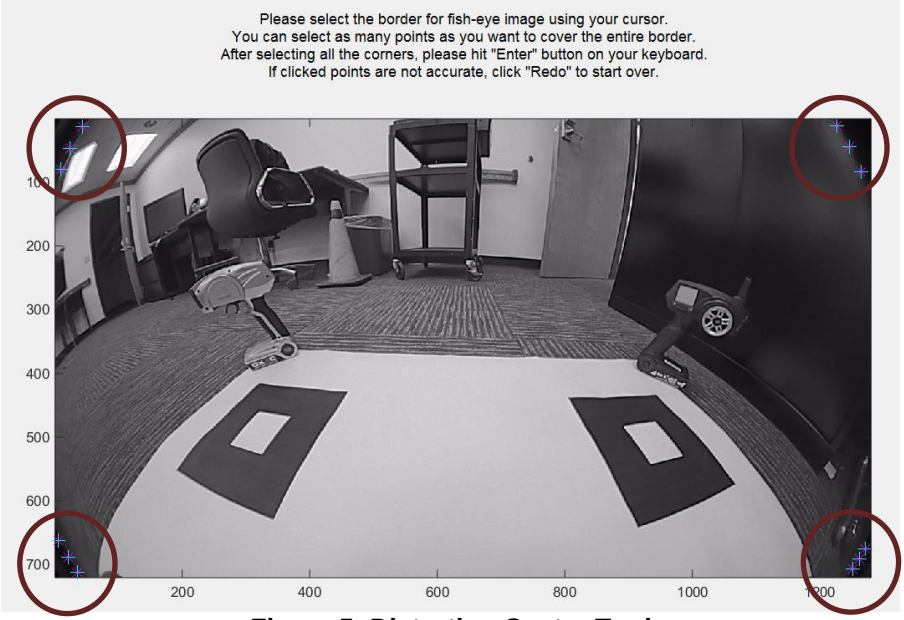

Figure 5: Distortion Center Tool

5 Step 3: Lens Distortion LUTs

Please refer to Figure 6. A lens look-up table (LUT) represents the lens distortion function of a lens. The lens distortion function describes how incoming light rays are bent by the lens. The bending of light rays is necessary for fisheye lenses to achieve a wide angle field of view and lens vendors put great effort in designing their profiles. For vision applications, however, it is often desirable to have an "undistorted" image, where distortions caused by the lens are removed. To convert an image to its undistorted image and vice versa, the lens distortion function needs to be applied. This is done by lens distortion look-up tables. TI's vision demos use a specific format for the lens LUTs. This step provides means for the user to specify their lens function, convert it to TI's format and test the quality of the LUT.

