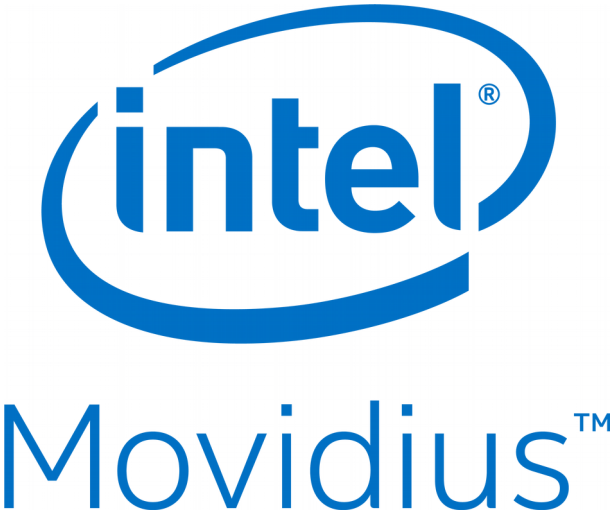


Myriad 2 Stereo Modules Calibration

Application Note

v1.0 / June 2018

Copyright and Proprietary Information Notice

Intel® Movidius™
2200 Mission College Blvd
M/S SC11-201
Santa Clara, CA 95054
http://www.movidius.com/

Revision History

Date	Version	Description
June 2018	1.0	Initial version.

Table of Contents

1 Introduction

In order to produce clean disparity maps and calculate accurate depths from a stereo depth solution, good (stereo) calibration is essential. Stereo calibration involves the determination of the rotation and translation of the second camera with respect to the first camera.

2 Requirements

In order to run the stereo pipeline, the following are needed:

1. Calibration checkerboard pattern.
2. Uncalibrated images captured with the checkerboard pattern.
3. Calibration tool (requires any OpenCV higher than 3.1).
4. Stereo demo.

The following sections will describe in detail the requirements for each step. As well as tips to achieve the best calibration result.

Package contents:

- stereoCapture: application for capturing stereo images.
- Rectification: application for calibration that uses the captured stereo images.
- MDK patch.

3 Checkerboard Pattern

A 6x9 calibration pattern can be found in the appendix on page 12 or online here. Print the checkerboard out in an A4 landscape orientation and measure the edges of a few squares. All measurements should be equal, this will ensure the tiles are square. Record the measurement as it will be needed for calculations later. Take care to fix the checkerboard pattern to a flat surface. Imperfections on the surface or the pattern can affect the accuracy of the calibration.

NOTE: Make sure to use a 6x9 corners calibration pattern. A corner is defined as the point of intersection between two squares. The figure below illustrates how the corners are counted for one axis.

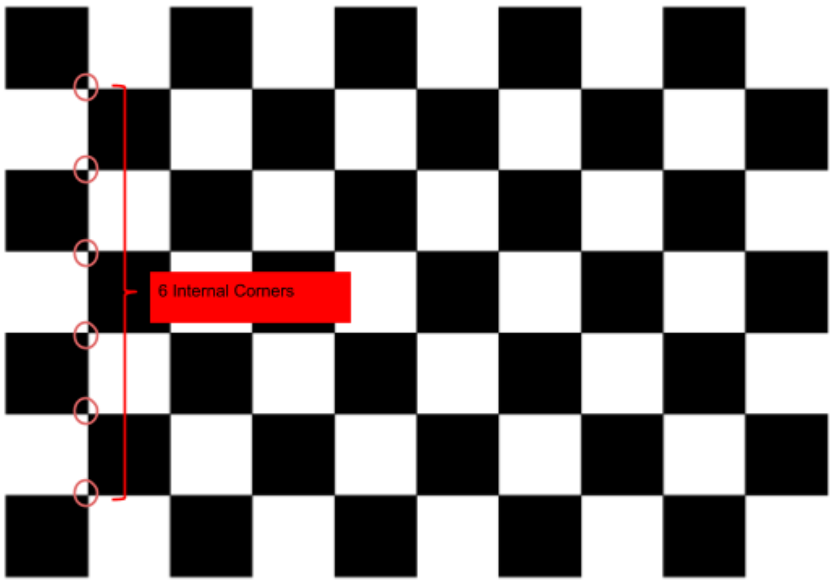


Figure 1: Corner counting for one axis

4 Stereo Image Capture

For greater calibration accuracy, follow these instructions for setting up the cameras, and capturing the images:

- Use a tripod to place the camera on.
- Keep the pattern visible in both left and right images. The pattern should occupy ~80% of the image.
- Use a variety of checkerboard pattern poses, at least 10 (see Figure 2).
- Images are taken in pairs (left and right camera) and images from each pair should be synchronized (take images at the same time).

