Multi-sensor Fusion Platform Stereo Calibration Procedure

(v03.x)

User Guide

IMPORTANT NOTICE

Texas Instruments and its subsidiaries (TI) reserve the right to make changes to their products or to discontinue any product or service without notice, and advise customers to obtain the latest version of relevant information to verify, before placing orders, that information being relied on is current and complete. All products are sold subject to the terms and conditions of sale supplied at the time of order acknowledgment, including those pertaining to warranty, patent infringement, and limitation of liability.

TI warrants performance of its products to the specifications applicable at the time of sale in accordance with TI's standard warranty. Testing and other quality control techniques are utilized to the extent TI deems necessary to support this warranty. Specific testing of all parameters of each device is not necessarily performed, except those mandated by government requirements.

Customers are responsible for their applications using TI components.

In order to minimize risks associated with the customer's applications, adequate design and operating safeguards ought to be provided by the customer so as to minimize inherent or procedural hazards.

TI assumes no liability for applications assistance or customer product design. TI does not warrant or represent that any license, either express or implied, is granted under any patent right, copyright, mask work right, or other intellectual property right of TI covering or relating to any combination, machine, or process in which such products or services might be or are used. TI's publication of information regarding any third party's products or services does not constitute TI's approval, license, warranty or endorsement thereof.

Reproduction of information in TI data books or data sheets is permissible only if reproduction is without alteration and is accompanied by all associated warranties, conditions, limitations and notices. Representation or reproduction of this information with alteration voids all warranties provided for an associated TI product or service, is an unfair and deceptive business practice, and TI is neither responsible nor liable for any such use.

Resale of TI's products or services with *statements different from or beyond the parameters* stated by TI for that product or service voids all express and any implied warranties for the associated TI product or service, is an unfair and deceptive business practice, and TI is not responsible nor liable for any such use.

Also see: Standard Terms and Conditions of Sale for Semiconductor Products.
www.ti.com/sc/docs/stdterms.htm

Mailing Address:

Texas Instruments
Post Office Box 655303
Dallas, Texas 75265

TABLE OF CONTENTS

1 Introduction

This user's guide details the calibration procedure for each of the sensor present in the TDA2xx based Multi-sensor fusion platform (MonsterCam / MSF).

2 Calibrating the stereo cameras

Monstercam is equipped with two RCCC sensors used by the stereovision algorithm to produce a disparity map of the field of view of both sensors. The lens covering each sensor have distortions that will cause errors in the stereovision algorithm. Also if both sensors are misaligned then the algorithm will produce a bad disparity map.

Both lens distortions and misalignment need to be corrected by a process called rectification. The rectification is handled by a warping algorithm and is performed in real-time. The warping algorithm takes a pixel remap lookup table as parameter to remap each pixel of its input to a new position in the output.
The calibration process consists of finding this remap lookup table and is composed of 2 steps:

1. Use Monstercam to capture of several (15) left and right images in which a checkerboard pattern is shown at different positions and angles. Each image is saved into a **.pgm* format.
2. Execute the *camAutoCalib.exe* application, which automatically corrects lens distortions first and then corrects misalignments based on the images sequence taken from the previous step. The result of this step are 4 files:
   rectMapRight_int_converted.c, *rectMapLeft_int_converted.c,*
   rectMapRight_int_converted.bin, *rectMapLeft_int_converted.bin.*
   The first two files *rectMapRight_int_converted.c*, *rectMapLeft_int_converted.c* can be used to set the remap lookup tables at compile time. These tables are used by the warping algorithm running on EVE to rectify the input images. The two *.c files must be copied to the directory *vision_sdk\examples\tda2xx\src\alg_plugins\remapmerge* and the vision SDK must be rebuilt in order for the new mapping to take effect.
   The last two files *rectMapRight_int_converted.bin*, *rectMapLeft_int_converted.bin* can be used to update the tables without having to rebuild the codebase, by writing into the target board's flash memory through the network utility.

2.1 Capturing images from Monstercam

- Print the following checkerboard/chessboard pattern: http://docs.opencv.org/_downloads/pattern.png and glue it to a flat cardboard or something flat and rigid.

- Measure the (X,Y) dimensions of each square in the checkerboard, in mm. You will need this information to be passed to the configuration file of the *camAutoCalib.exe* application.

- Place the checkerboard in front of Monstercam at some angle and select **"Stereo Use-cases, (TDA2x MonsterCam ONLY)"** from runtime menu on UART console and run usecase **"2CH VIP capture + SoftISP + Remap + Display - USED for Stereo Calibration".**

- Use `stereo_calib_image_save` command in network_ctrl tool (*VISION_SDK\tools\network_tools\bin*) to save images from the left and right sensor into **.pgm* files.

  The syntax of the command is:

  ```
  network_ctrl --ipaddr <ipaddr> [--port <server port>] --cmd
  stereo_calib_image_save <filename_prefix>.
  ```

- Filenames produced are in the format *left_<filename_prefix>.pgm* and *right_<filename_prefix>.pgm* . The simplest values for <*filename_prefix*> would be “00”, “01” ,”02”, etc. Note that the numbering must have at least two digits in order for the application *camAutoCalib.exe* to successfully go through the sequence of saved images.