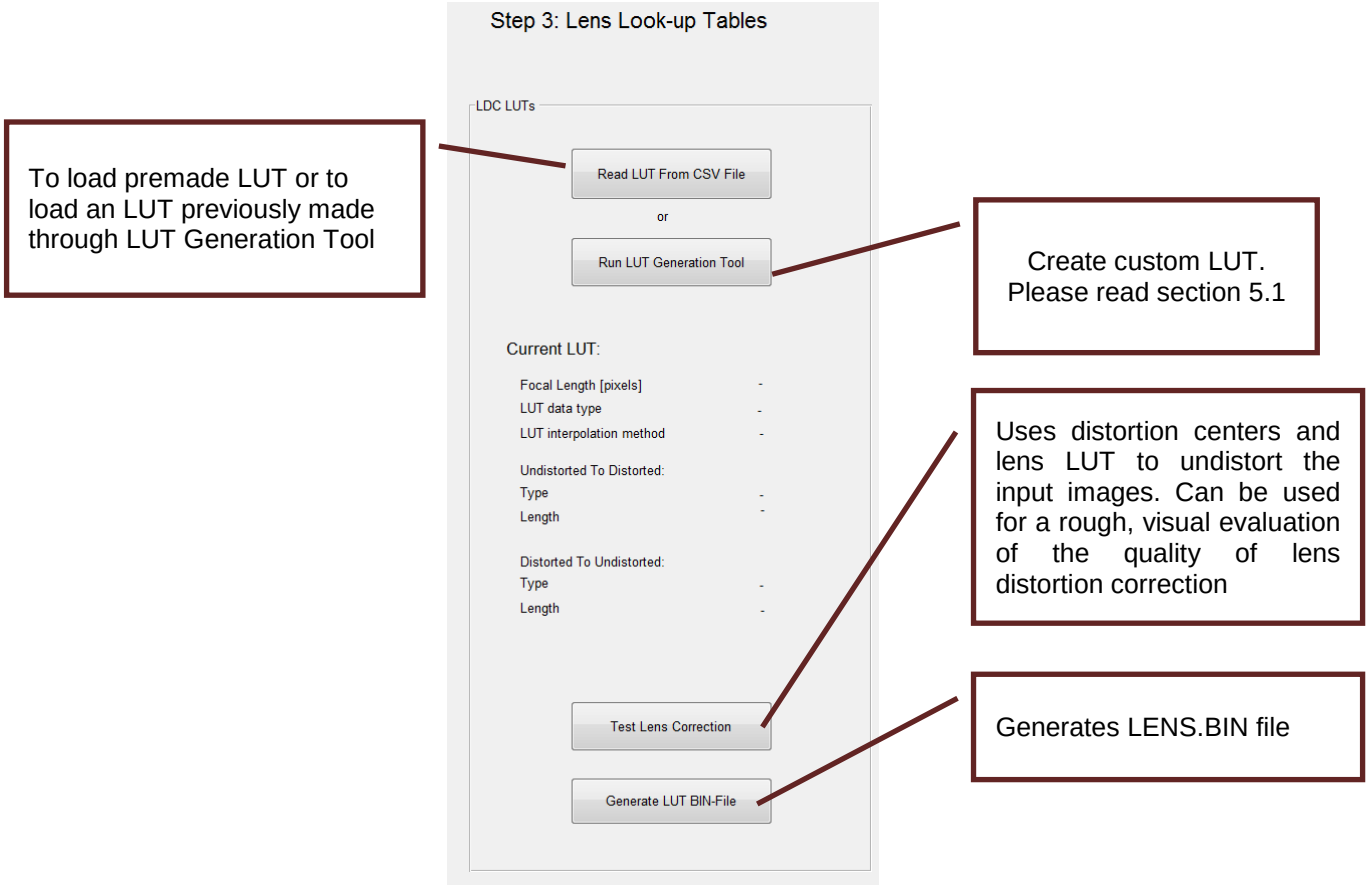

Figure 6: Step 3 - Lens Look-up Tables

This tool comes with some pre-made LUTs for common camera-lens modules used in TI's Surround Vision demos. If you are using either of these

- 2DSV Equisolid (Sunex DSL219E-670-F2.0 lens + OV10635 sensor)
- TIDA00262 module (Sunex DSL218 lens + AR01040AT sensor)
- IMI DSSMR05 module

click on the button "Read LUT from CSV-File", navigate to subfolder "<Tool Installation Directory>/exe_out/ldc_luts/" and select the appropriate csv-file. All LUT parameters are loaded and stored and you are done with this section.

If you have a different camera module, your job is a little harder. You need to run the "Lens LUT Generation Tool" LUT by clicking the button to create your custom. Please refer to section 5.1 for details.

The lens LUT column provides two more functionalities. The button "Generate LUT BIN-File" generates a binary file LENS.BIN, which is needed for the Surround Vision demos in VisionSDK. This function relies on all previous data except input images. For backward compatibility with previous versions of VisionSDK, the tool also outputs a c-file ldc_lut.c, which can be compiled directly in C-applications.

The button "Test Lens Correction" serves to roughly evaluate the quality of the distortion centers and lens LUT. It displays the undistorted input images using the currently specified lens distortion parameters. For a good set of parameters, straight lines should appear straight in the undistorted image. This function requires all previous data fields.

5.1 Lens Look-up Table Generation Tool

The objective of this tool is to let the user specify his/her lens distortion function and convert it to the format required by TI's vision demos. To use this tool, an understanding of lens distortion is required. We start with a brief tutorial on lens distortion, before explaining the tool itself.

5.1.1 Theoretical Background

A perfect distortion-free camera is modeled by the pinhole camera (Figure 7). The pinhole camera is defined by its principal point on the sensor and the focal length f. The optical axis extends from the principal point perpendicular to the sensor plane. The focal point is the point on the optical axis that is f away from the sensor. A light ray coming the world is projected onto the sensor through the focal point on a straight line. We call the angle between the optical axis and a light ray θ (theta). Theta is always non-negative. A light ray with angle theta hits the sensor at a certain distance ru from the principal point. This distance ru is called the undistorted radius.

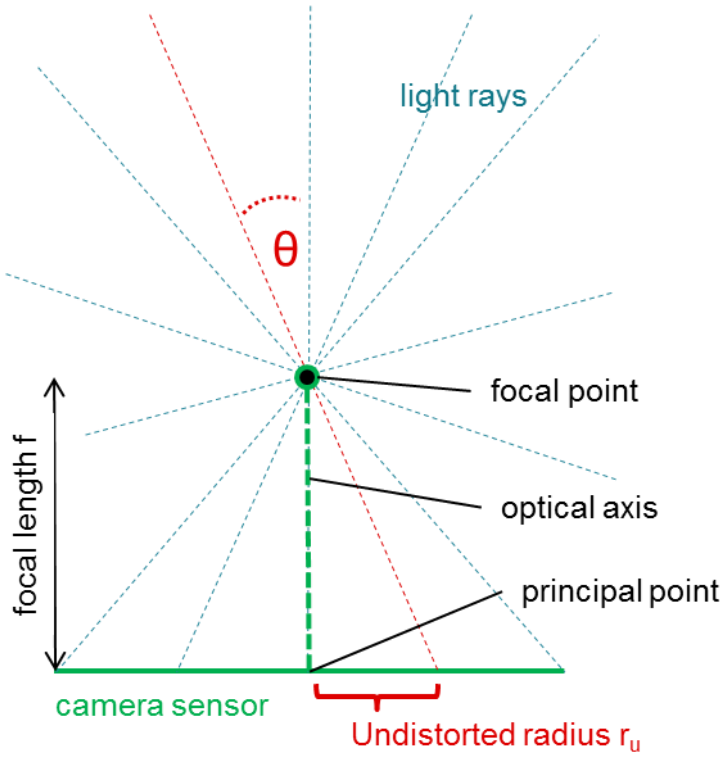

Figure 7: Pinhole Camera (distortion-free)

Even though Figure 7 is drawn in 2 dimensions for convenience, it is important to realize the 3-dimensional geometry of the pinhole camera: The sensor is a plane in 3D space, light rays are lines in 3D space, etc. The set of light rays with common angle theta will project onto a set of points on the sensor forming a circle on the sensor with radius ru from the principal point. It makes sense, therefore, to associate a theta value with a ru value and the geometrical relationship is easily seen to be ru = f * tan(theta).