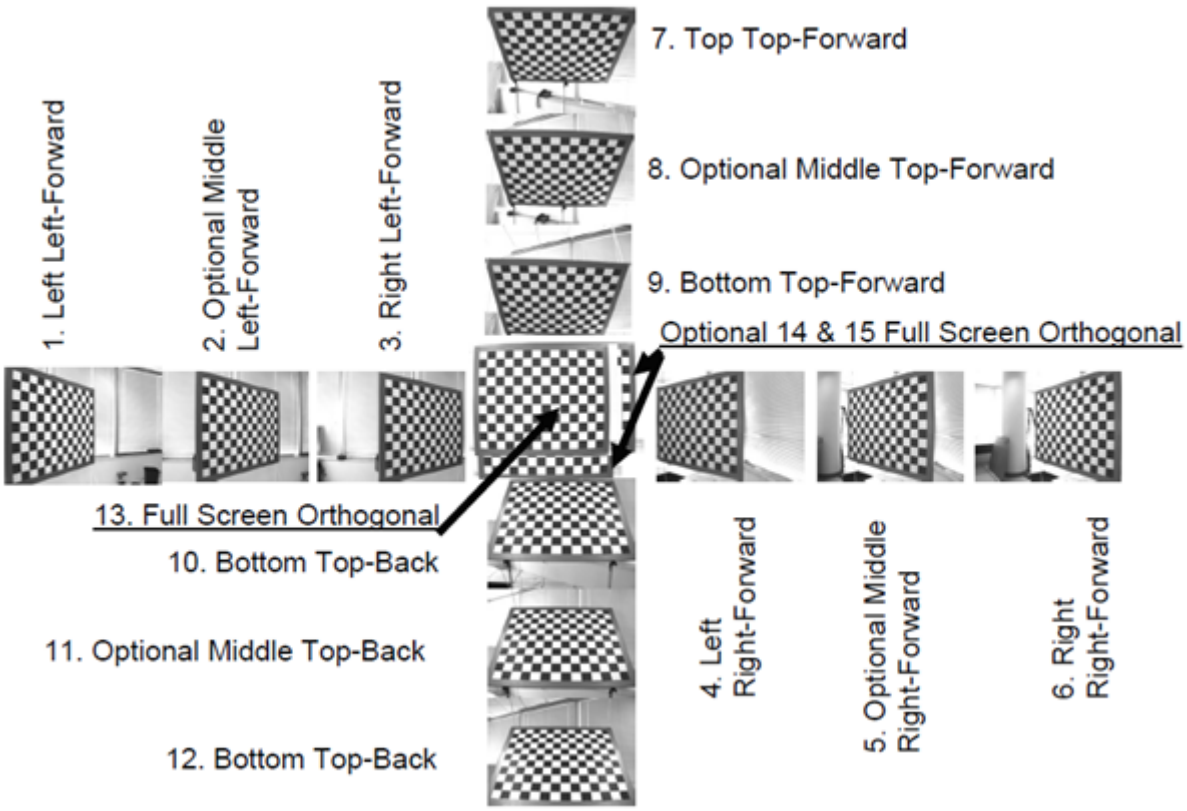


Figure 2: Checkerboard pattern poses

➢ **Good Capture Example**

Given the rules above are respected, a good stereo pair should look as in the example in Figure 3.

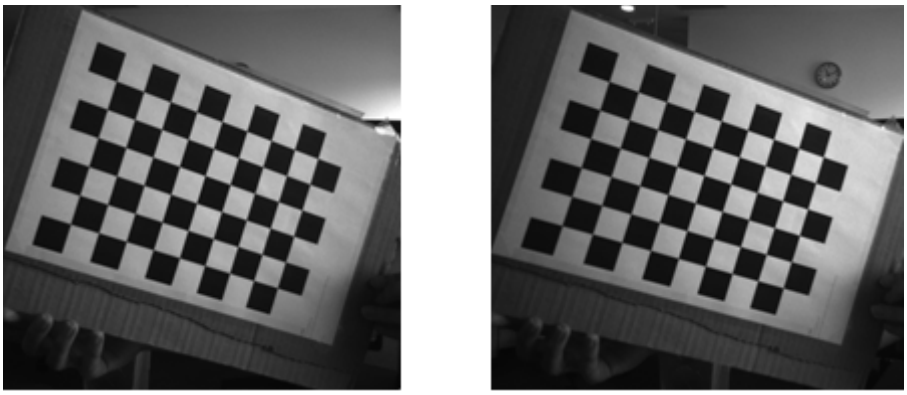


Figure 3: Good Stereo Capture Example

4.1 Stereo Image Capture Tool

Make the following steps:

1. Unpack the archive calibrationM2V4_0.2.zip
2. Apply MDK patch:

   `# sh extract.sh ../mdk_release_/mdk/common/` (run at location of `extract.sh`)
3. Run Myriad 2 streaming app:

   ```
   # cd mdk/projects/DataCollectionSystem/apps/DataCollectionSystem
   # make start_server
   # make debug
   ```
4. Open another terminal and run PC viewer app:

   ```
   # cd mdk/projects/DataCollectionSystem/apps/pcUsbTool
   # cmake .
   # make
   # ./dataShower
   ```

   A stereo stream will be displayed. Move the pattern in different poses, as instructed in chapter 4. When finished, press Esc.

   The images will be saved in the `pcUsbTool/output/` directory.

5 Calibration Tool

The offline calibration tool is based on OpenCV's API `cv::stereoCalibrate` and `cv::stereoRectify` that computes meshes for the Left and Right cameras. Both Left and Right images will be calibrated.

Before building, please create 2 text files where you select what images to calibrate.

Given the images were captured in real time, you may want to jump through indexes in order to reach all the poses. For example, don't use cam0_0, cam0_1, etc. Recommendation is to use: cam0_10, cam0_30, etc. Also, don't use the index 0 poses, as the auto-exposure does not work very well for the first couple of frames, until it gets initialized.

```
cam0List.txt
      cam0_10.png
      cam0_40.png
      …
cam1List.txt
      cam1_11.png
      cam1_41.png
      …
```

The mandatory parameters are:

- `-c0 Cam0ImgList`: file containing the first camera image set.
- `-c1 Cam1ImgList`: file containing the second camera image set.
- `-nx nb_cX`: checkerboard number of squares in X direction [e.g. 9].
- `-ny nb_cY`: checkerboard number of squares in Y direction [e.g. 6].
- `-ss squareSize`: checkerboard side size in centimeters [e.g. 2.6].

```
# cd calibrationM2V4_0.2/Rectification
# mkdir build
# cd build
# cmake ../
# make
# ./Rectification -c0 cam0List.txt -c1 cam1List.txt -nx 9 -ny 6 -ss
2.5
```

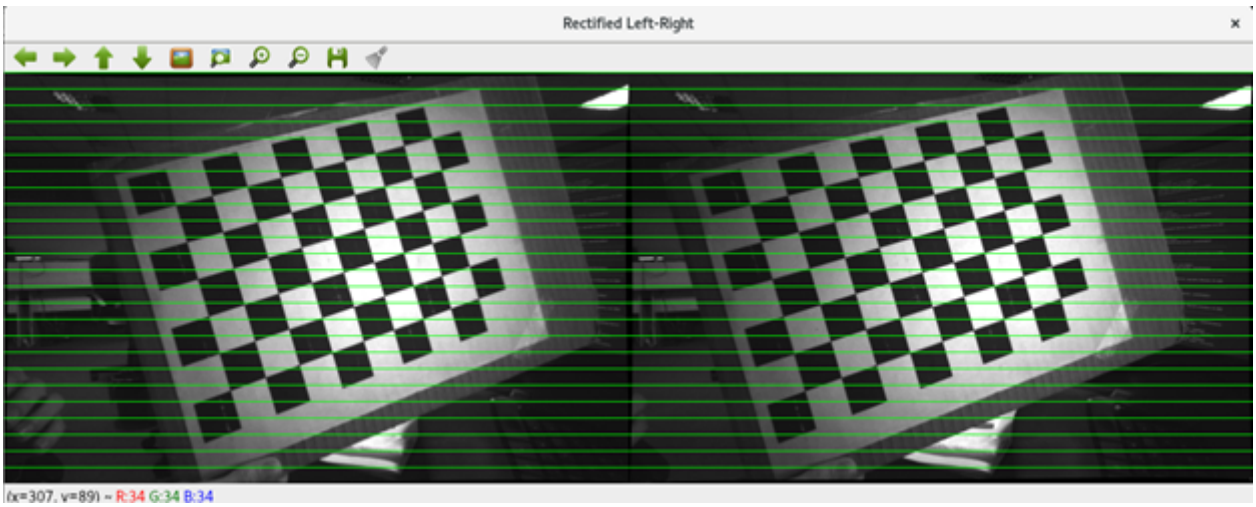
When running the rectification application, a display will be shown where the user can check visually the rectified images. Press space until all pairs have finished displaying.

