NXP Semiconductors
User's Guide

Document Number: SVAUG
Rev. 1, 07/2018

Surround View Application

1. Introduction

This document describes how to properly set up the 360-degree Surround View application on the i.MX 8QM device. The Surround View 3D (SV3D) project is divided into two separate parts: the calibration section and the rendering application. The whole calibration section is divided into these separate parts:

- Lens camera calibration (including the image capturing and the omnidirectional camera calibration), which is the intrinsic camera calibration.
- System calibration, which executes the whole preprocessing calculation and generates the files for texture mapping (rendering). The calibration can be done only once, because the camera system is fixed and the cameras do not move relatively to each other. Otherwise, the calibration process must be repeated.

The proper order of these calibrations is also important. The intrinsic (lens) calibration must be performed first.

The rendering application maps the camera's frames on a prepared 3D mash and blends the frames.

For details about the application's software theory, see the *Surround View Application Reference Manual* [1].

Contents

2. Hardware setup

The Surround View development platform consists of the parts listed in Table 1 (see also Figure 1). It is assumed that you can built the demo kit yourself using all released hardware and software sources.

Table 1. Surround view kit Bill of Material (BOM)

Item	Pcs.	Description	P/N
1	1	i.MX 8QM-MEK	MCIMX8QM-CPU
2	1	MAX9286 deserializer with miniSAS cable	MX8XMIPI4CAM2
3	4	Camera with coaxial cable	MCIMXCAMERA1MP
4	1	i.MX 8QM-MEK base board (optional only)	MCIM8-8X-BB
5	1	Display converter with miniSAS cable	IMX-DLVDS-HDMI
6	1	Mini-tripod	CL-TPPRE150
7	4	Camera enclosure	3D-printed
8	1	Camera stand	3D-printed
9	1	SDHC 32 (16) GB, UHS-I Class 10	SDSDUNC-032G-GN6IN
10	1	System (extrinsic camera) calibration pattern	530x530 mm
11	1	Lens (intrinsic camera) calibration pattern	297x297 mm

The below components are not a part of the development platform, but they are needed to run the whole camera system. It is assumed that you have all the remaining components and tools (see Table 2).

Table 2. Surround view additional components

Item	Description
12	USB keyboard
13	USB mouse
14	USB-C (male) to USB 3.0 (female) adapter cable + USB hub
15	HDMI monitor with HDMI cable
16	64-bit Windows® OS PC with Matlab or Matlab Runtime installed
17	Ethernet cable

If you have all hardware components delivered, perform these steps before running the applications (calibration, rendering):

- Connect all boards as shown in Figure 1. Connect the deserializer to the MIPI-CSI0 connector and the display converter to the LVDS0-CH0 connector on the CPU board. Interconnect these boards using miniSAS cables.
- Print the camera stand and the camera enclosure using a 3D printer. Use the sources available on the NXP webpage [3], located in the */Tools/CamStand* folder.
- Place the camera module to the bigger camera enclosure part with clips, plug in the smaller enclosure part, and click both camera covers (left + right). Finally, plug each camera module to the camera stand. See Figure 2.
- Screw the tripod to the camera stand using the built-in ¼″ tripod screw (see Figure 3).
- Interconnect all cameras with the deserializer using the coaxial cables. Keep the right camera order (see Figure 1). Keep the clockwise camera connection on the camera stand. That means

that the next camera is housed on the right side to the previous camera (see Figure 3), not across them.

- Connect both the USB keyboard and the USB mouse to the USB-C port on the CPU board (use an USB hub for this step).
- Plug the SD card into the slot on the CPU board.
- Remove the plastic lens covers from the four-camera module.
- Connect an external HDMI monitor to the display converter.
- Connect the AC/DC adapter to the CPU board and switch on the board using the ON/OFF push-button.