Due to curvatures on a real lens (intended by design or unintended by imperfections), lens distortion makes the light rays bend at the focal point as shown in Figure 8. This bending is necessary for fisheye lenses to achieve a large field of view for a given sensor size. Since lenses are typically manufactured with circular symmetry, we can assume that all light rays with the same angle theta are bent by the same degree. This means that the set of light rays with common angle theta will project onto a set of points on the sensor forming a circle on the sensor with radius rd from the principal point. The principal point is the distortion center, and we call rd the distorted radius. The lens distortion is, therefore, fully described by the mapping of theta onto rd. We call this mapping the lens distortion function.

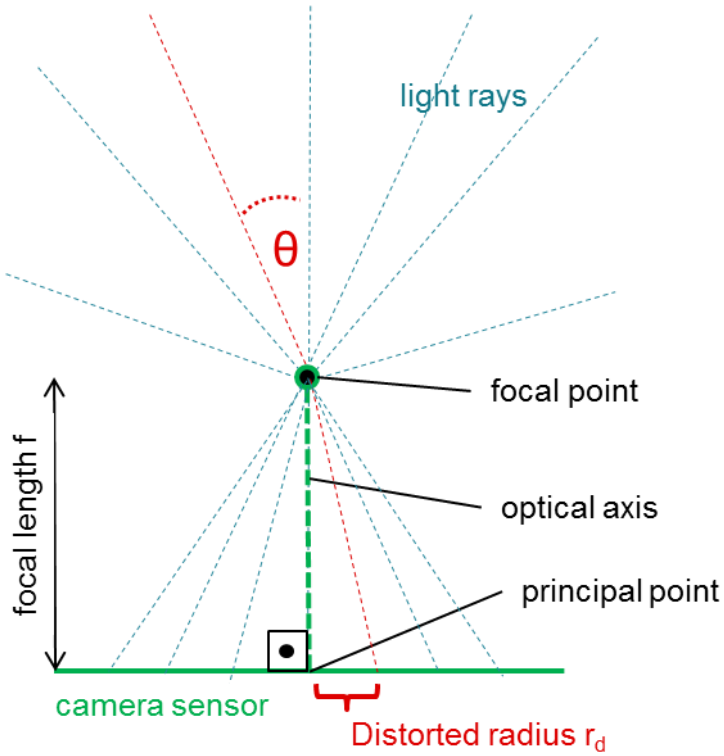

Figure 8: Camera with Lens Distortion

Some lenses are manufactured to obey an analytical model for the lens distortion function. Common models are, for example,

- equisolid model: rd = 2 * f * sin(theta/2)
- stereographic model: rd = 2 * f * tan(theta/2)

Other lenses are custom-designed by the lens vendor. In this case, the lens vendor typically provides the lens distortion function. Otherwise, external tools exist to estimate the lens distortion function from images (not part of this tool, though).

A note on units. Focal length, distorted and undistorted radius are lengths and their physical unit is, therefore, meters [m]. This makes sense when describing the physical properties of a lens. If we attach a sensor and start imaging, however, it becomes more natural to talk about lengths in terms of pixels [p]. A pixel is a square cell on the sensor with a certain width, the pixel size given in meters [m]. The pixel size should be provided by the sensor manufacturer. One can convert any length measured in meters into unit of pixels by diving it by the pixel width, that is length L[p] = L[m]/pixel size[m].

5.1.2 LUT Generation Tool Details

We now turn to the lens LUT generation tool (Figure 9). There are two types of fields:

A) Fields that describe the camera-lens module
B) Fields that describe the properties of the custom LUTs generated by the tool.

The typical user only needs to modify fields A) and they are described next. The focal length needs to be specified in units of pixels. If the focal length is provided in meters, please convert to pixels as described above. If your lens follows the equisolid or the stereographic model, choose according radio button on the top right.

