SVAUG

Surround View Application User Guide

Rev. 4 — 22 May 2025 User guide

Document information

Information	Content
Keywords	Surround View application, i.MX 95 EVK
Abstract	This document describes how to set up properly the 360-degree Surround View application on the i.MX 95 EVK device.

1 Introduction

This document describes how to set up properly the 360-degree Surround View application on the i.MX 95 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 relative to each other. Otherwise, the calibration process must be repeated.

The proper order of the 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* (document SVARM).

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 build 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 95 EVK	
2	1	MIPI-CSI De-serializer module.	MX95MBDESER01
3	4	Camera with coaxial cable	ox03c10
4	1	Display converter with miniSAS cable	IMX-DLVDS-HDMI
5	1	Mini-tripod	Tripod
6	4	Camera enclosure	3D-printed (see Figure 9)
7	1	Camera stand	3D-printed (see Figure 9)
8	1	SDHC 32 (16) GB, UHS-I Class 10	SDSDUNC-032G-GN6IN
9	1	System (extrinsic camera) calibration pattern	530x530 mm (see Figure 9)
10	1	Lens (intrinsic camera) calibration pattern	297x297 mm (see Figure 3)

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-CSI connector and the display converter to the LVDS0-CH0 connector on the CPU board. Interconnect these boards using miniSAS cables.
- In this demo, an enclosure printed by a 3D printer is used to fix the camera positions. If you do not have such enclosure, find a way to fix the position of cameras.
- Interconnect all cameras with the deserializer using the coaxial cables. Keep the right camera order.
- Connect both the USB keyboard and the USB mouse to the USB-C port on the CPU board.
- 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 pushbutton.