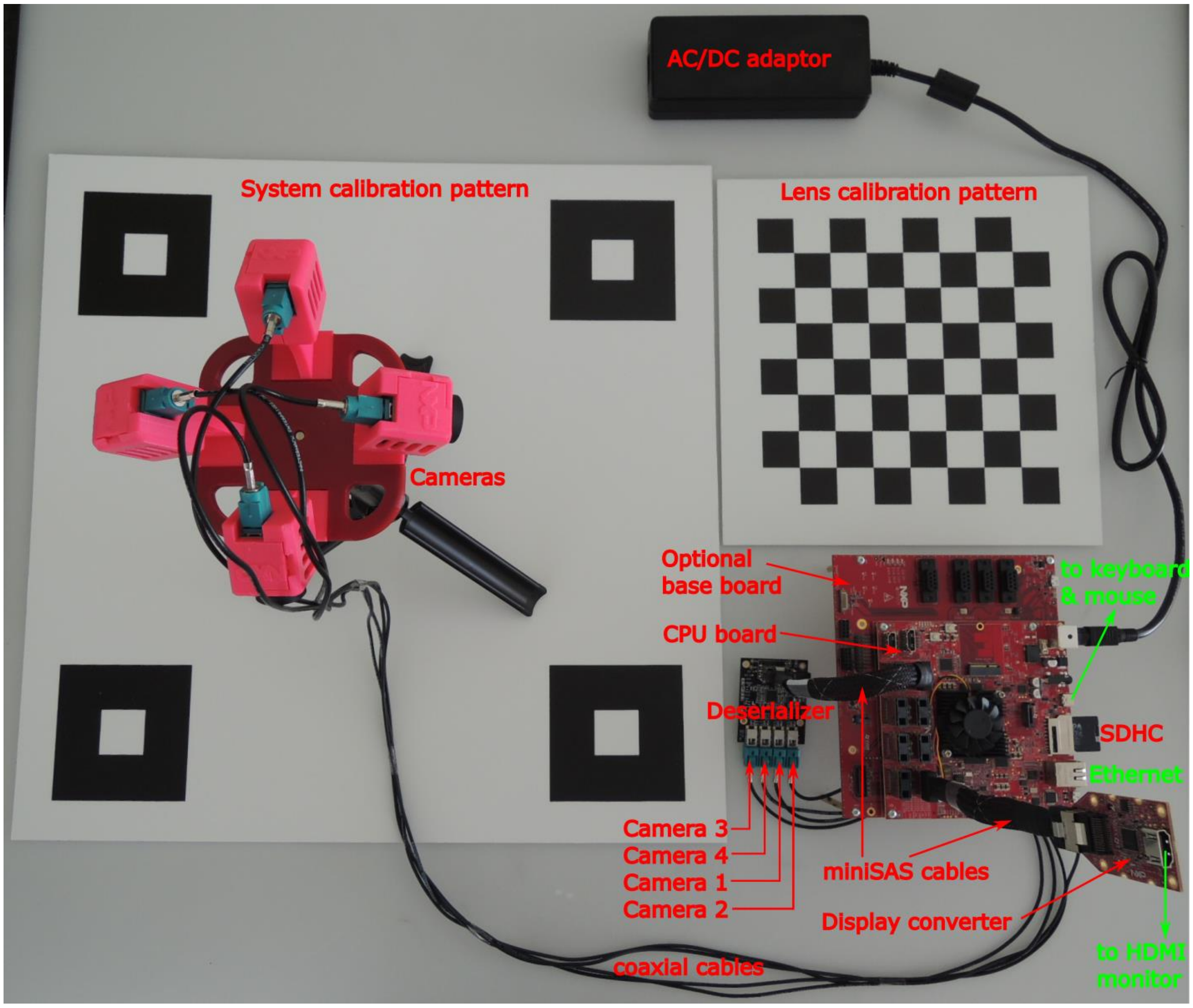


Figure 1. Surround View development platform based on i.MX 8QM device

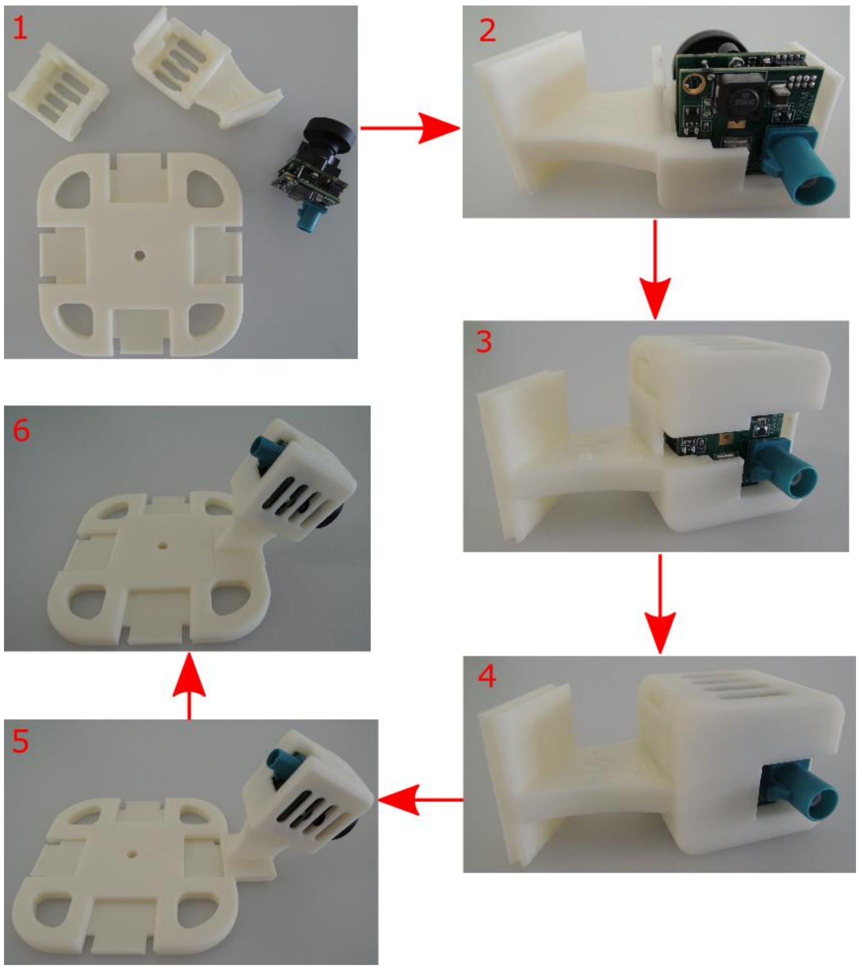
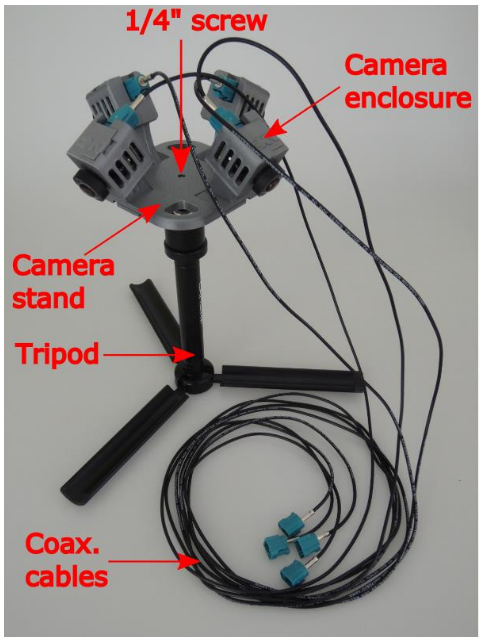


Figure 2. Camera stand arrangement



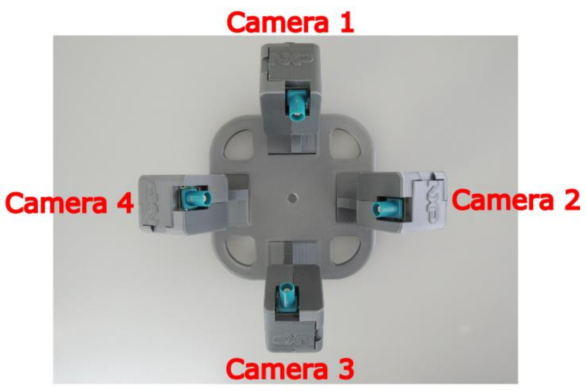


Figure 3. Camera system detail

3. Application tools overview

The detailed directory structure of the whole Surround View application is shown in Figure 4. Perform these steps for building the application from sources[1] [3]:

- Download the latest Yocto Linux OS image from the official web page[2].
- Save this image on the SD card using some raw disk image writing tool (for example, Win32DiskImager)
- Download the latest Surround View application source code from the repository[3], unzip this file, save it on the system SD card, and build the application using the prepared Yocto image.

The instructions on how to build the application are included in the *README.md* file in the root directory. It is also described in the *Surround View Application Reference Manual* [1].