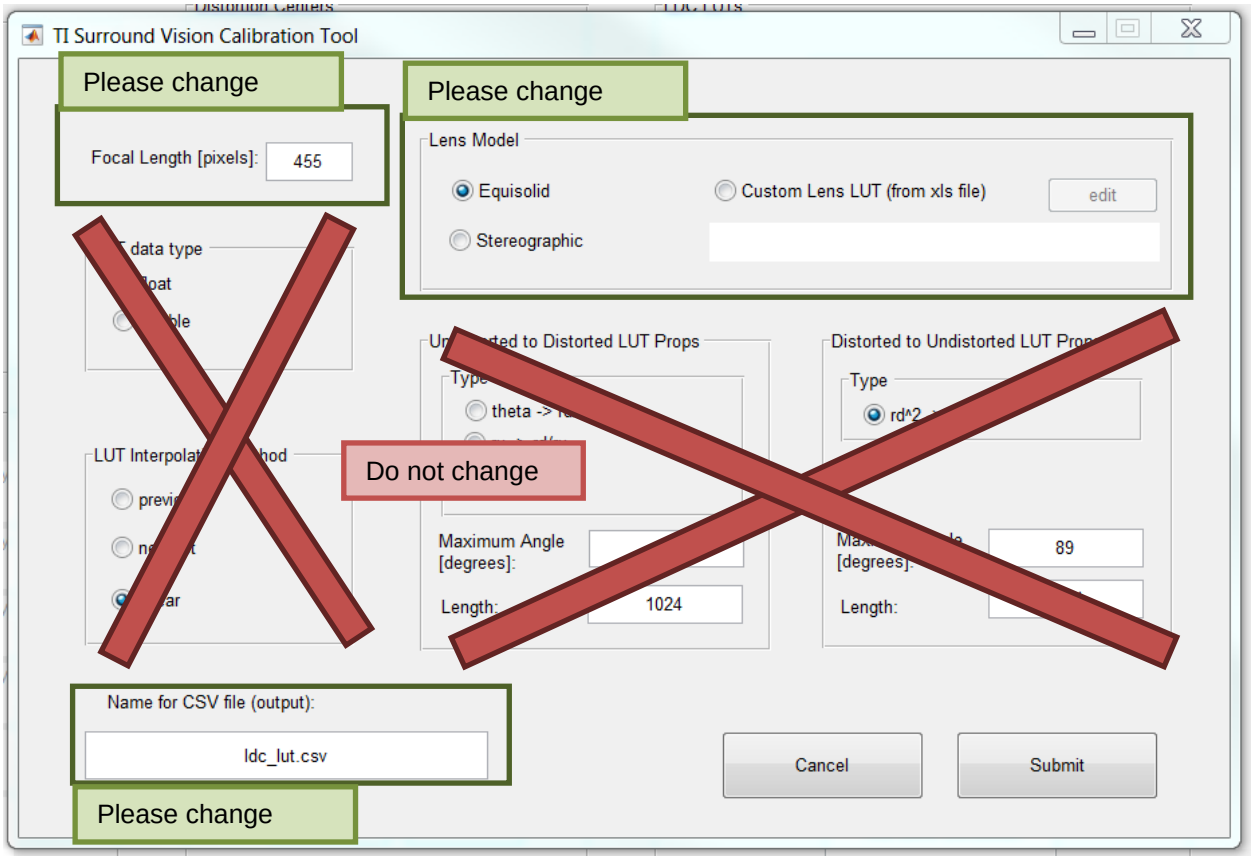

Figure 9: Lens LUT Generation Tool

If you have a custom lens distortion function, it needs to be provided in form of an Excel (.xls) sheet where the first column provides the angles theta in degrees in increasing order starting at 0, and the second column provides the corresponding distorted radius rd in pixels. Choose the "Custom Lens LUT" button and select your .xls sheet.

Finally, if desired, change the file name for the csv-file that will be generated for future reference. The csv-file will contain all info related to your lens LUT and can be loaded in the future in the main screen of the calibration tool through the "Read LUT from Csv-File" button. The format of the csv-file is defined in section 5.1.3, even though for normal use of the tool, the format does not need be known.

The other fields in this window pertain to B) and should only be changed if the user is certain of its consequences.

For completeness and better understanding of the generated lens LUT, we provide an explanation of the remaining parameters. The typical reader may skip this material.

The tool generates two LUTs one mapping points from undistorted to distorted (u2d) coordinates, and the other doing the inverse mapping (d2u). Each LUT implicitly assumes an input argument which is sampled uniformly with a fixed step size starting at zero. The input samples are defined by their type (what is the input to the LUT, e.g., theta, rd, ru), step size and number of samples. The output argument is defined by its type (what is the output of the LUT, e.g., rd/ru) and the output values (array) for each input sample.

The remaining fields of the Lens LUT Generation Tool window then have the following meaning:

- Data Type: data type of the values in the LUT array. At runtime, all operations will be performed with this datatype/precision as well.
- Interpolation Method: The method of how output values are created for input values lying between two sample points.
  - Previous: Use output value at previous input sample point
  - Nearest: Use output value at closest input sample point
  - Linear: Linearly interpolate output value between previous and next sample point

- LUT properties
  - Type: input and output types for the table look-up. Different types represent tradeoffs between speed and accuracy.
    - u2d:
      - ru->rd/ru: fast, not very accurate
      - theta->rd/ru: medium fast, medium accurate
      - theta->rd: most accurate, slowest
    - d2u:
      - rdSquared->ru/rd: very fast, accurate
  - Maximum Angle: maximum angle theta that the LUT can support. Angles larger than this produce invalid results and need to be handled at runtime.
  - Length: number of samples for the LUT. The higher, the more accurate the table look-up will be, but at the cost of increased memory space.

5.1.3 Format of the .csv File

Please read section 5.1.2 first.
File format for Ldc_lut.csv file: One line per value, no commas.