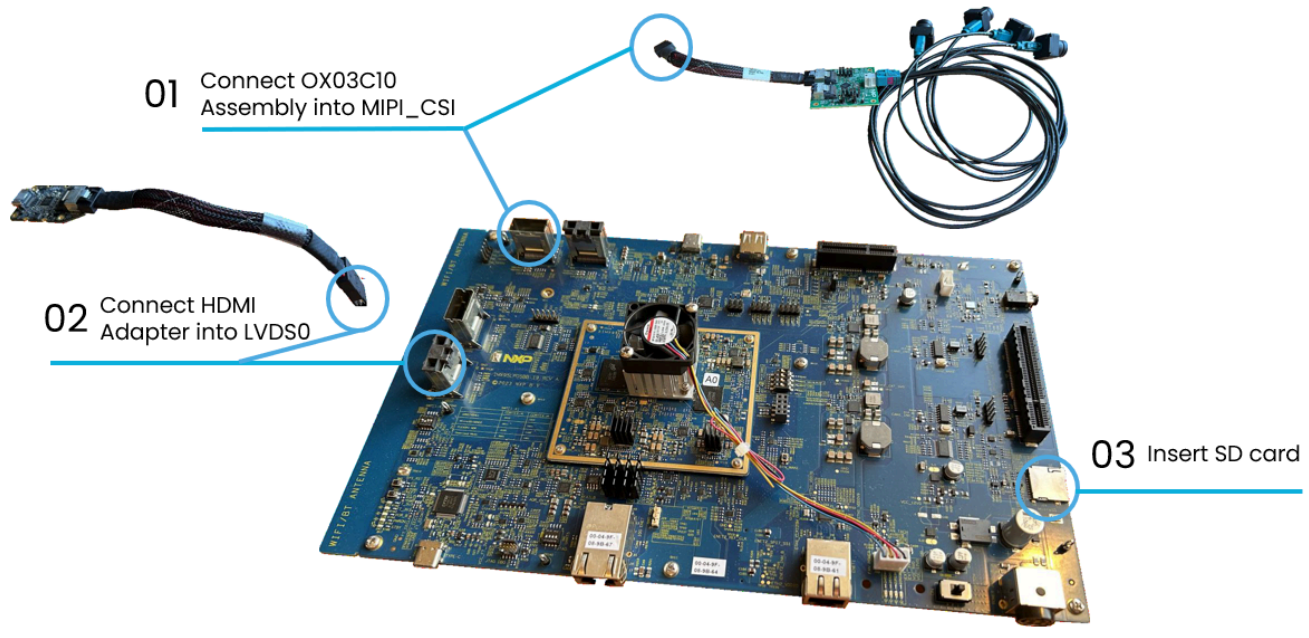

Figure 1. Surround View development platform based on i.MX 95 EVK device

3 Application tools overview

The detailed directory structure of the whole Surround View application is shown in Figure 2. Perform these steps for building the application from Surround View Application:

- Download the latest Yocto Linux OS image from the official web page
- 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 and build the application using the prepared Yocto image.

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

The root application directory is */root/SV3D-1.4*. It 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, 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*).

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.*

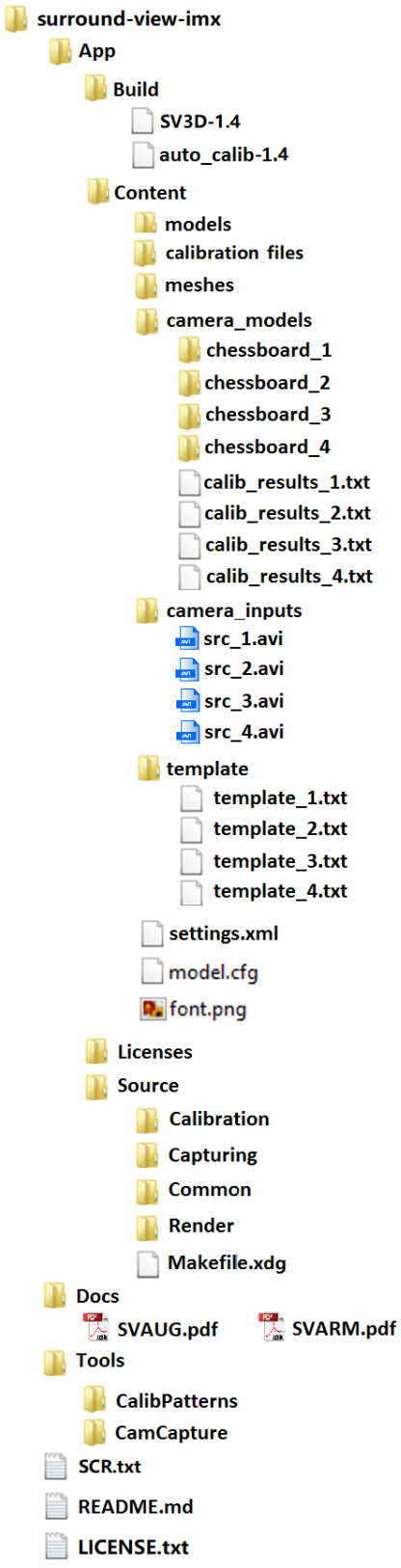

Figure 2. Surround View application directory structure

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`). It is necessary to define the 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* (document SVARM). for a more detailed description of each section in the `settings.xml` file.

5 Lens calibration

This section provides information on the 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 to do this (see Figure 3). The original PDF pattern file is at `Tools/CalibPatterns/patternLensCameraCalibration.pdf`.

Figure 3. 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 GitHub. Use the camera image capturing application (binary file) with the 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 camera for different angles of view.
- Place the checkerboard as close to the cameras as possible (see Figure 6). It 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, a white border must be visible around the pattern.
- Take pictures of the checkerboard to cover all the visible area on the camera. By doing it, 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 4.