NOTE

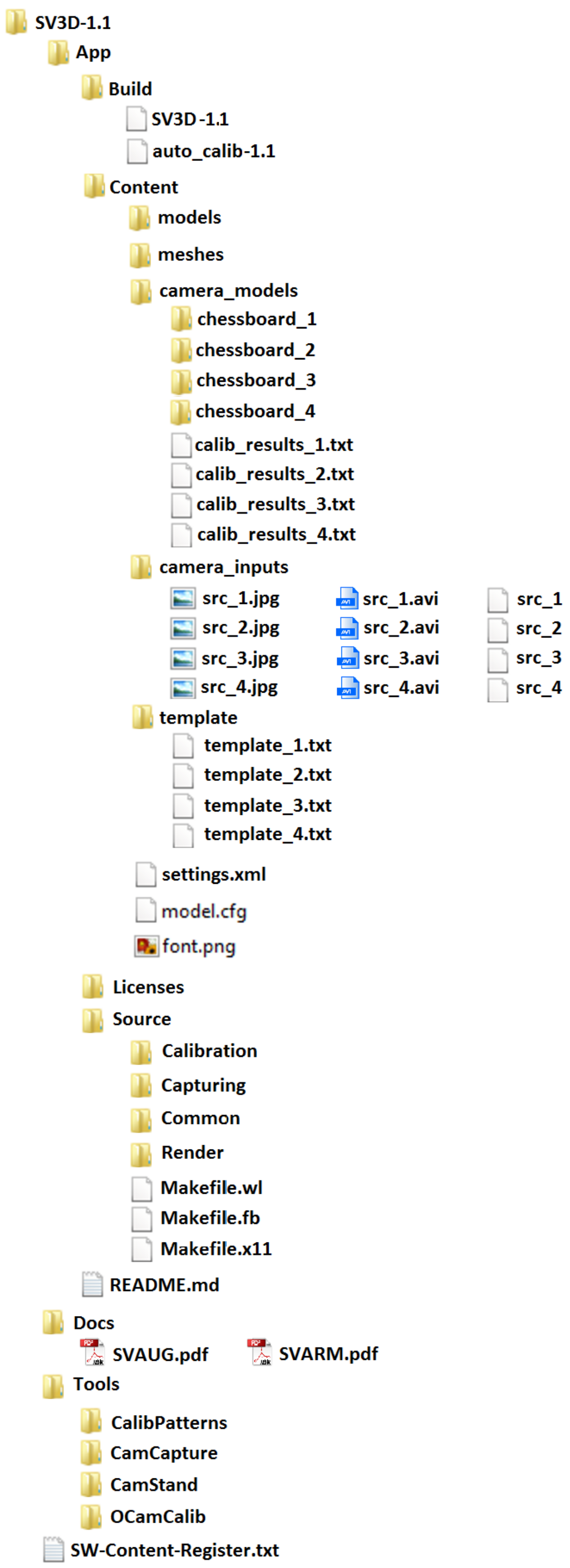
The official Yocto Linux OS image contains both the root file system and kernel.

The root application directory is */home/root/SV3D-1.1*. This is the description of the most important subdirectories of the archive:

- *App*—embedded code for the i.MX platform.
  - *Build*—contains the binary files of the main applications (*SV3D, auto_calib*).
  - *Content*—contains the calibration and settings files, car model, static images, and video files.
  - *Source*—contains the source files for the i.MX embedded code (*SV3D, auto_calib, capturing*).
- *Doc*—contains the User's Guide and the Reference Manual in the PDF format.
- *Tools* – contains the helper tools.
  - *CalibPatterns*—contains PDF versions of both calibration patterns (lens, system).
  - *CamCapture*—contains the camera capture binary file (*capturing*).
  - *CamStand*—contains the model of the camera enclosure and the stand for 3D printing.
  - *OCamCalib*—contains the camera lens calibration tool (*ocam_calib.exe*).

NOTE

Only the lens camera calibration tool (OCamCalib) runs on the Windows OS platform. Other applications (image capturing, system calibration, rendering) run on the Yocto Linux OS platform.



```
SV3D-1.1
    App
        Build
            SV3D-1.1
            auto_calib-1.1
        Content
            models
            meshes
            camera_models
                chessboard_1
                chessboard_2
                chessboard_3
                chessboard_4
                calib_results_1.txt
                calib_results_2.txt
                calib_results_3.txt
                calib_results_4.txt
            camera_inputs
                src_1.jpg     src_1.avi     src_1
                src_2.jpg     src_2.avi     src_2
                src_3.jpg     src_3.avi     src_3
                src_4.jpg     src_4.avi     src_4
            template
                template_1.txt
                template_2.txt
                template_3.txt
                template_4.txt
            settings.xml
            model.cfg
            font.png
        Licenses
        Source
            Calibration
            Capturing
            Common
            Render
            Makefile.wl
            Makefile.fb
            Makefile.x11
        README.md
    Docs
        SVAUG.pdf     SVARM.pdf
    Tools
        CalibPatterns
        CamCapture
        CamStand
        OCamCalib
    SW-Content-Register.txt
```

Figure 4. Surround View application directory structure

4. System settings

Because the keyboard and mouse are used on the target board, their proper setup (mapping) must be performed before running all the Yocto Linux OS-based applications (capturing, auto_calib, SV3D). These settings are ignored for the x11 graphical backend, but they are mandatory for the framebuffer or the XWayland graphical backends. In these cases, it is necessary to define proper keyboard, mouse, and display devices (events) on the target board.

These settings are recorded in the separate *settings.xml* file, located in the */App/Content/* folder. This file contains also other system settings. The default camera configuration and the keyboard and mouse events are in the *<fb>* section. These events are read from the */dev/input/by-path/* directory on the SD card and they are unique for the particular boards.

- <*keyboard*>—the absolute path of a keyboard device.
- <*mouse*>—the absolute path of a mouse device.
- <*display*>—the absolute path of a display device.

To change other parameters (no need for the default camera configuration), see the *Surround View Application Reference Manual* [1] for a more detailed description of each section in the *settings.xml* file.

5. Lens calibration

5.1. Image capturing

For a proper lens (intrinsic camera) calibration, the image capturing for each separate camera must be done first. Use a chessboard calibration pattern for this (see Figure 5). The original PDF pattern file is located at *Tools/CalibPatterns/patternLensCameraCalibration.pdf*.

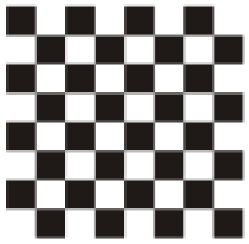


Figure 5. Lens calibration pattern

You may also print your own pattern on an A4 or A3 paper and place it on a piece of cardboard. There must be a thick white border around the pattern. This white border is needed by the “Automatic Checkerboard Extraction” tool to facilitate the corner extraction (see Section 5.2, “Camera intrinsic parameters estimation tool”).

The application sources to build this application are available on the NXP webpage [3]. Use the camera image capturing application (binary file) with a right parameter (1-4) for these purposes:

- */Tools/CamCapture/capturing 1*—capture the first camera frames.
- */Tools/CamCapture/capturing 2*—capture the second camera frames.
- */Tools/CamCapture/capturing 3*—capture the third camera frames.

- */Tools/CamCapture/capturing 4*—capture the fourth camera frames.

While the current application is running, use the “p” key to save the current frame and the “Esc” key to quit. The application saves all frames to the current directory in the *frameX_Y.jpg* format, where “X” is the camera number (1…4), and “Y” is the frame order number (starts from 0, incremented by the application).

To obtain good calibration results, these requirements must be met:

- Capture a minimum of 10 frames per each camera for different angles of view.
- Place the checkerboard as close to the cameras as possible (see Figure 6). This improves the calibration and helps the Automatic Checkerboard Extraction tool to find all corners. Make sure that every corner of the checkerboard is visible in each image. For the Automatic Checkerboard Extraction tool, it is important that a white border is visible around the pattern.
- Take pictures of the checkerboard to cover all the visible area on the camera. By doing this, you enable the calibration to compensate for possible camera misalignments. It also helps to detect the center of the omnidirectional image.

Perform these additional steps:

- Copy all the *frameX_Y.jpg* files to the respective *App/Content/camera_models/chessboard_X* directory (X is 1…4).
- Set the right <*chessboard_num*> parameter in the *settings.xml* file for each camera (1 to 4). This number must be the same as the number of JPEG pictures in the respective */cheesboard_X* subdirectory.