  Example:

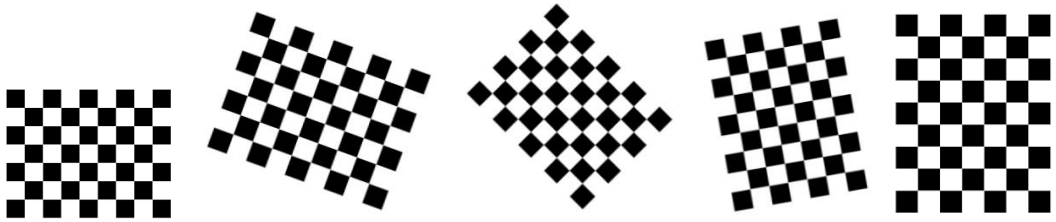
```
network_ctrl --ipaddr 192.168.1.2 --cmd stereo_calib_image_save 00
network_ctrl --ipaddr 192.168.1.2 --cmd stereo_calib_image_save 01
network_ctrl --ipaddr 192.168.1.2 --cmd stereo_calib_image_save 02
…
```

  Will generate files *left_00.pgm, right_00.pgm, left_01.pgm, right_01.pgm, left_02.pgm, right_02.pgm, etc .*

  For more details, refer Section ‘2’ in *VisionSDK_NetworkTools_UserGuide.docx* document (*VISION_SDK\tools\network_tools\docs*) for usage and tool details.

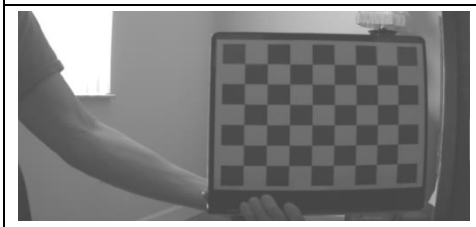
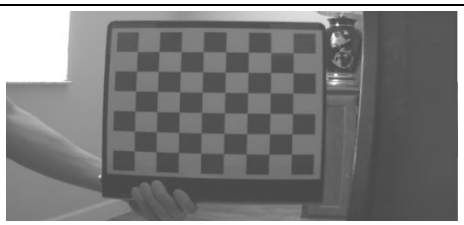
  Multiple snapshots (10 to 15) are required to guarantee a good calibration. The checkerboard pattern must be presented in various poses, covering as many locations within the field of view and at different distances from the camera. The camera must remain at the same position throughout the snapshots, only the checkerboard can be moved around. Place the checkboard at different positions within the field of view and at different distances from the camera. Avoid poses in which the checkerboard angle is too close to 90 degrees.

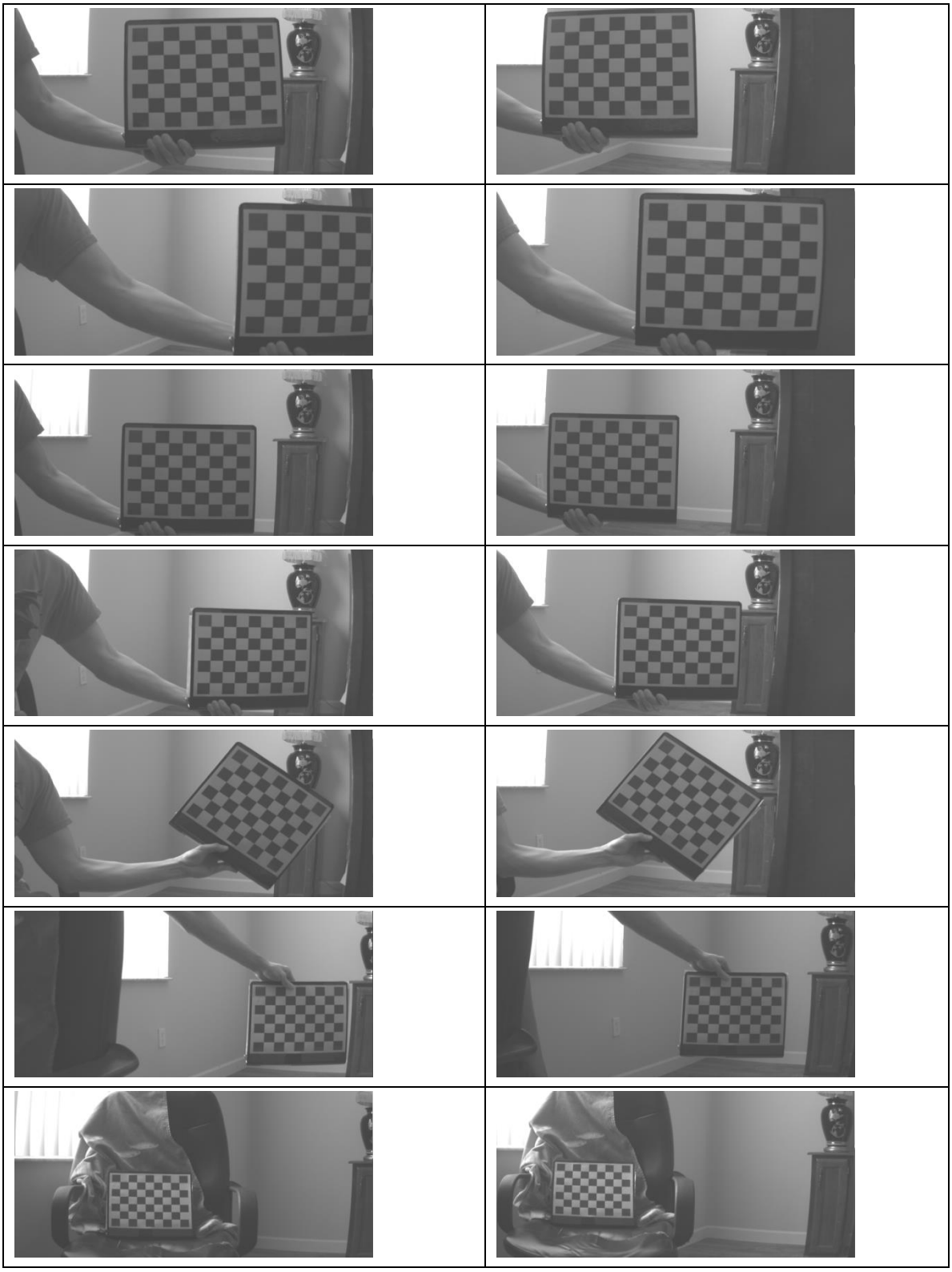
  The figures below show what would be considered as good or bad poses.