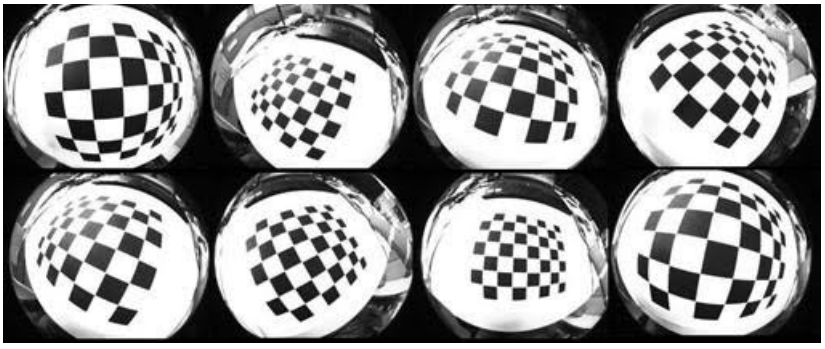

Figure 4. Image capturing example

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 is used to estimate the lens distortion, affine transformation, and image center. Follow the instructions described in the tutorial on the official webpage 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. To run the binary, the `autoCornerFinder` folder from the original location or the public repository is needed. Copy this folder to the Windows root directory, where `ocam_calib.exe` is copied.

Figure 5 shows OCamCalib GUI:

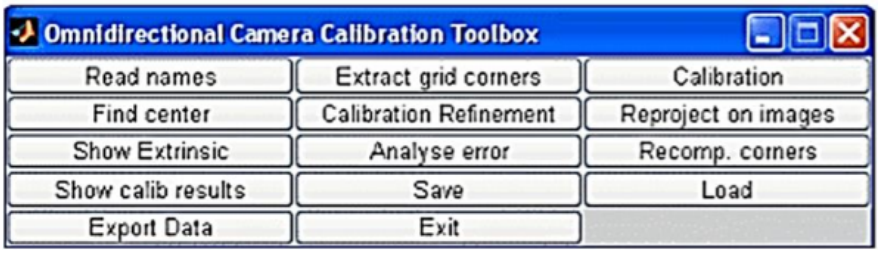

Figure 5. OcamCalib GUI

Note: *The OCamCalib application runs only on the Windows OS platforms.*

5.2.1 Matlab Runtime installer

If using the OCamCalib Matlab toolbox , 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. First, download the Matlab Runtime from the official webpage. Follow the instructions from the installation tool. You do not need a Matlab license to install the Runtime version. Second, 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:

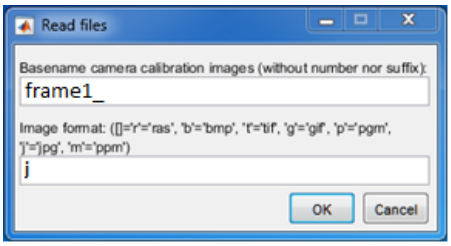

Figure 6. OCamCalib “Read files” menu

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 7).

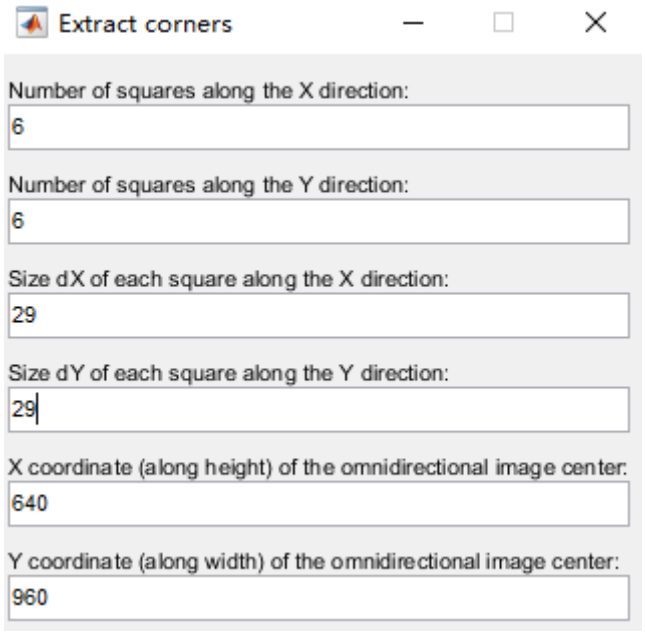

Figure 7. OCamCalib "Extract corners" menu

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 7).