1 Line:	focal length (in pixels)
1 Line:	data type (refer to ldc_config.h in VisionSDK)
1 Line:	interpolation method (refer to ldc_config.h in VisionSDK)
1 Line:	u2d.type: (refer to ldc_config.h in VisionSDK)
1 Line:	u2d.step: step size of the LUT in the input argument
1 Line:	u2d.length: number of samples in the LUT
<u2d.length> Lines:	u2d.table entries
1 Line:	d2u.type: (refer to ldc_config.h in VisionSDK)
1 Line:	d2u.step: step size of the LUT in the input argument
1 Line:	d2u.length: number of samples in the LUT
<d2u.length> Lines:	d2u.table entries

6 Step 4: Camera Poses Estimation

The objective of camera pose estimation is to generate the extrinsic calibration of a system of cameras. In this particular tool, we enable calibration for a four camera system that is mounted on a platform in the Surround configuration. The pre-requisite to perform this step is that one has captured four input images with the calibration chart visible with the cameras positioned in their final correct positions. Instructions can be found in Section 4.2.2 of "VisionSDK_SurroundVisionDemos_UserGuide.doc".

Click on "Run Surround Calibration Tool" shown below in Figure 10 to begin calibration procedure.

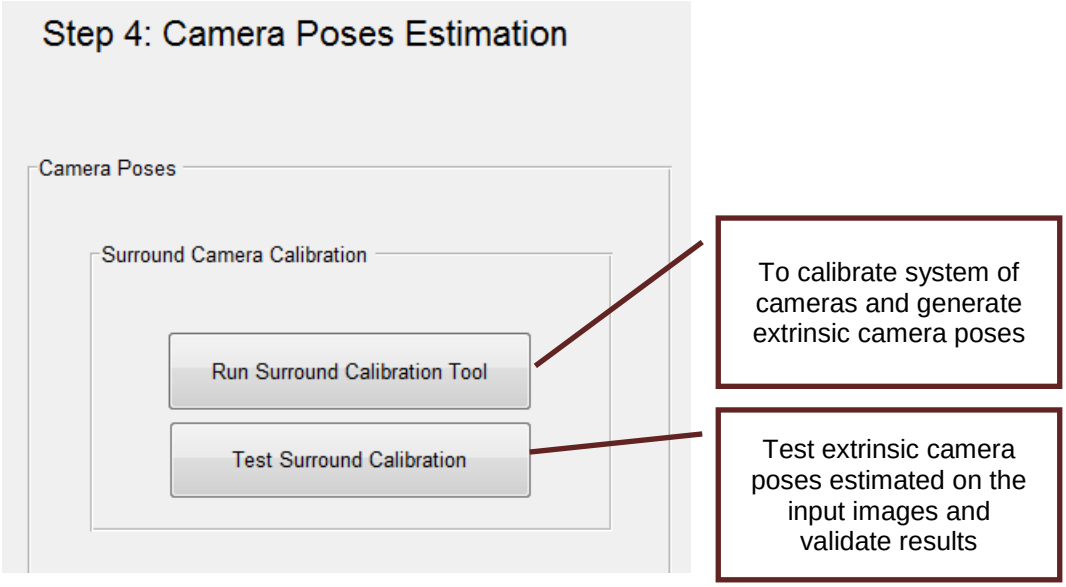

Figure 10: Step 4 – Camera Poses Estimation

6.1 Chart dimensions

The tool is designed to enable certain modifications to chart dimensions, if necessary to accommodate changes in size of setup. One can change the chart dimensions as per the setup. The constraint on the chart dimensions are that the patterns on the four corners need to be squares. Also, the patterns should only have a vertical and horizontal translation and there should be no rotation w.r.t each other. It is ideal to place the patterns on a single piece of paper, if possible, to avoid inconsistency.

The box titled Pattern Width is the measurement of both the height and width of each square. We assume that all patterns are squares they are of the same size. The pattern in the bottom left corner is the reference pattern placed at origin marked by an 'X'. The subsequent patterns are placed with offset on X- and Y-axis and the offsets can be updated here. Click "Continue" to proceed to next step.

NOTE: The dimensions seen in the example below are the dimensions of the chart provided in ./docs/poster_calib_chart.pdf.