An example of proper sample images captured with a fisheye camera with a 180-degree field of view is shown in Figure 6.

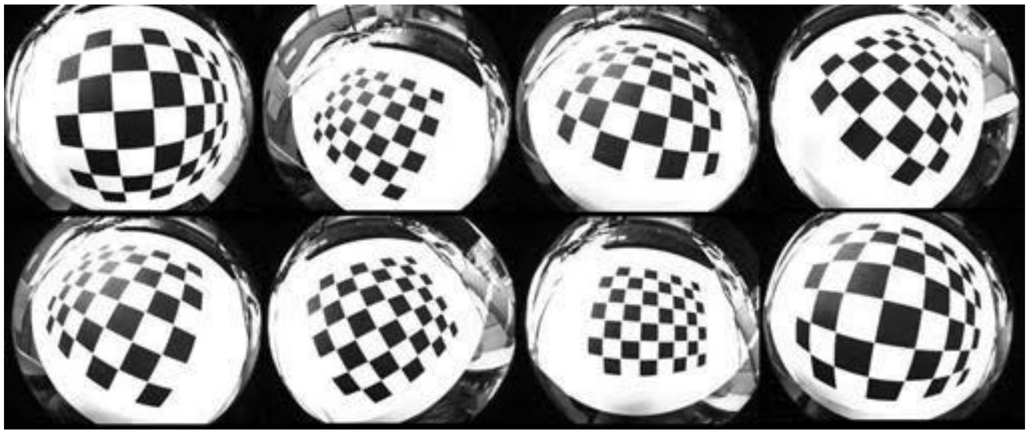


Figure 6. Image capturing example

5.2. Camera intrinsic parameters estimation tool

Each camera (lens) must be calibrated to find the camera’s intrinsic parameters. The OCamCalib Toolbox for Matlab [4] is used to estimate the lens distortion, affine transformation, and image center. Follow the instructions described in the tutorial on the official webpage [4] to run the application using Matlab. The second way to run the application is using a prebuilt binary file (*ocam_calib.exe*) with Matlab Runtime. This binary file is located in the */Tools/OCamCalib* folder (see Figure 4). To run the binary, the *autoCornerFinder* folder from the original location[4] or from the public repository [6] is needed. Copy this folder to the Windows root directory, where *ocam_calib.exe* is copied.

Figure 7 shows the OCamCalib GUI:

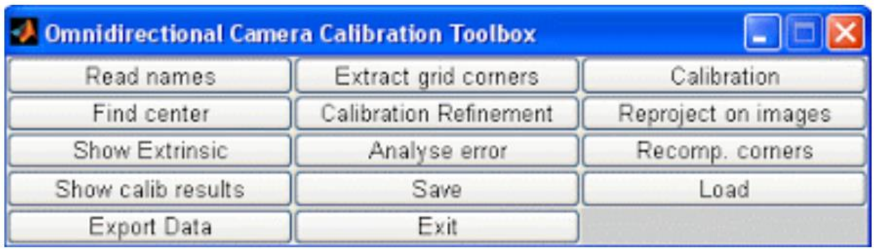


Figure 7. OcamCalib GUI

NOTE

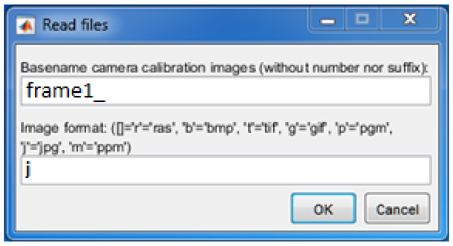
The OCamCalib application runs only on the Windows OS platforms.

5.2.1. Matlab Runtime installer

If using the OCamCalib Matlab toolbox [4], skip directly to Section 5.2.2, „Load images“, because installing the Matlab Runtime is not needed in this case. To run the standalone binary application, install the Matlab Runtime on your Windows PC. Firstly, download the Matlab Runtime from the official webpage [5]. Follow the instructions from the installation tool. You do not need a Matlab license to install the Runtime version. Secondly, run the ocam_calib.exe application on Windows PC platform.

5.2.2. Load images

Copy the captured images (*frameX_Y.jpg*, see Section 5.1, "Image capturing") to the folder from which the *ocam_calib.exe* or OCamCalib Matlab toolbox is run. After running the application (Matlab toolbox or binary file), click the "Read names" button. A dialog window asking for the base names and format of your images appears:



Basename camera calibration images (without number nor suffix):

frame1_

Image format: ([]='r'='ras', 'b'='bmp', 't'='tif', 'g'='gif', 'p'='pgm', 'j'='jpg', 'm'='ppm')

j

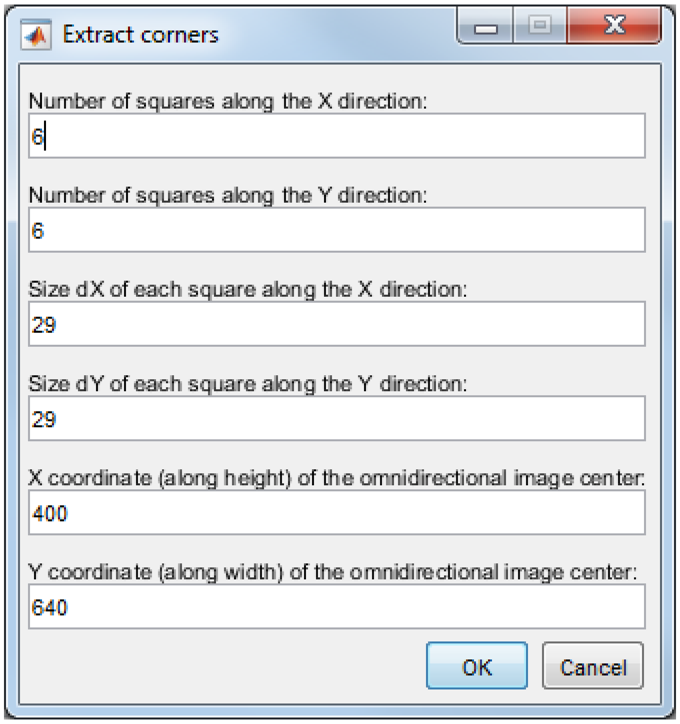
Figure 8. OCamCalib "Read files" menu

If your images are named *frame1_0.jpg*, *frame1_1.jpg* ... *frame1_9.jpg*, type "frame1_" into the first prompt and "j" into the second prompt. Repeat this step for all cameras, that is for "frame2_", frame3_", and "frame4_" image names, after performing the next steps (see sections 5.2.3 to 5.2.7).

5.2.3. Extraction of grid corners

The extraction of grid corners is the most important phase of the calibration, as the calibration results depend on the position of the checkerboard corners in each image.

Click the “Extract grid corners” button. A dialog box with the default values appears (see Figure 9).