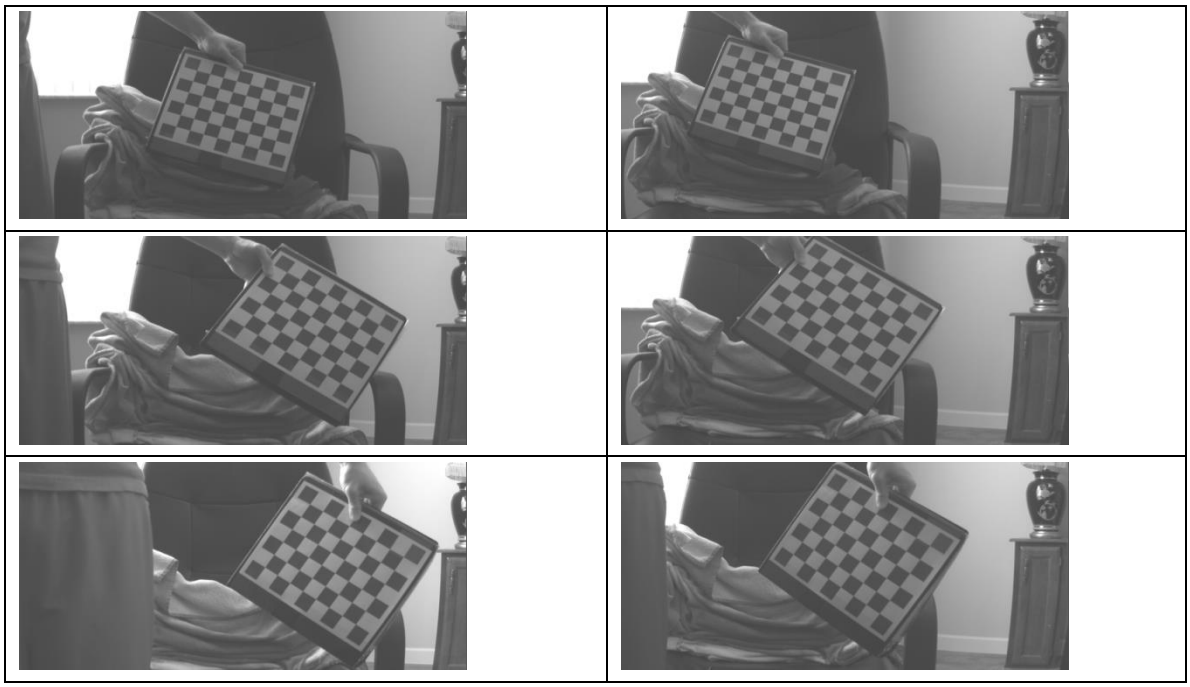


GOOD GOOD FAIR BAD BAD

For reference, similar poses to the following can be used:

Left	Right
	





2.2 Usage of camAutoCalib.exe

2.2.1 Setup

The following steps need to be performed before invoking *camAutoCalib.exe*:

1. Copy the following files from *vision_sdk/tools/camera_calibration_tools/bin* to the same directory where the image sequence is located at:

   - *camAutoCalib.exe*: application based on openCV functions `findChessboardCorners()`, `findCircleGrid()`, `cornerSubPix()`, `calibrateCamera()`, `computeReprojectionError()`, `stereoRectify()` to correct the lens distortion and left-right camera misalignment. The source file *camAutoCalib.cpp* is also provided in case one wants to customize the code and rebuild the final application [(3)].

   - *calib.xml*: configuration file in which calibration parameters and path to the image sequences are provided. This file contains default parameter settings and next paragraph will explain how to set these.

   - *remapConvertTable.eve.out.exe*: executable invoked by *camAutoCalib.exe* to generate files *rectMapRight_int_converted.c* and *rectMapLeft_int_converted.c.*

   - *remapExecute.out.exe*: executable invoked by *camAutoCalib.exe* to validate the generated lookup tables by applying the PC built version of the EVE warping function called "*remap*" to the first pair of stereo images in the sequence. For proof-checking purpose, it generates files *eve_test_left.pgm* and *eve_test_right.pgm* that can be viewed to validate the quality of the rectification.