Second, fill in 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, it 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). It 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 it, 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.

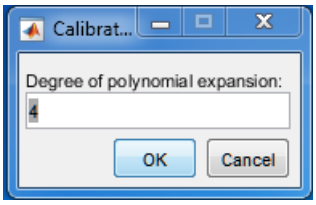

Figure 8. OCamCalib “Calibration” menu

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. The automatic detection of the image center is done by the 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 9).

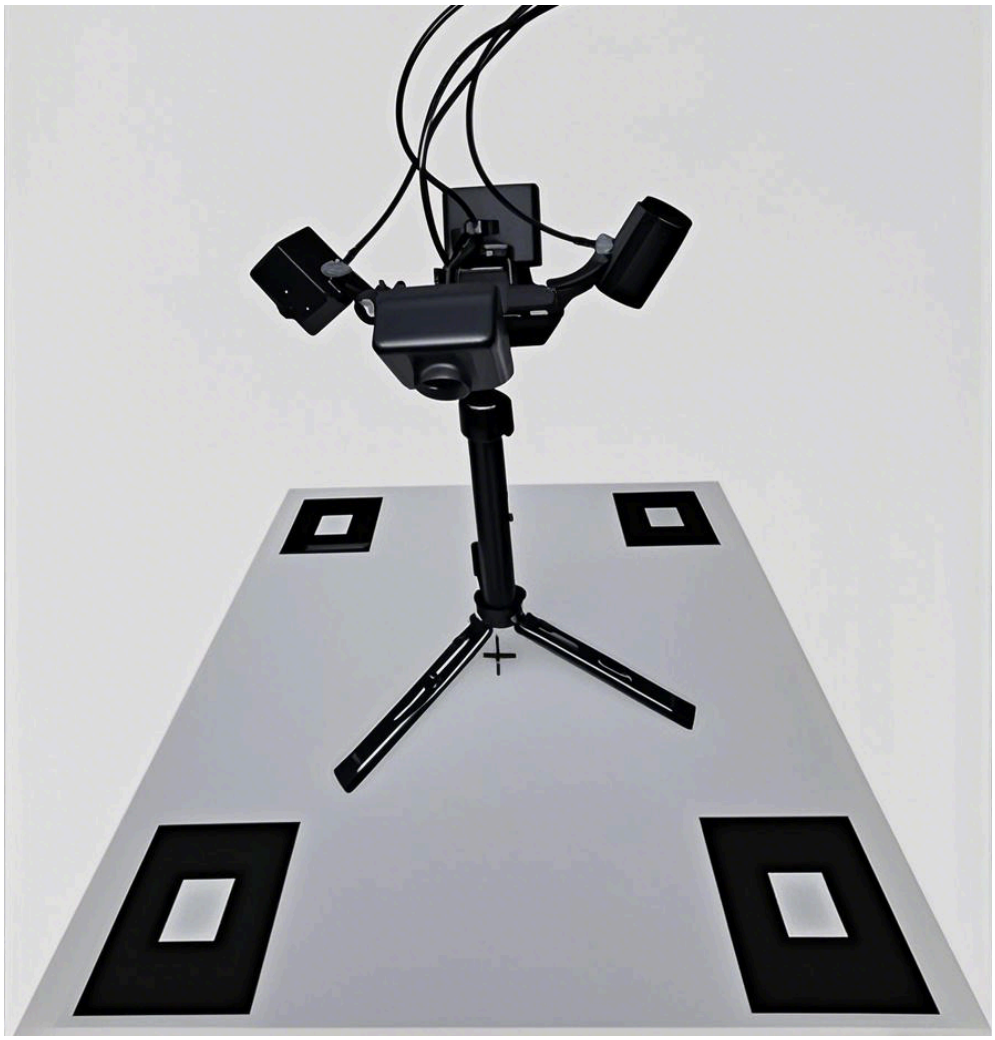

Figure 9. Extrinsic camera calibration setup

- 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 in the center of the big calibration pattern (see Figure 9).
- Use the *auto_calib_1.4* binary file located in the `/App/Build` SD card folder. See the applicable *Surround View Application Reference Manual* (document SVARM). 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.