Extract corners

Number of squares along the X direction: 6

Number of squares along the Y direction: 6

Size dX of each square along the X direction: 29

Size dY of each square along the Y direction: 29

X coordinate (along height) of the omnidirectional image center: 400

Y coordinate (along width) of the omnidirectional image center: 640

Figure 9. OCamCalib “Extract corners” menu

First, type the number of inner squares present along the X and Y directions. This number is lower by 2 than the actual number of the square along each direction in the checkerboard, because only the inner squares are considered. In case of the default (included) calibration pattern, type “6” for the number of squares along both the X and Y directions (see Figure 9).

Second, fill the actual size of the squares. In this case (for instance), it is 29 mm in both the X and Y directions.

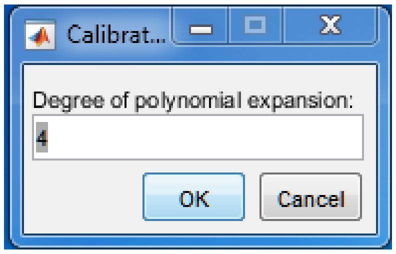
The checker size is only used to recover the absolute positions of the checkerboards. For the intrinsic parameters, this is not needed and you may leave this field as it is.

In the last two prompts, the toolbox asks for the position (rows, columns) of the center of the omnidirectional image. Because the OcamCalib toolbox can automatically determine the location of the center, you can leave the default values (height/2 and width/2). This is not important, because you are going to use the “Find center” button later to find the correct position of the center. However, you may optionally specify the location of the center and refine it later using the “Find center” button.

After submitting the parameters, the corners are automatically extracted and the results of the extraction are displayed.

5.2.4. Calibration

You are now ready to calibrate the omnidirectional camera. To do this, click the “Calibration” button. An input dialog asking for the degree of the polynomial expansion appears. This parameter enables you to set the maximum order of the polynomial which approximates the function that back-projects every pixel point into a 3D space. Several experiments with different camera models show that a polynomial order of 4 provides the best results.



Degree of polynomial expansion:
4

Figure 10. OCamCalib “Calibration” menu

At the end of the calibration, the toolbox displays a graph which shows the plot of function “F”, and the plot of angle “THETA” of the corresponding 3D vector with respect to the horizon.

5.2.5. Find Center tool

Use the Find Center tool always before using the Calibration Refinement. In fact, the automatic detection of the image center is done by iterative application of a linear estimation method, which is suboptimum. When you estimate the image center, then you may run the Calibration Refinement, which refines all calibration parameters and the position of the center using a non-linear method.

This routine tries to extract the image center automatically. If you did not set the correct values for the center of the omnidirectional image during the grid corner extraction, then you may use the automatic detection of the center. To do this, click the “Find center” button and the OCamCalib Toolbox starts an iterative method to compute the image center, which minimizes the reprojection error of all grid points. The automatic center detection may take some time (around 10 minutes).

5.2.6. Calibration Refinement

By clicking the “Calibration Refinement” button, the Toolbox starts the non-linear refinement of the calibration parameters using the Levenberg-Marquadt algorithm. The optimization is performed by attempting to minimize the sum of squared reprojection errors.

5.2.7. Export Data

Click the “Export Data” button to export the calibration results to the *calib_results.txt* file. It is saved to the folder in which the application is placed. Append the camera number (1, 2, 3, or 4) as a postfix to the file name (for example, *calib_results_4.txt*) and copy the file from the current Windows OS folder to the Linux OS */App/Content/camera_models* folder.

NOTE

Use a raw image writing tool (for example, Win32DiskImager) to copy these files from Windows OS to Linux OS (i.MX board) via an Ethernet cable.

6. System calibration

The system calibration is performed using the Automatic Calibration application. It is a one-shot, stand-alone application which executes the whole preprocessing calculation and generates the "mask" and "array" files for texture mapping located in the */App/Build* folder finally.

6.1. Arrangement

To perform the extrinsic calibration of the 4-camera system, a square poster of a known size is used (see Figure 11).

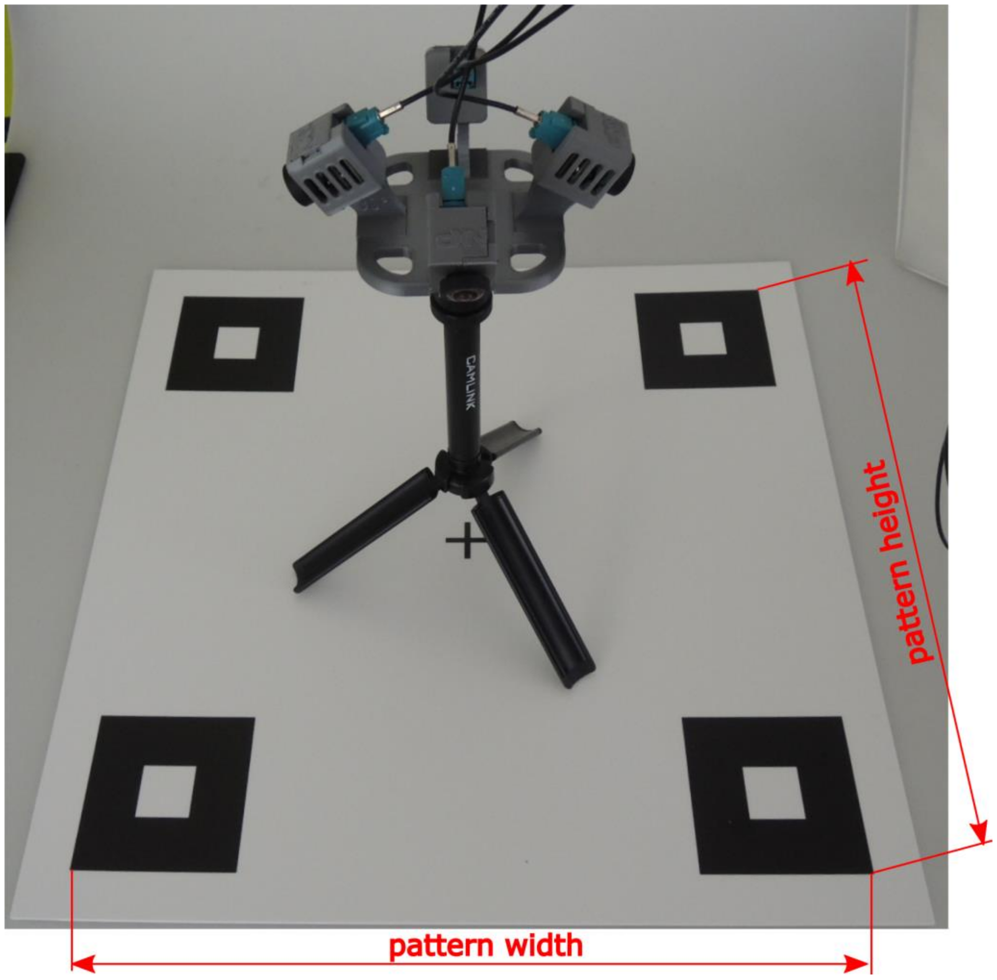


Figure 11. Extrinsic camera calibration setup

- Perform the whole hardware setup described in Section 2, "Hardware setup" and the system settings described in Section 4, "System settings".