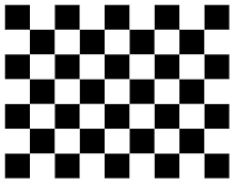
2. Install openCV because the executable *camAutoCalib.exe* needs openCV's DLLs to execute. The application was validated with openCV 2.4.10, however subsequent version should also work as long as the functions used keep the same interface.
   Also make sure that you have the environment variable *OPENCV_DIR* set to *c:\<OPENCV_INSTALL_DIR>\build\x86\vc12* otherwise Windows will complain that it cannot find the DLLs when *camAutoCalib.exe* is executed.

2.2.2 Using camAutoCalib.exe

There are two steps:

2.2.2.1 Editing calib.xml

- Open and edit the following fields in *calib.xml*:
  - `Board_Width, Board_Height`: number of internal corners per item row and column. An internal corner is defined as a point where the black squares touch each other. For instance for this chessboard:



We have `Board_Width= 8` and `Board_Height= 6`.

  - `Square Size`: length of one side of a square in meters, example 0.030 for 30 mm square size.

  - `Use_Hints`: 1 means use initial hints for focal length and center's position. The values for these hints are contained in the subsequent fields `FocalX_Hint, FocalY_Hint, CenterX_Hint, CenterY_Hint`. If the lens characteristics are known, initializing the hint values will produce better calibration results. 0 means ignore the hint values.

  - `FocalX_Hint`: hint value for the focal length along X direction.

  - `FocalY_Hint`: hint value for the focal length along Y direction.

  - `CenterX_Hint`: hint value for the lens' center's X coordinate.

  - `CenterY_Hint`: hint value for the lens' center's Y coordinate.

  - `Reproj_Error_Thresh`: After a first calibration iteration, frames that have a reprojection error greater than this threshold are discarded before a second calibration iteration is executed. The process repeats until there is no frame discarded.

  - `Calibrate_Pattern`: can be set to CHESSBOARD, CIRCLES_GRID or ASYMMETRIC_CIRCLES_GRID.

    Printout are available for download from the web:

    Chess board:
    http://docs.opencv.org/_downloads/pattern.png

assymetric circles grid:
http://robocraft.ru/files/opencv/acircles_pattern.png

- `InputLeft:` filename pattern for the left image sequence. Default value is "*left_%02.pgm*" allowing to process up to 100 images. Note that there should not be any gap in the numbering of the image sequence. For instance the image sequence *left_00.pgm*, *left_01.pgm*, *left_02.pgm*, *left_04.pgm*, *left_05.pgm*, *left_06.pgm* has a gap because *left_03.pgm* is missing. This will result in having only images *left_00.pgm* to *left_02.pgm* being processed.

- `InputRight`: filename pattern for the right image sequence. Default value is "*right_%02.pgm*" allowing to process up to 100 images. Note that there should not be any gap in the numbering of the image sequence.

- `Input_FlipAroundHorizontalAxis`: in case the sequence of images was captured upside-down, a '1' in this field will enable flipping the images back to their correct orientation. Default value is '0'.

- `Calibrate_NrOfFrameToUse`: maximum number of frames which will be processed by *camAutoCalib.exe*. This field is used to only inform the application about the maximum number of frames. The sequence of images pointed by `InputLeft` and `InputRight` doesn't have to contain as many frames. Default value is 100.

- `Calibrate_Stereo`: Enable stereo calibration in addition to lens distortion correction. The latter is always performed whereas stereo calibration is optional in order to allow mono-camera calibration. Stereo calibration does left and right camera epipolar alignment. Default value is '1'. If value '0' is set then no stereo calibration is done and the generated map files will correct for lens distortion effects only. This is desirable in the case of mono-camera lens distortion in which `InputLeft` and `InputRight` would be set to the same image sequence.

- `Stereo_alpha_factor`: free scaling parameter. '0' means rectified images are zoomed and shifted so that only valid pixels are visible (no black areas after rectification).
  '1' means that the rectified image is decimated and shifted
  so that all the pixels from the original images from the cameras are retained in the rectified images (no source image pixels are lost).
  Obviously, any intermediate value yields an intermediate result between those two extreme cases. Default value is '0' so that the rectified image fills the same area covered by the input resolution. Otherwise black pixels or undefined pixels would fill some border regions of the rectified frame.

- `GenerateEveRemapFiles`: '1' means generate the files *rectMapRight_int_converted.c* and *rectMapLeft_int_converted.c*, which are compatible with EVE remap. These files must be copied to the directory *vision_sdk\examples\tda2xx\src\alg_plugins\remapmerge* before vision SDK is rebuilt. Default value is '1' of course as this is the final purpose of the calibration. You can set it to '0' in case some issues are encountered and need to debug.

- `RemapBlockWidth` and `RemapBlockHeight`: the EVE remap function processes the image in a block-wise fashion. Thus dimensions of the output block must be provided. The larger the block, the faster the processing is

because the control overhead is smaller. However the amount of EVE local memory limits how large the block can be and a large value for `RemapBlockWidth` or `RemapBlockHeight` can cause the calibration to fail and return an error to the console window. Default value `RemapBlockWidth`=128 and `RemapBlockHeight`= 16 should work for most of the cases. If the distortions are severe such as for fish-eye lens then try `RemapBlockWidth`= 64 and `RemapBlockHeight`= 8.

- `RemapColorFormat:` Color format which will be processed by remap, 0: Unsigned 8-bits Gray, 1; Signed 8-bits Gray, 2: Unsigned 16-bits Gray, 3: Signed 16-bit Gray, 4: YUYV, 5: UYVY, 6: YUV420 SP .