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_1.4` 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 10.

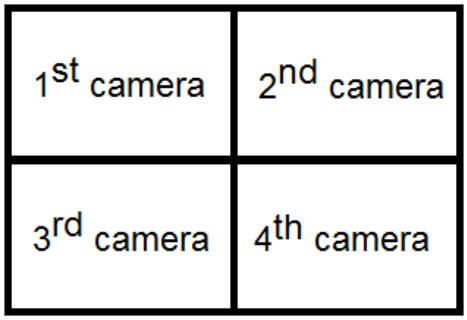

Figure 10. Camera view order

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 14

Figure 11. Fisheye camera views

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 are shown in Figure 12 and Figure 13 .

Figure 12. Non-fisheye camera views

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

Figure 13. Non-fisheye camera 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 14 appears in the terminal.

```
STEP 2: Search contours...
K:
[287.9112525904105, 0, 959.5;
 0, 288.1352138231671, 639.5;
 0, 0, 1]
K:
[291.8427977064738, 0, 959.5;
 0, 292.5027438617033, 639.5;
 0, 0, 1]
K:
[504.01824601668, 0, 959.5;
 0, 502.5422378521796, 639.5;
 0, 0, 1]
K:
[350.8678858267218, 0, 959.5;
 0, 352.0913914511757, 639.5;
 0, 0, 1]
```

Figure 14. Proper search contours terminal screenshot

The found contours are marked by the yellow color (see Figure 15).

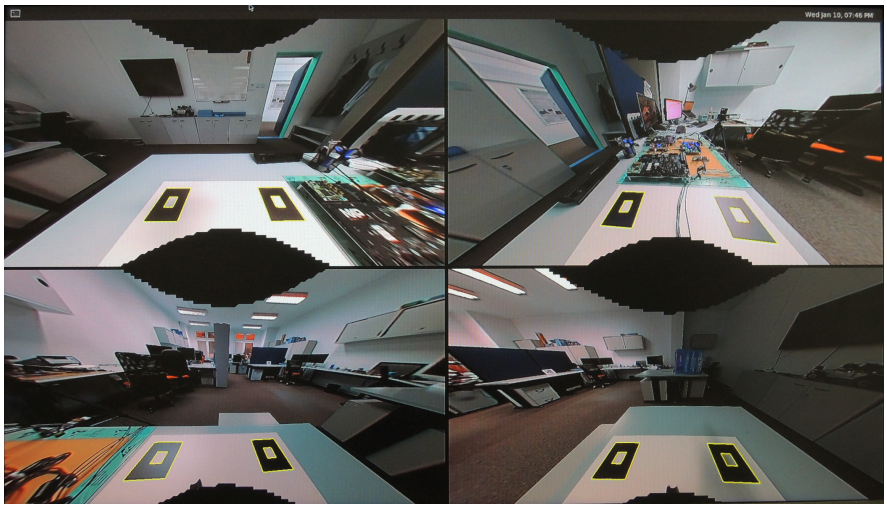

Figure 15. Found contours screenshot

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 16).

```
STEP 2: Search contours...
K:
[287.9112525904105, 0, 959.5;
 0, 288.1352138231671, 639.5;
 0, 0, 1]
Camera 0. The number of contours is fewer than 4. Change the calibration image
K:
[291.8427977064738, 0, 959.5;
 0, 292.5027438617033, 639.5;
 0, 0, 1]
Camera 1. The number of contours is fewer than 4. Change the calibration image
K:
[504.01824601668, 0, 959.5;
 0, 502.5422378521796, 639.5;
 0, 0, 1]
K:
[350.8678858267218, 0, 959.5;
 0, 352.0913914511757, 639.5;
 0, 0, 1]
```

Figure 16. 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 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 OpenCV with OpenVX and you see the terminal message shown in Figure 17, 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 17. 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* (document SVARM). and press the **F5** key to recalculate the meshes. Figure 18 and Figure 19 show the actual state of the screen and the terminal screenshot.