5.1 Calibration Tool Output

➢ **Example: good rectification output**

```
./Rectification -c0 cam0_ov9282.txt -c1 cam1_ov9282.txt -nx 9 -ny 6 -
ss 2.5

--> Stereo system rectification...
Compute 2D pattern corners from images... Order test OKAY... Done!
Compute 3D pattern points... Done!
Running calibration by Bouguet's method
      Compute camera matrix... Done!
      Compute stereo calibration... done with RMS error=0.362567
      Check calibration quality... average epipolar err = 0.325351
      Compute stereo rectification... Done!
      Save intrinsics... Done!
      Save extrinsics... Done!
      <-- Done!
Save dense rectification maps... Done!
Save rectification maps as meshes... Done!
Show stereo calibration results... Done!
DONE
```

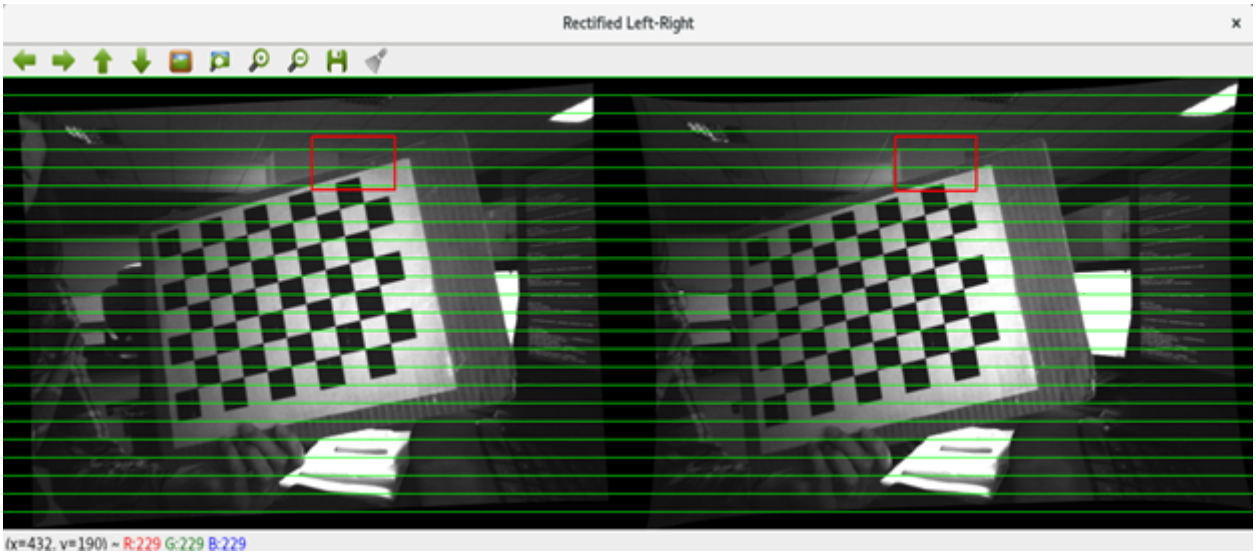


➢ **Example: bad rectification output**

```
./Rectification -c0 cam0_ov9282.txt -c1 cam1_ov9282.txt -nx 9 -ny 6 -
ss 2.5

--> Stereo system rectification...
Compute 2D pattern corners from images... Order test OKAY... Done!
Compute 3D pattern points...  Done!
Running calibration by Bouguet's method
      Compute camera matrix... Done!
      Compute stereo calibration... done with RMS error=2.42822
      Check calibration quality... average epipolar err = 1.53644
      Compute stereo rectification... Done!
      Save intrinsics... Done!
```

```
        Save extrinsics... Done!
        <-- Done!
Save dense rectification maps... Done!
Save rectification maps as meshes... Done!
```