- `OutputMap_qShift`: amount of left shift applied to the coordinates saved in the remap tables. This is to increase precision and to allow fractional coordinates. Typically `OutputMap_qShift`=2. For instance, coordinate (X= 80.56, Y=100.25) would then be scaled to (X=4*80.56, Y=4*100.25)= (X=322.24, Y=401). Then only the rounded integer part is kept and what is written into the map files is (X=322, Y=401). This shift amount information will be embedded into *rectMapRight_int_converted.c* and *rectMapLeft_int_converted.c* so the EVE remap function knows by how much it needs to downshift in order to derive the real coordinates.

- All the other fields `Write_leftOutputFileName`, `Write_rightOutputFileName`, `OutputMap_qShift`, `Write_leftOutMapFileName`, `Write_rightOutputMapFileName`, `Write_DetectedFeaturePoints`, `Write_extrinsicParameters`, `Show_UndistortedImage` can be left to their default values already present in file *calib.xml* .

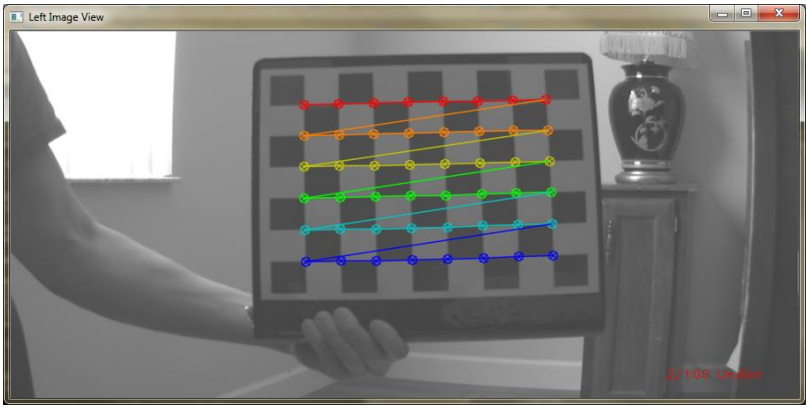
2.2.2.2 Executing camAutoCalib.exe

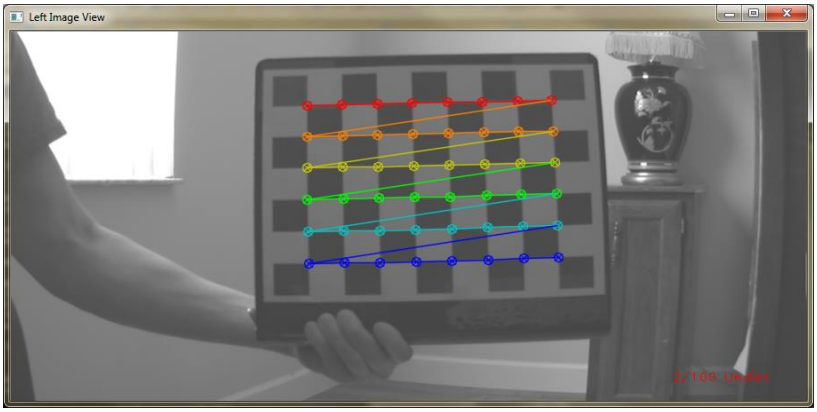
Simply open a command line window and go to the directory where the image sequences are and type `‘camAutoCalib.exe calib.xml ‘.`
Wait a few seconds and you should find the generated files *rectMapLeft_int_converted.c* and *rectMapRight_int_converted.c* in the same directory. Just copy them into the directory *vision_sdk\examples\tda2xx\src\alg_plugins\remapmerge* and then rebuild vision SDK.
This is all what it takes to calibrate the stereo cameras and generate the associated warping table.
Note that a more interactive version can be launched via:
`‘camAutoCalib.exe calib.xml -i‘`
This version will pause between each frame until a key is pressed. This is to give time for the user to inspect the quality of the corner detection.

2.2.2.3 Description of camAutoCalib.exe

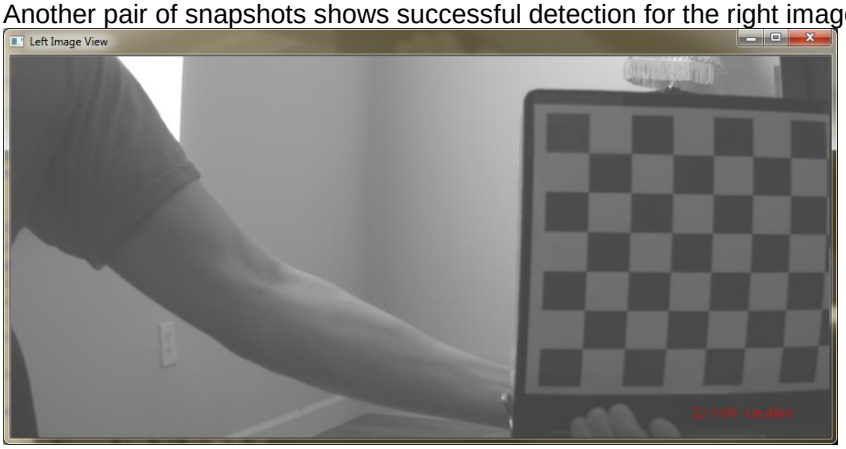
This paragraph provides some technical details on how `camAutoCalib.exe` works. Reader can skip this paragraph in the interest of time.
Basically, *camAutoCalib.exe* will go through each pair of left/right images following the filename’s pattern specified in the file *calib.xml*. For each pair, it first looks for the location of the internal corners and displays the result of the detection in two windows: one for left and one for right camera. Depending on the checkerboard pattern’s position, detection may not always be successful for both view. To be successful all the `Board_Width x Board_height` corners must be detected. The lower right corner of each window displays the number of images with successful detection of corners against the maximum number of frames to be processed. This number should increment as more stereo pairs are processed.