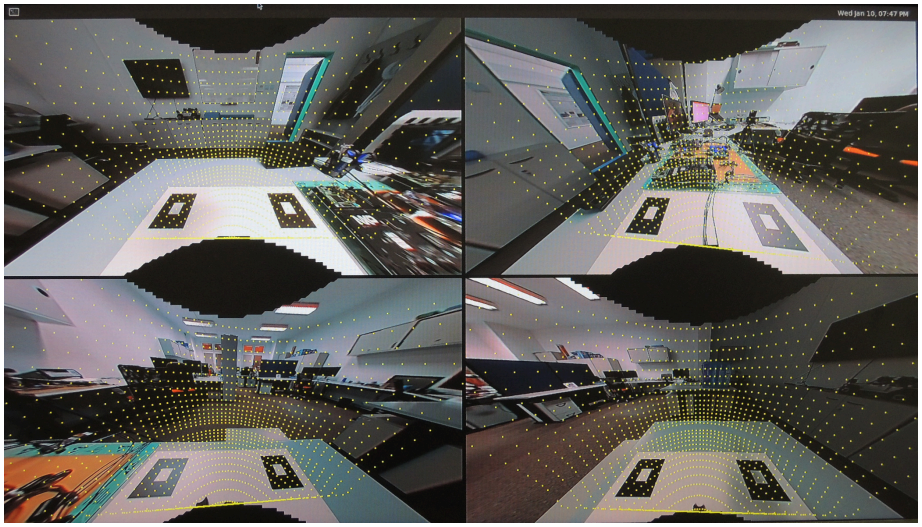

Figure 18. Prepare meshes views

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

Figure 19. Prepare meshes 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 resulting view does not look exactly like 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 and Figure 21 show 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.

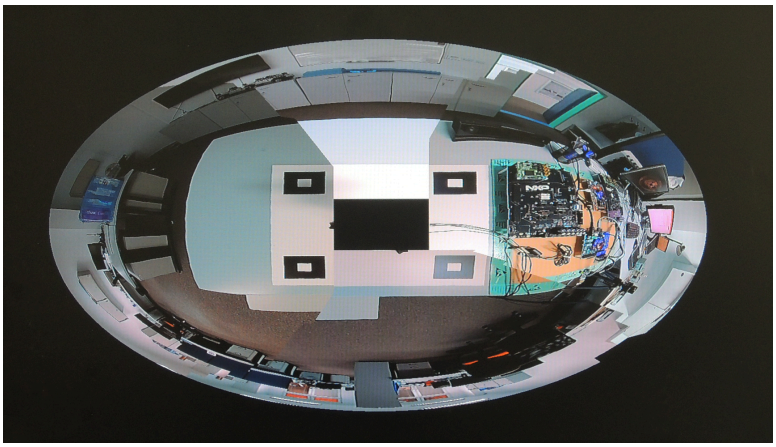

Figure 20. Result surround view

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

Figure 21. Result surround terminal screenshot

7 Real-Time Rendering application

The Real-Time Rendering application (called SV3D-1.4) 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 video files (demo mode). The Real-Time Rendering application stitches four input images into a single output and displays it using the GPU managed by the OpenGL ES 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), or video files.

It is assumed that you built the application from Surround View Application. The 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-1.4*. Figure 22 shows the output on the screen.