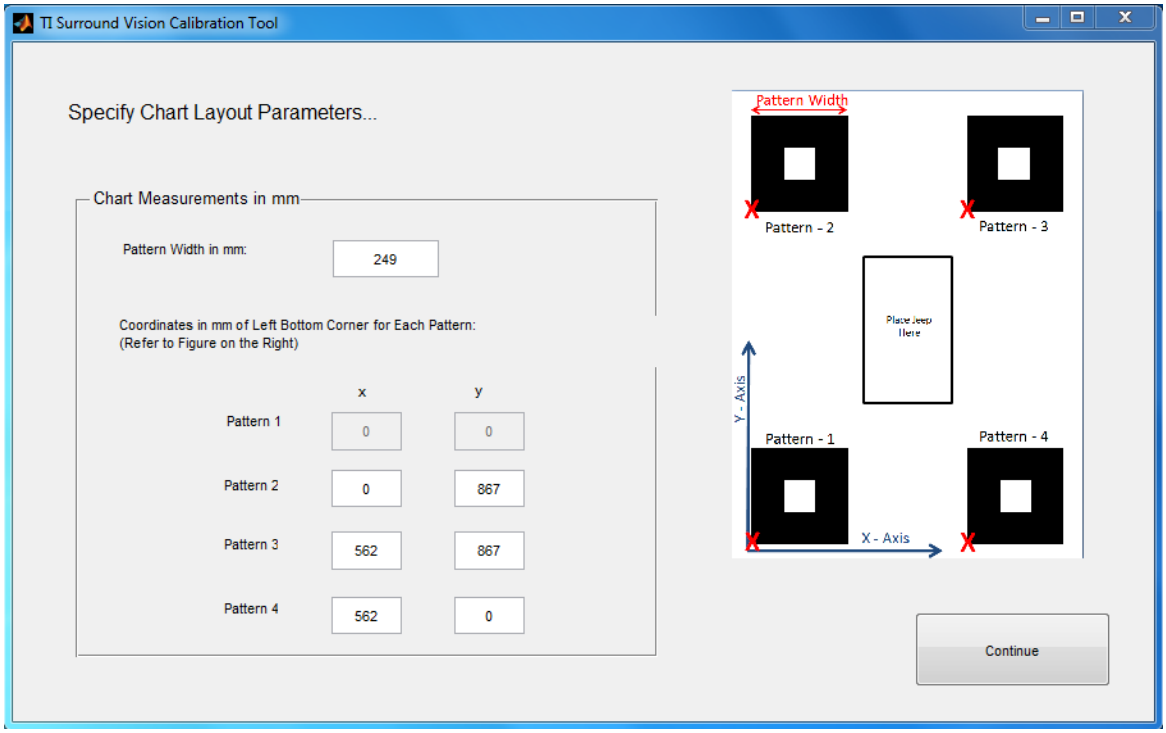

Figure 11: Step 4.1 – Enter camera calibration chart dimensions

6.2 Click pattern corners

The next step allows one to click the corners of the patterns in the captured images manually. Click 8 points on each image to choose the corners of the pattern. Please follow the order shown in Figure 12 below. Once you click all eight corners in the correct order press "Enter".

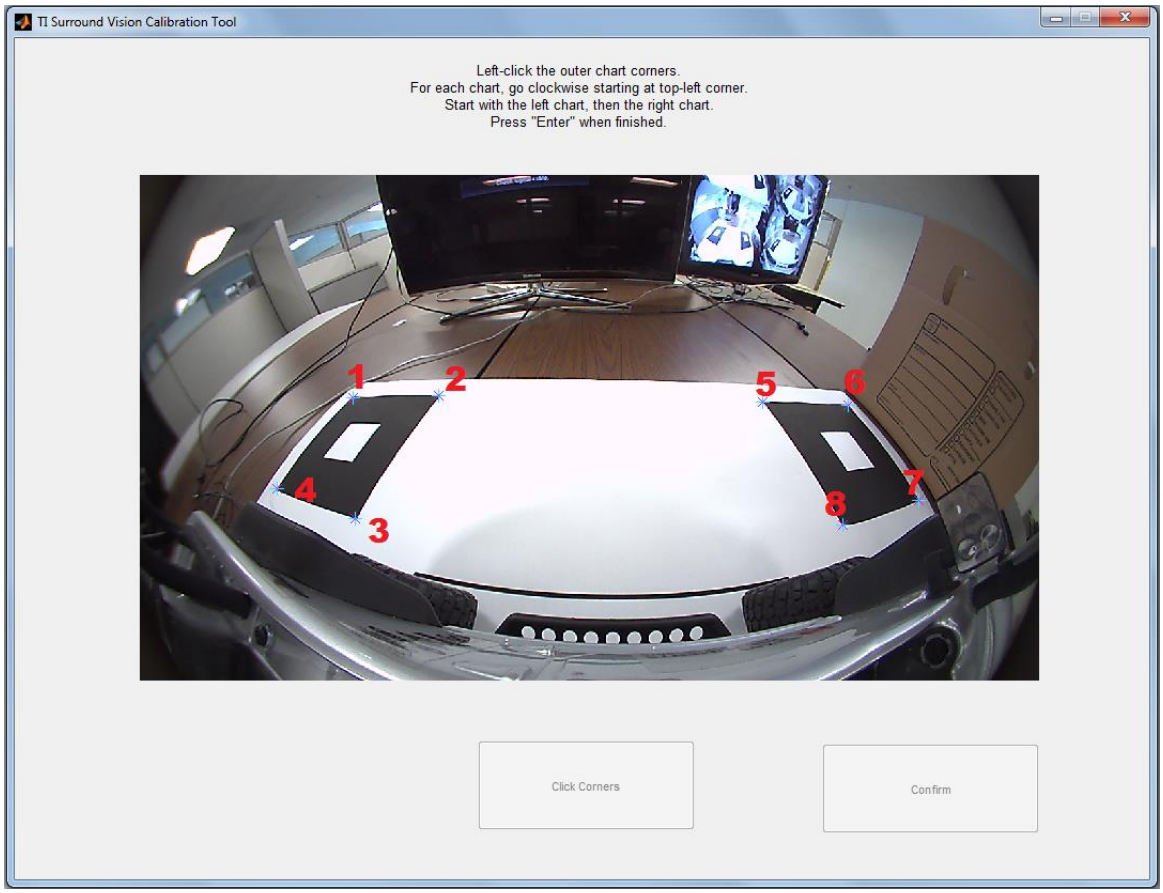

Figure 12: Step 4.2 – Click chart corners on four captured images

6.3 Confirm clicked corners

Once the clicked locations in the image are highlighted, review and confirm to move to the next image. If you are not satisfied with the clicked points, use the "click corners" button again to redo the points for the chosen image. When you have clicked all the locations in the input images you will receive a confirmation prompt shown in Figure 14. This confirmation suggests that the output files necessary to obtain calibrated output on the embedded system have been generated in the working directory. The calmat.c file is a source code file with the extrinsic camera calibration parameters. The bin file "CALMAT.BIN" stores the extrinsic camera calibration parameters and "CHARTPOS.BIN" encodes the relative positions of the patterns used for calibration.