Here are snapshots for the case when corner detection is successful for both images in the pair:

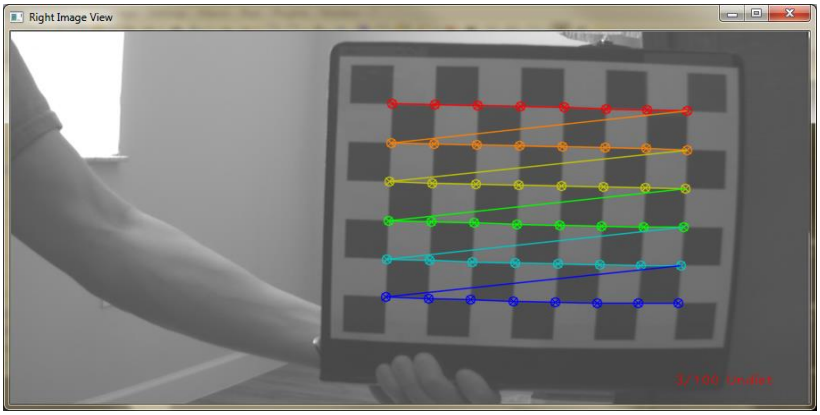




Another pair of snapshots shows successful detection for the right image only:



In the left view above, the checker-board is not entirely visible within the frame, causing unsuccessful detection of all the 8x6 corners.



Because of the wide baseline of the monstercam' stereo cameras, there will be times where in one of the view, the checker-board pattern doesn't fit. For the lens distortion correction, since each camera is independently corrected, having unequal corner detection outcomes between left and right camera is not an issue. But later, for the stereo calibration, we need to have as many successful detected pairs as possible. Usually 2-3 undetected frames out of 10, per view, is acceptable.

Once corner positions of every pair in the sequence are found, the utility will compute each camera's extrinsic and intrinsic parameters by calling the openCV function `calibrateCamera()`. After that, the console window should display the following two lines:

```
Left Calibration succeeded. avg re-projection error = ????
Right Calibration succeeded. avg re-projection error = ????
```

The error values depend on the amount of distortion present in the lens but shouldn't exceed 0.2 (20%). If they do, then either not enough snapshots were taken or many of them had unsuccessful corners detection. In that case, a new sequence of snapshots should be taken.

If the field `Calibrate_Stereo` in *calib.xml* is '1' then afterwards, stereo calibration is performed to correct misalignment between left and right cameras. The type of misalignment which can be corrected includes rotation, translation. The console window should display the outcome of the stereo calibration:

```
Running stereo calibration ...
done with RMS error= ????

Valid ROI for left image @(x=???, y=???), width=???, height=???
Valid ROI for right image @(x=???, y=???), width=???, height=???
Common ROI for both images @(x=???, y=???), width=???, height=???)
```

Error should be below 0.2 (20%).

The valid ROIs are output rectangles inside the rectified images where all the pixels are valid. Indeed due to the distortion correction, some pixels in the rectified image map to non-existent pixels in the input. Besides, left and right images can have different ROIs, especially if the lens are not very well aligned.
The dimensions of these valid ROIs directly depend on the value of the parameter `Stereo_alpha_factor`.
If `Stereo_alpha_factor=0` , the dimensions of the valid ROI are equal to the input image due to some internal scaling which resizes the valid region up to the dimensions of the input images. Any other value between 0 and 1 will generate ROIs whose dimensions are smaller than the input images.

After stereo calibration is done, if `GenerateEveRemapFiles=1` then `rectMapLeft_int_converted.c` and `rectMapRight_int_converted.c` are generated by calling *remapConvertTable.eve.out.exe*. These files are used with EVE's remap function and needs to be copied to the directory *vision_sdk\examples\tda2xx\src\alg_plugins\remapmerge* before rebuilding

vision SDK. Note that *remapConvertTable.eve.out.exe* must be present in the same directory as *camAutoCalib.exe* for this to work.
In addition to these two C files, the equivalent binary files *rectMapRight_int_converted.bin* and *rectMapLeft_int_converted.bin* are produced. These files can be used to update the tables without having to rebuild the codebase, by writing into the target board's QSPI flash memory through the network utility. Please refer to the document *VisionSDK_NetworkTools_UserGuide.pdf* for more details.