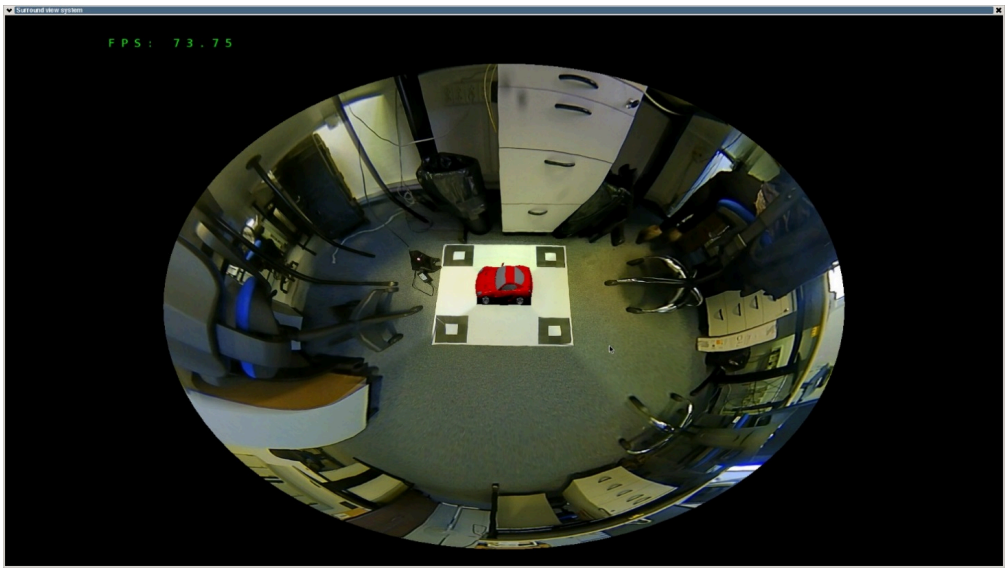

Figure 22. Composite 3D surround view

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.MX 6 / i.MX 7 / i.MX 8 / i.MX 9 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-omnidirectional-camera-calibration-toolbox-for-matlab
5. Matlab Runtime R2017a, available at www.mathworks.com
6. OCamCalib, "Omnidirectional Camera Calibration Toolbox for Matlab Runtime" available at https://github.com/nxp-imx/OCamCalib.

9 Revision history

Table 3. Revision history

Document ID	Release date	Description
SVAUG v.4.0	22 May 2025	i.MX 95 support is added.
SVAUG v.3.0	October 2022	Sections 3, 6.1, 6.2, 7 are updated.
SVAUG v.2.0	August 2022	Information about i.MX 8QXP is added.
SVAUG v.1.0	July 2018	Added Matlab Runtime for OCamCalib. Web links are updated.
SVAUG v.0.0	March 2018	Initial version

10 Note about the source code in the document

Example code shown in this document has the following copyright and BSD-3-Clause license:

Copyright 2025 NXP Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

1. Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.
2. Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials must be provided with the distribution.
3. Neither the name of the copyright holder nor the names of its contributors may be used to endorse or promote products derived from this software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN

CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

Legal information

Definitions

Draft — A draft status on a document indicates that the content is still under internal review and subject to formal approval, which may result in modifications or additions. NXP Semiconductors does not give any representations or warranties as to the accuracy or completeness of information included in a draft version of a document and shall have no liability for the consequences of use of such information.

Disclaimers

Limited warranty and liability — Information in this document is believed to be accurate and reliable. However, NXP Semiconductors does not give any representations or warranties, expressed or implied, as to the accuracy or completeness of such information and shall have no liability for the consequences of use of such information. NXP Semiconductors takes no responsibility for the content in this document if provided by an information source outside of NXP Semiconductors.

In no event shall NXP Semiconductors be liable for any indirect, incidental, punitive, special or consequential damages (including - without limitation - lost profits, lost savings, business interruption, costs related to the removal or replacement of any products or rework charges) whether or not such damages are based on tort (including negligence), warranty, breach of contract or any other legal theory.

Notwithstanding any damages that customer might incur for any reason whatsoever, NXP Semiconductors' aggregate and cumulative liability towards customer for the products described herein shall be limited in accordance with the Terms and conditions of commercial sale of NXP Semiconductors.

Right to make changes — NXP Semiconductors reserves the right to make changes to information published in this document, including without limitation specifications and product descriptions, at any time and without notice. This document supersedes and replaces all information supplied prior to the publication hereof.