The user guide for the respective demo that you are calibrating for specifies which of these files are needed and how they are used.

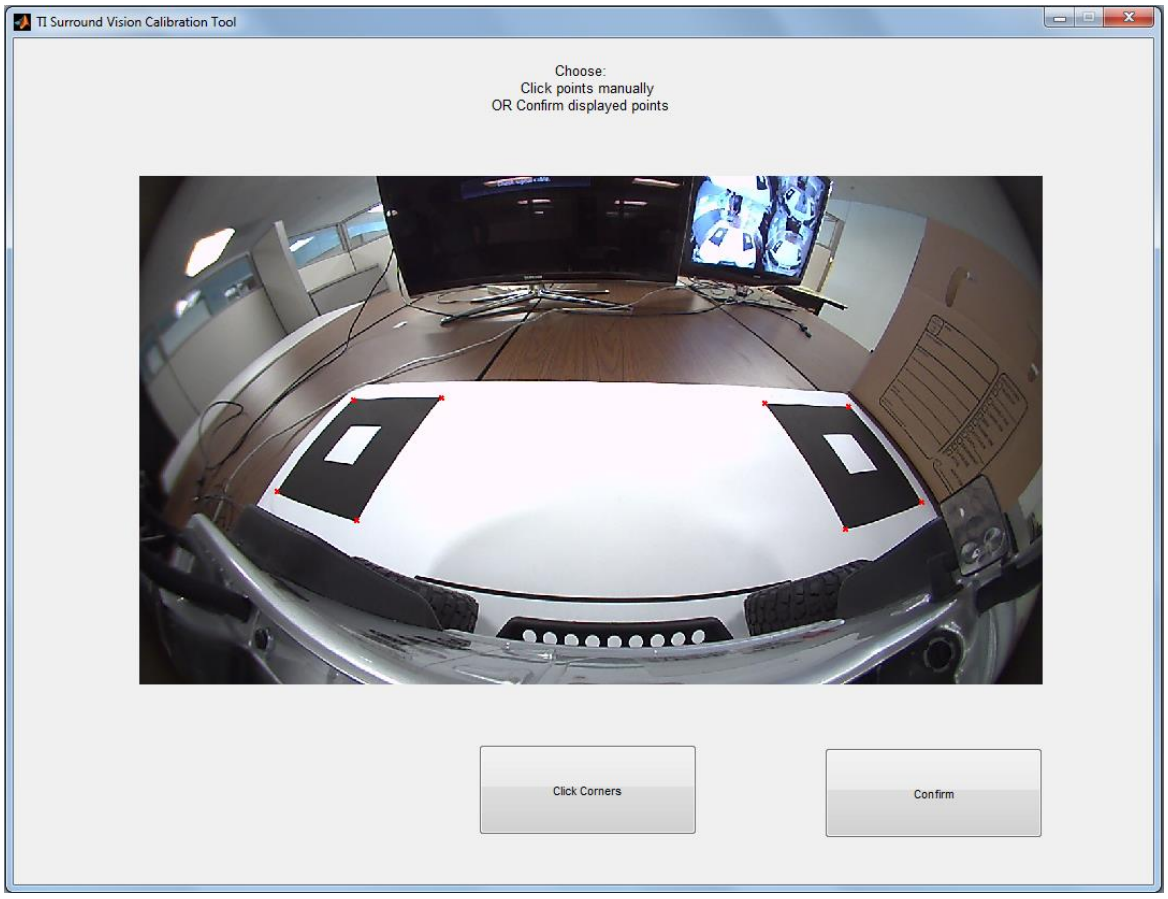

Figure 13: Step 4.3 – Confirmation of clicked points. If un-satisfied with clicked points for the image use the button "click corners" and redo corner selection

Figure 14: Step 4.4 - This message confirms that the camera calibration procedure is complete.

6.4 Generate Test output to validate calibration

Upon generating the calibration data, one can validate the result by creating a stitched top-down output. Once you return to the main screen, use the "Test Calibration" option shown in Figure 10. This will use the input images with the visible patterns and the generated extrinsic calibration parameters to synthesize a stitched top-down view. An example is shown below in Figure 15. If the output is unsatisfactory, one can run the calibration tool again to re-generate new calibration data.