The console window should display after successful execution:

```
Table conversion to EVE format successful: tables converted into files
rectMapLeft_int_converted.c and rectMapRight_int_converted.c
Rectified left and right images of the first stereo pair generated into
eve_rect_test_left.pgm, eve_rect_test_right.pgm and available for inspection
Visualization of rectified left and right views enabled. Press any key to
advance to the next set of views or 'q' or ESC to quit the visualization
```

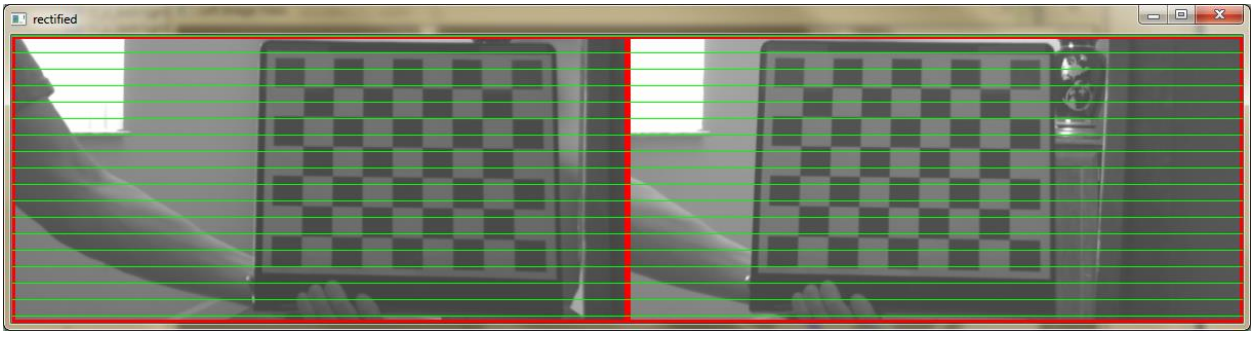
Sometimes the conversion fails and the following message is displayed:

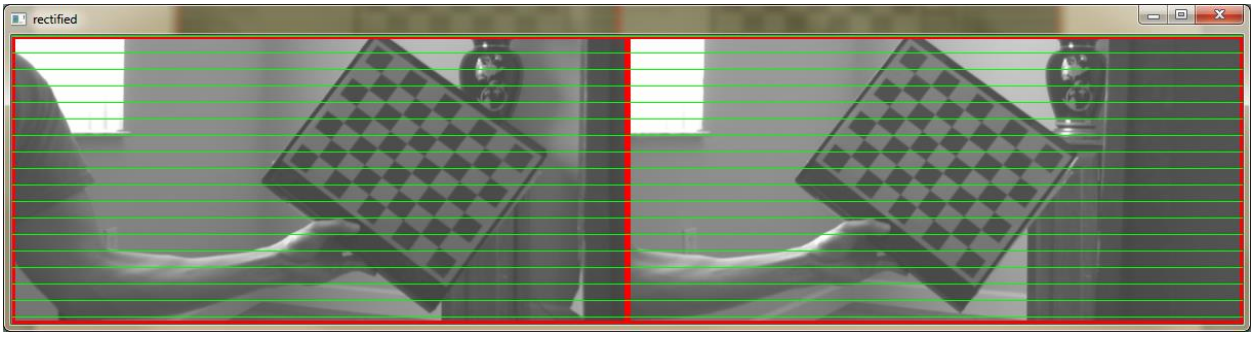
```
Remap initialization error: decrease blockWidth and blockHeight values for
the use case processed by remapConvertTable
Error use case #0, stopped processing
Remap initialization error: decrease blockWidth and blockHeight values for
the use case processed by remapConvertTable
Error use case #1, stopped processing
Table conversion to EVE format failed, please try to reduce parameters
RemapBlockWidth or RemapBlockHeight in the xml configuration file
```

As the message says, to remedy this error, the parameters blockWidth and blockHeight must be reduced.

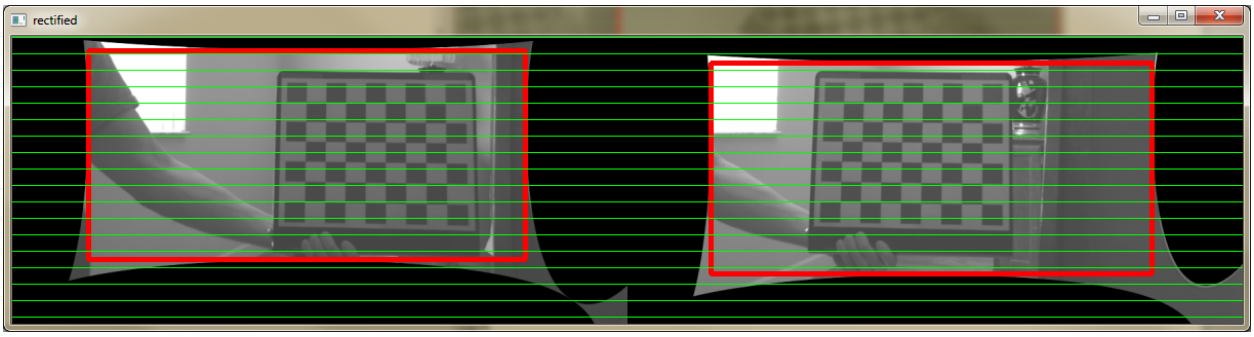
In addition to producing these converted tables, rectified left and right images of the first stereo pair of the image sequence are also generated into the files *eve_rect_test_left.pgm and eve_rect_test_right.pgm* . The user can check the quality of the rectification by inspecting these files. For this to work, *remapExecute.out.exe* must be present in the same directory as *camAutoCalib.exe.*