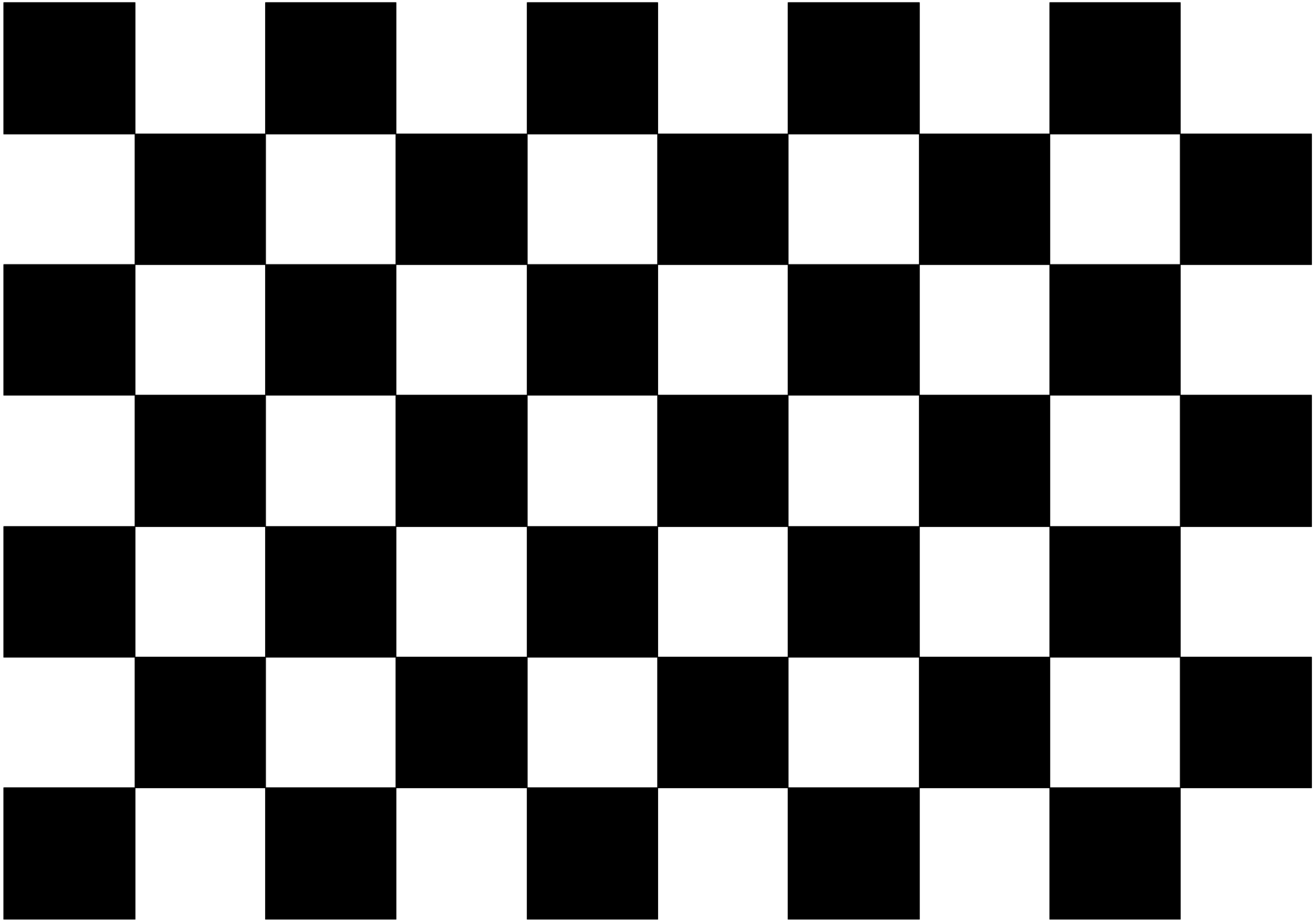


5.2 Using the New Calibration

The `mesh.h` created during this calibration procedure can now be used in the Myriad 2 demo package stereoEvalM2_QVGA_MDKxx.yy / stereoEvalM2_VGA_MDKxx.yy, at the following location:

`projects/DataCollectionSystem/apps/DataCollectionSGBM/myriadDemo/leon`

APPENDIX: CALIBRATION PATTERN



This is a 9x6
OpenCV chessboard
http://sourceforge.net/projects/opencvlibrary/