- Perform the intrinsic (lens) camera calibration described in Section 5, “Lens calibration”.
- Place the tripod with the cameras to the center of the big calibration pattern (see Figure 11).
- Use the *auto_calib_1.1* binary file located in the */App/Build* SD card folder. You may also build the automatic calibration application from sources. See the applicable *Surround View Reference Manual* [1] section.

There are five calibration steps in the Automatic Calibration application:

1. Fisheye camera view
2. Fisheye distortion removal
3. Contours’ search
4. Meshes’ preparation
5. Result view calculation

The Automatic Calibration application is controlled using these keyboard keys on the target board:

- “Right arrow” key—go to the next calibration step.
- “Left arrow” key—go to the previous calibration step.
- “F5” key—update the current step.
- “Esc” key—terminate the application.

NOTE

If using the framebuffer or Wayland graphical backends, set the right keyboard/mouse device names in the *settings.xml* file. If you do not update the names with actual values, these devices are not going to work.

6.2. Calibration process

Put the camera system into the center of the calibration pattern. There should not be any other objects on the calibration pattern except for the camera system. It is also recommended to manually hold the coaxial cables above the camera stand during the search contours calibration step (see Section 6.2.3, “Contours”). Go to the */App/Build* folder and run the *./auto_calib_1.1* file.

6.2.1. Fisheye camera view

The first application view shows the fisheye cameras’ frames. You may see the original cameras’ outputs on the screen. The camera view order is shown in Figure 12.

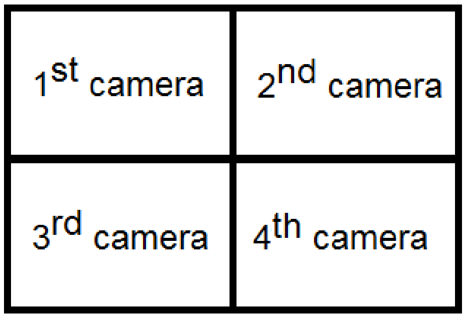
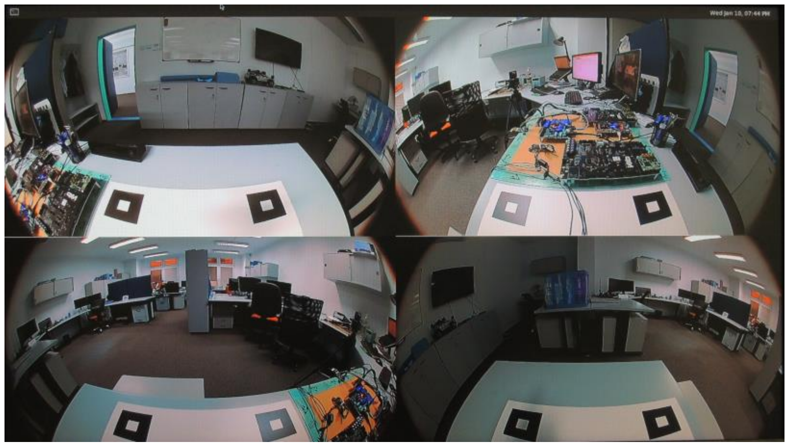
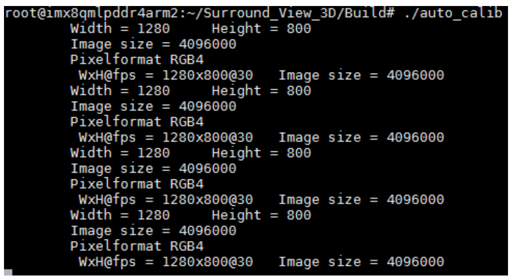


Figure 12. Camera view order

Make sure that your camera order is right and you see all templates on the cameras' frames. The actual state of the screen and terminal screenshots is shown in Figure 13.





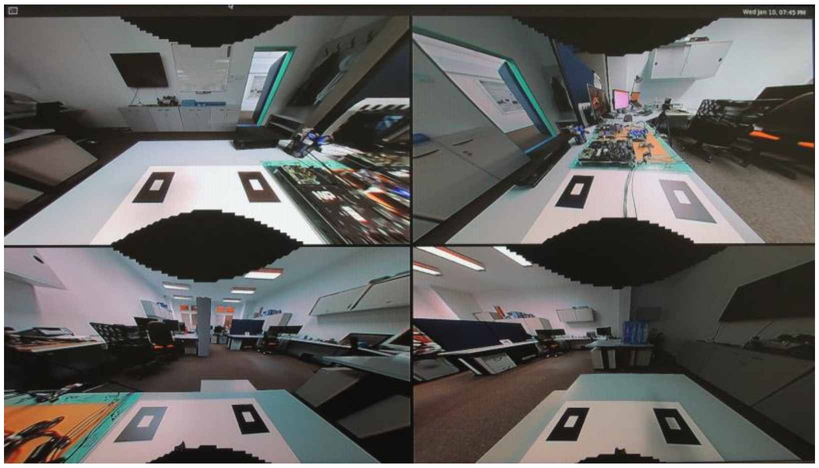
```
root@imx8qmlpddr4arm2:~/Surround_View_3D/Build# ./auto_calib
        Width = 1280     Height = 800
        Image size = 4096000
        Pixelformat RGB4
         WxH@fps = 1280x800@30   Image size = 4096000
        Width = 1280     Height = 800
        Image size = 4096000
        Pixelformat RGB4
         WxH@fps = 1280x800@30   Image size = 4096000
        Width = 1280     Height = 800
        Image size = 4096000
        Pixelformat RGB4
         WxH@fps = 1280x800@30   Image size = 4096000
        Width = 1280     Height = 800
        Image size = 4096000
        Pixelformat RGB4
         WxH@fps = 1280x800@30   Image size = 4096000
```

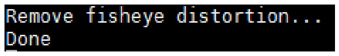
Figure 13. Fisheye camera views and terminal screenshot

Press the "Right Arrow" key on the keyboard to continue or the "Esc" key to terminate the application.

6.2.2. Fisheye distortion removal

The second application view shows the camera frames after the fisheye distortion is removed. Make sure that you see all templates on the camera frames and the lighting conditions are good (reflections may affect the templates' view). The calibration result is better with a natural light. If you saw all template corners on the previous view and now cannot see all corners, or the templates look too small after removing the fisheye distortion, you can change the scale factor in the */App/Content/settings.xml* file. It is not needed to terminate the "auto_calib" application. Change the <*sf*> values in the */App/Content/settings.xml* file, save the changes, and press the "F5" key to update the camera parameters. The actual state of the screen and terminal screenshot is shown in Figure 14.