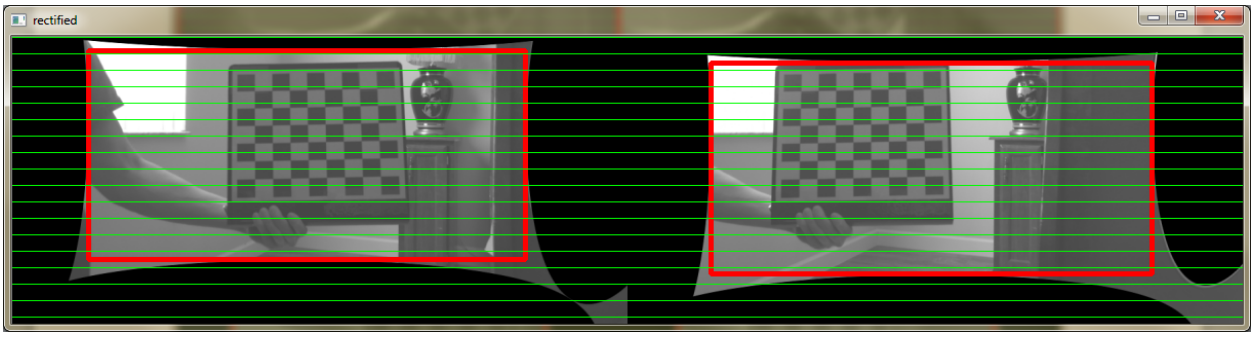
Finally, to assist further in validating the quality of the rectification, each rectified stereo pair is also shown on the monitor, with a grid of green epipolar lines. Here are a few snapshots of the validation windows:





Note that a red rectangle represents a valid ROI. The snapshots above correspond to `Stereo_alpha_factor` =0. For `Stereo_alpha_factor` =1, we would obtain the following outputs:





These two snapshots clearly illustrate the fact that the valid ROI red rectangles are smaller than the input image's dimensions.

The user can exit the visualization of the stereo outputs at any time by pressing 'q' or 'ESC'. All the output files are already saved so there is no bad consequence of quitting the visualization part.

3 Additional notes

- Note that after using the network utility to flash the content of *rectMapRight_int_converted.bin* and *rectMapLeft_int_converted.bin* to the target board's QSPI flash memory, any tables contained in the files *rectMapLeft_int_converted.c* and *rectMapRight_int_converted.c* will always be ignored. The priority is always given to the tables located in the QSPI flash memory instead of the ones provided in the C files. The only exception is after an erasure of the flash memory, the system selects the ones specified by the C files since the tables have been deleted from

QSPI. To help identifying which set of tables are used at runtime, the Vision SDK demo will display the following message on the console window whenever a stereo-vision use case is executed. If the tables have been flashed to QSPI, these messages are displayed:

```
ALGORITHM: RemapMerge is using right LUT stored in QSPI, table
in rectMapRight_int_converted.c is ignored.

ALGORITHM: RemapMerge is using left LUT stored in QSPI, table
in rectMapLeft_int_converted.c is ignored.
```

If instead, there isn't any table in QSPI, these messages are displayed:

```
ALGORITHM: RemapMerge is using right LUT stored in QSPI, table
in rectMapRight_int_converted.c is ignored.


ALGORITHM: RemapMerge is using left LUT stored in QSPI, table
in rectMapLeft_int_converted.c is ignored.
```

- The following extra files are generated into the directory but they can be ignored: *rightInputMap_int.bin, rectMapRight_int_converted.bin, leftInputMap_int.bin, rectMapLeft_int_converted.bin, eve_test_left.pgm, eve_test_right.pgm, eveRemapExecute_config.txt, eveRemapExecute.cfg, eveRemapConvertTable.cfg, eveRemapConvertTable_config.txt*.
- The files *out_left_camera_data.xml and out_right_camera_data.xml* can be of interest as they contain the camera intrinsic and extrinsic parameters after lens distortion correction but before stereo rectification.
- Since *camAutoCalib.exe.cpp* is provided, one can modify and rebuild *camAutoCalib.exe* using Microsoft Visual C++ or other code generation tools. Only requirement is to have openCV installed. The openCV static libraries (in *OPENCV_DIR/build/x86/vc12/staticlib*) that are required to be linked are: *opencv_features2d2411.lib, opencv_highgui2411.lib, opencv_calib3d2411.lib, opencv_core2411.lib, opencv_imgproc2411.lib, opencv_ml2411.lib, libtiff.lib, zlib.lib, libpng.lib, IlmImf.lib, libjasper.lib, libjpeg.lib, opencv_flann2411.lib*. In addition two Microsoft libraries: *vfw32.lib* and *comctl32.lib* are required.

  It is probable that later versions of openCV libraries will also work but it has not been tried yet.

4 References

(1) http://docs.opencv.org/doc/tutorials/calib3d/camera_calibration/camera_calibration.html
(2) http://docs.opencv.org/doc/tutorials/introduction/windows_install/windows_install.html
(3) http://docs.opencv.org/doc/tutorials/introduction/windows_visual_studio_Opencv/windows_visual_studio_Opencv.html#windows-visual-studio-how-to
(4) http://www.vision.caltech.edu/bouguetj/calib_doc/

5 Revision History

Version	Date	Revision History

1.0	15 April 2015	Initial Version from Victor
3.2	24 May 2015	Mention of loading LUT tables to flash memory.
3.3	8th July 2015	Updated for usecase selection
3.4	22nd August 2016	In additional notes: clarified when the system uses the LUT from QSPI flash memory instead of those provided at compile time as C files.