Applications — Applications that are described herein for any of these products are for illustrative purposes only. NXP Semiconductors makes no representation or warranty that such applications will be suitable for the specified use without further testing or modification.

Customers are responsible for the design and operation of their applications and products using NXP Semiconductors products, and NXP Semiconductors accepts no liability for any assistance with applications or customer product design. It is customer's sole responsibility to determine whether the NXP Semiconductors product is suitable and fit for the customer's applications and products planned, as well as for the planned application and use of customer's third party customer(s). Customers should provide appropriate design and operating safeguards to minimize the risks associated with their applications and products.

NXP Semiconductors does not accept any liability related to any default, damage, costs or problem which is based on any weakness or default in the customer's applications or products, or the application or use by customer's third party customer(s). Customer is responsible for doing all necessary testing for the customer's applications and products using NXP Semiconductors products in order to avoid a default of the applications and the products or of the application or use by customer's third party customer(s). NXP does not accept any liability in this respect.

Terms and conditions of commercial sale — NXP Semiconductors products are sold subject to the general terms and conditions of commercial sale, as published at https://www.nxp.com/profile/terms, unless otherwise agreed in a valid written individual agreement. In case an individual agreement is concluded only the terms and conditions of the respective agreement shall apply. NXP Semiconductors hereby expressly objects to applying the customer's general terms and conditions with regard to the purchase of NXP Semiconductors products by customer.

Suitability for use in automotive applications — This NXP product has been qualified for use in automotive applications. If this product is used by customer in the development of, or for incorporation into, products or services (a) used in safety critical applications or (b) in which failure could lead to death, personal injury, or severe physical or environmental damage (such products and services hereinafter referred to as "Critical Applications"), then customer makes the ultimate design decisions regarding its products and is solely responsible for compliance with all legal, regulatory, safety, and security related requirements concerning its products, regardless of any information or support that may be provided by NXP. As such, customer assumes all risk related to use of any products in Critical Applications and NXP and its suppliers shall not be liable for any such use by customer. Accordingly, customer will indemnify and hold NXP harmless from any claims, liabilities, damages and associated costs and expenses (including attorneys' fees) that NXP may incur related to customer's incorporation of any product in a Critical Application.

Export control — This document as well as the item(s) described herein may be subject to export control regulations. Export might require a prior authorization from competent authorities.

HTML publications — An HTML version, if available, of this document is provided as a courtesy. Definitive information is contained in the applicable document in PDF format. If there is a discrepancy between the HTML document and the PDF document, the PDF document has priority.

Translations — A non-English (translated) version of a document, including the legal information in that document, is for reference only. The English version shall prevail in case of any discrepancy between the translated and English versions.

Security — Customer understands that all NXP products may be subject to unidentified vulnerabilities or may support established security standards or specifications with known limitations. Customer is responsible for the design and operation of its applications and products throughout their lifecycles to reduce the effect of these vulnerabilities on customer's applications and products. Customer's responsibility also extends to other open and/or proprietary technologies supported by NXP products for use in customer's applications. NXP accepts no liability for any vulnerability. Customer should regularly check security updates from NXP and follow up appropriately.

Customer shall select products with security features that best meet rules, regulations, and standards of the intended application and make the ultimate design decisions regarding its products and is solely responsible for compliance with all legal, regulatory, and security related requirements concerning its products, regardless of any information or support that may be provided by NXP.

NXP has a Product Security Incident Response Team (PSIRT) (reachable at PSIRT@nxp.com) that manages the investigation, reporting, and solution release to security vulnerabilities of NXP products.

NXP B.V. — NXP B.V. is not an operating company and it does not distribute or sell products.

Trademarks

Notice: All referenced brands, product names, service names, and trademarks are the property of their respective owners.

NXP — wordmark and logo are trademarks of NXP B.V.

MATLAB — is a registered trademark of The MathWorks, Inc.

Contents