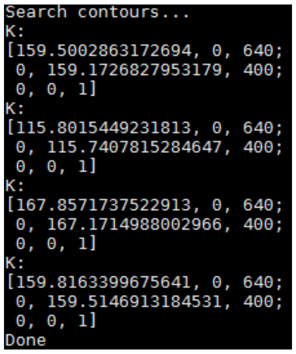
```
Remove fisheye distortion...
Done
```

Figure 14. Non-fisheye camera views and terminal screenshot

To return to the previous step, press the “left arrow” key. To terminate the application, press the “Esc” key. To continue to the next step, press the “right arrow” key.

6.2.3. Contours’ search

The third step applies the contours searching. It is the slowest step in the automatic calibration process. Be patient and wait until the text in Figure 15 appears in the terminal.



```
Search contours...
K:
[159.5002863172694, 0, 640;
 0, 159.1726827953179, 400;
 0, 0, 1]
K:
[115.8015449231813, 0, 640;
 0, 115.7407815284647, 400;
 0, 0, 1]
K:
[167.8571737522913, 0, 640;
 0, 167.1714988002966, 400;
 0, 0, 1]
K:
[159.8163399675641, 0, 640;
 0, 159.5146913184531, 400;
 0, 0, 1]
Done
```

Figure 15. Proper search contours terminal screenshot

The found contours are marked by the yellow color (see Figure 16).

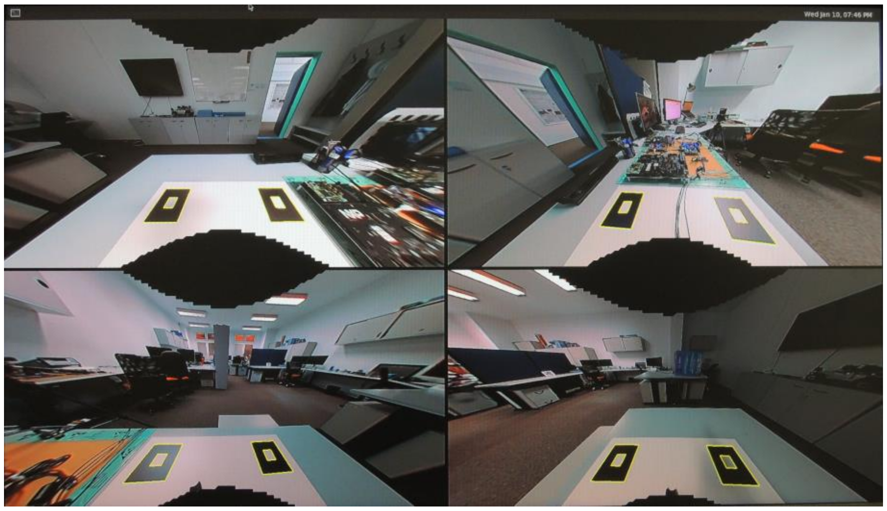
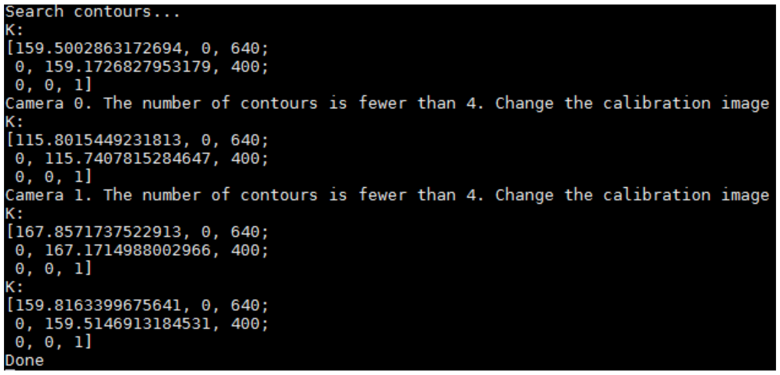


Figure 16. Found contours screenshot

If you see the text “*The number of contours is fewer than 4*” in the terminal window, it means that the contours were not found (see Figure 17).



```
Search contours...
K:
[159.5002863172694, 0, 640;
 0, 159.1726827953179, 400;
 0, 0, 1]
Camera 0. The number of contours is fewer than 4. Change the calibration image
K:
[115.8015449231813, 0, 640;
 0, 115.7407815284647, 400;
 0, 0, 1]
Camera 1. The number of contours is fewer than 4. Change the calibration image
K:
[167.8571737522913, 0, 640;
 0, 167.1714988002966, 400;
 0, 0, 1]
K:
[159.8163399675641, 0, 640;
 0, 159.5146913184531, 400;
 0, 0, 1]
Done
```

Figure 17. Bad search contours terminal screenshot

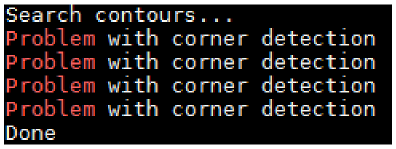
In this case, check the contours on the screen and identify the problem. For example, it can be one of these problems:

- The small contours were not found. Solution: change *<contour_min_size>* value in the *settings.xml* file.
- The contours were not found because of the lighting conditions. Solution: change the lighting conditions.
- The application did not find the contours that are located too high on the frame. Solution: change the *<roi>* value in the *settings.xml* file.
- Some additional contours were found. Solution: change the place for the calibration or move these additional objects away from the camera view. You can also decrease the *<roi>* value in the *settings.xml* file if these additional contours are located at the top part of the calibrating frame.

When you change the *settings.xml* file and you want to run the contours searching process again, it is not necessary to terminate the “auto_calib” application. Press the “F5” key to update the settings. If you change the scale factor at this step of the calibration, it is not going to be updated. The scale factor is updated only for the second step (fisheye distortion removal). If you want to update the scale factor in the third step, press the “left arrow” key and return to the second step.

CAUTION

If you use OpenCV3.2 with OpenVX and you see the terminal message shown in Figure 18, disable the OpenVX version using the “export NO_OPENVX=1” command in the terminal.



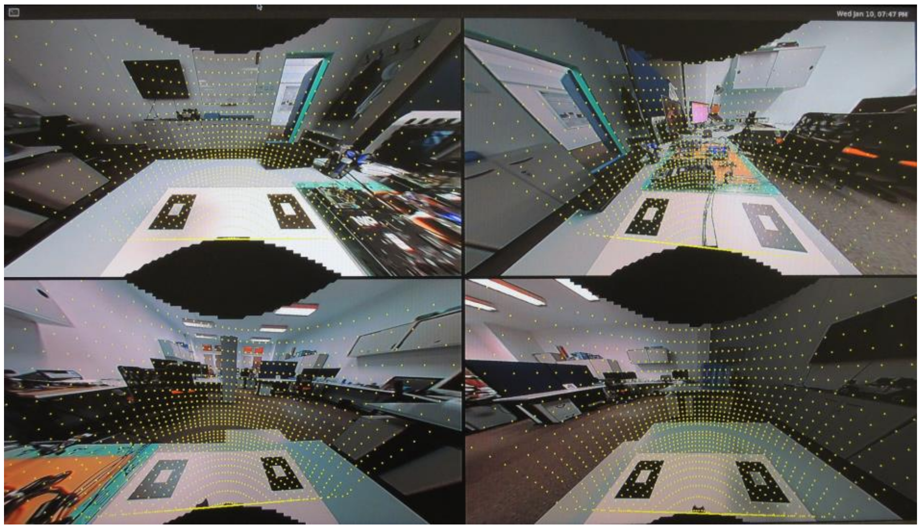
```
Search contours...
Problem with corner detection
Problem with corner detection
Problem with corner detection
Problem with corner detection
Done
```

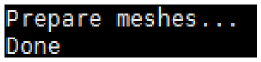
Figure 18. A specific application run-time error

Press the “right arrow” key to continue or the “Esc” key to terminate the application.