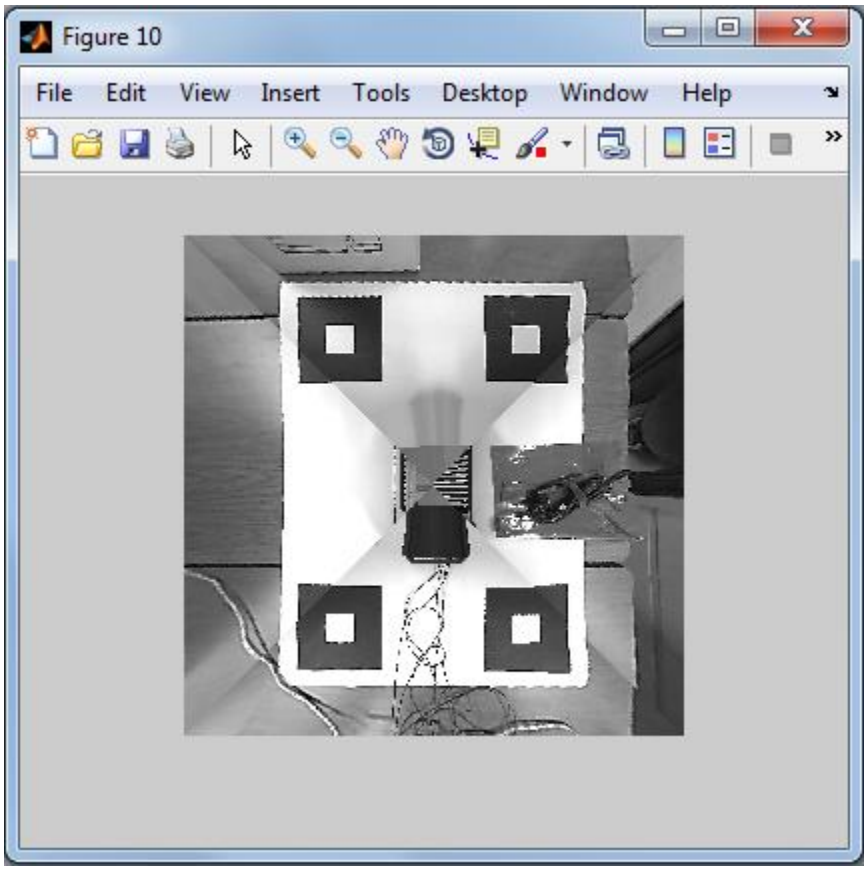

Figure 15: Step 4.5 - Test generated calibration to ensure correct output

7 Common Tasks to Be Accomplished

7.1 Create LENS.BIN File ("Intrinsic Camera/Lens Parameters") for VisionSDK

This task describes how to get the LENS.BIN files needed for VisionSDK implementation of TI's Surround Vision demos. Please read introductory sections 1 and 2 before proceeding.

Then, follow section according to camera modules that match your setup:

Error! Reference source not found. 2DSV Equisolid or TIDA00262

7.1.2 IMI DSSMR05

7.1.3 None of the above, and I know the distortion centers for the input images

7.1.4 None of the above, and I don't know the distortion centers for the input images

7.1.1 2DSV Equisolid or TIDA00262

Step 1: Complete section 3 including paths to input images

Step 2: Run Distortion Center Tool in section 4 to estimate distortion centers

Step 3: In section 5, use "Read LUT from CSV-File" and load ldc_lut_equisolid.csv (for 2D SV Equisolid) or ldc_lut_sunex_dsl218.csv (for TIDA00262) provided with this tool in folder ./ldc_luts.

Step 4: Press button "Generate LUT BIN-file". Find LENS.BIN in working directory

7.1.2 IMI DSSMR05

Step 1: Complete section 3 except paths to input images

Step 2: Use "Use Image Centers" for distortion centers in section 4

Step 3: In section 5, use "Read LUT from CSV-File" and load ldc_lut_imi.csv provided with this tool in folder ./ldc_luts.

Step 4: Press button "Generate LUT BIN-file". Find LENS.BIN in working directory

7.1.3 Custom camera module with known distortion centers

Step 1: Complete section 3 except paths to input images

Step 2: Use "Use Image Centers" or enter manually distortion centers in section 4

Step 3: In section 5, use "Run LUT Generation Tool" and follow section 5.1.

Step 4: Press button "Generate LUT BIN-file". Find LENS.BIN in working directory

7.1.4 Custom camera module with unknown distortion centers

Step 1: Complete section 3 including paths to input images

Step 2: Run Distortion Center Tool in section 4 to estimate distortion centers

Step 3: In section 5, use "Run LUT Generation Tool" and follow section 5.1.

Step 4: Press button "Generate LUT BIN-file". Find LENS.BIN in working directory

7.2 Create Extrinsic Calibration Results

Obtain/save calibration images with calibration charts present as described in section 6.

If you already completed section 7.1 before, make sure that paths to the calibration images are specified. All other data should be remembered and loaded automatically by the tool if you are using the same working directory.

If you haven't been through section 7.1, please do so with following additional instructions:

1. You have to provide paths to input calibration images
2. You do not need to press "Generate LUT C-file"

Then, follow section 6 to create Calmat.c, "CALMAT.BIN" and "CHARTPOS.BIN" in working directory.

Revision History

Version #	Date	Revision History
00.10	11/March/2016	First draft
00.20	27/June/2016	Output ldc_lut_tda2x.c and ldc_lut_tda3x.c instead of ldc_lut.c; Internal note: changed Calmat format from R=11.20/t=11.20 to R=1.30/t=21.10
00.30	09/Jan/2017	Output LENS.BIN file and ldc_lut.c (for backward compatibility). LENS.BIN format is specified in ldc_config.h file in VisionSDK. Add TIDA00262 camera module's precomputed csv file (ldc_lut_sunex_dsl218.csv) file
00.40	13/Jan/2017	Add format definition for ldc_lut.csv file

««« § »»»