6.2.4. Meshes’ preparation

The fourth calibration step shows the mesh for mapping. The meshes are drawn on the screen as yellow dots. If you want to change the mesh density or height, change the <grid> parameters in the *settings.xml* file according to the *Surround View Application Reference Manual* [1] and press the “F5” key to recalculate the meshes. Figure 19 shows the actual state of the screen and the terminal screenshot.





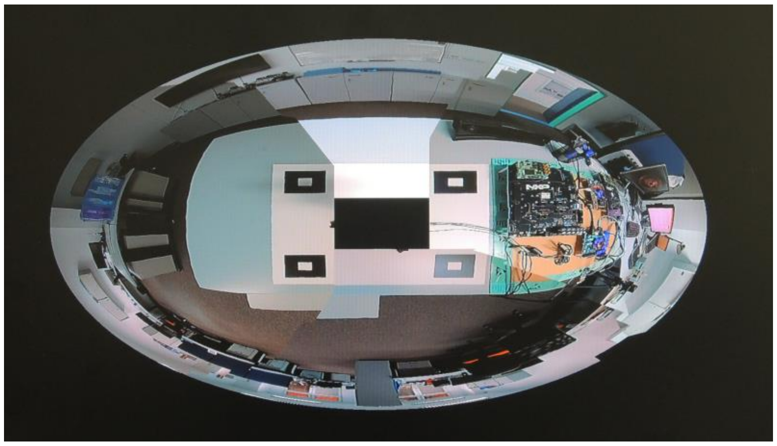
```
Prepare meshes...
Done
```

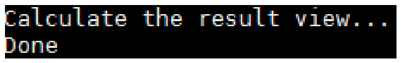
Figure 19. Prepare meshes views and terminal screenshot

Press the “right arrow” key to continue or the “Esc” key to terminate the application. If you want to return to the previous step, press the “left arrow” key.

6.2.5. Result view calculation

The result view does not look exactly as a true surround view output. To perform the calculation faster, it looks like an ellipse instead of a circle. There is no mouse cursor in this view. That means you cannot rotate it, but you can evaluate the quality of stitching. Figure 20 shows the actual state of the screen and a terminal screenshot. Currently, the application generates the "mask" and "array" files in the */App/Build* folder. All these files are finally used by the rendering application.





```
Calculate the result view...
Done
```

Figure 20. Result surround view and terminal screenshot

7. Real-Time Rendering application

The Real-Time Rendering application (called SV3D-1.1) is the main part of the Surround View project. It renders the camera frames (default mode) on a prepared 3D mesh and blends them. Alternatively, it can also render static images or video files (demo mode). The Real-Time Rendering application stitches four input images (static images or camera frames) into a single output and displays it using the GPU managed by the OpenGL ES 3.0 standard. Finally, a simple car model is stitched to the center of the scene. The application performs the 3D rendering using all the "array" and "mask" files generated by the Automatic Calibration application from the current camera frames (default mode), static images, or video files.

It is assumed that you built the application from sources [1][3]. Rendering application binary file is located in the */App/Build* directory after building. To run the Real-Time Rendering application in the Linux OS terminal, type *./SV3D-1.1*. Figure 21 shows the output on the screen.

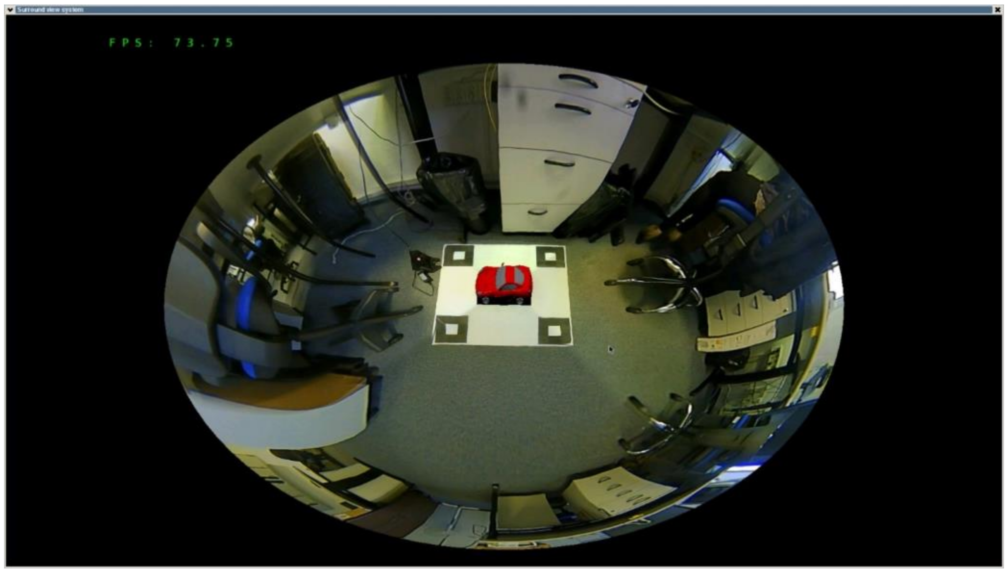


Figure 21. Composite 3D surround view

The current frame rate value is in the top left corner of the screen. The default application view is the top-down view. To change the angle view (image rotation), move the mouse while holding its left button. To zoom, use the mouse scroll wheel. To close the application, press the “Esc” key on the keyboard.

NOTE

If your keyboard or mouse does not work on the target board, perform the proper system settings (see Section 4, “System settings”).

8. References

1. *Surround View Application Reference Manual* (document SVARM)
2. *i.MX6 / i.MX7 / i.MX 8 Series Software and Development Tool Resources*, available at www.nxp.com
3. “Surround View Application”, Reference Design Project page available at www.nxp.com
4. Davide Scaramuzza, *OCamCalib: Omnidirectional Camera Calibration Toolbox for Matlab*, available at https://sites.google.com/site/scarabotix/ocamcalib-toolbox
5. Matlab Runtime R2017a, available at www.mathworks.com
6. OCamCalib, "Omnidirectional Camera Calibration Toolbox for Matlab Runtime” available at https://source.codeaurora.org/external/imx.

9. Revision history

Table summarizes the changes done to this document since the initial release.

Table 3. Revision history

Revision number	Date	Substantive changes
0	03/2018	Initial release.
1	07/2018	Added Matlab Runtime for OCamCalib. Changed web links in Table 1.

How to Reach Us:

Home Page:
www.nxp.com

Web Support:
www.nxp.com/support

Document Number: SVAUG
Rev. 1
07/2018