Application Notes

Getting Started with Camera Calibration

THE IMAGINGSOURCE
TECHNOLOGY BASED ON STANDARDS

Table of Contents

1. What is camera calibration 3
   1.1 Applications 4
2. Programming example and results 5
   2.1 Calibration result 5
   2.2 Application Example (C++) 5
3. Calibration Procedure 8

1. What Is Camera Calibration?

Camera calibration is the process of determining a camera's intrinsic parameters and lens distortion to mathematically correct captured images. The resulting calibration data is used to remove geometric distortion, straighten lines, and ensure that image measurements are accurate and consistent across the entire field of view.

An example image taken with a 3.5 mm lens looks like this in its unprocessed form:

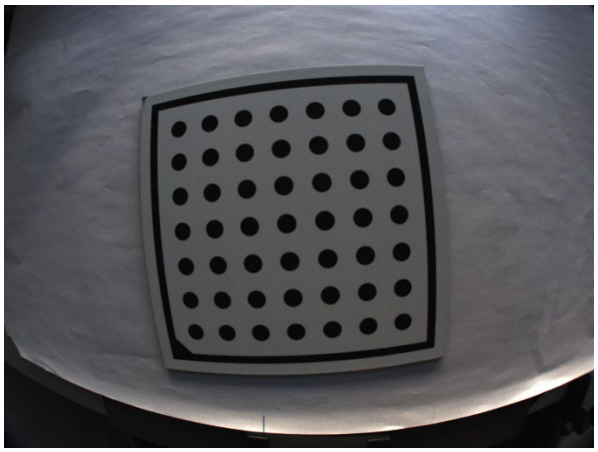


Fig. 1 Example of a distorted image.

After processing, it looks like this:

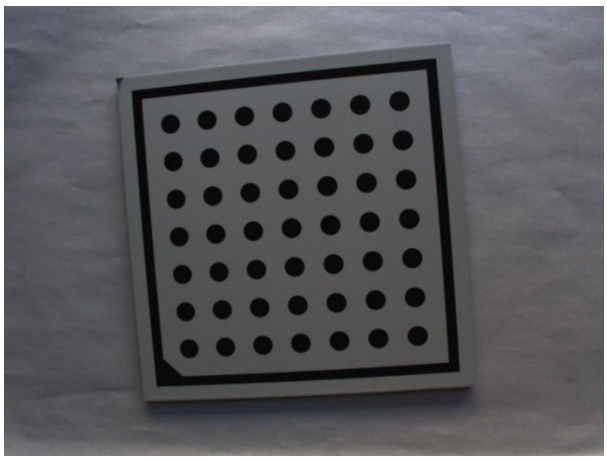


Fig. 2 Example of undistorted image.

This is an extreme example, but it clearly shows how the original image from the camera is altered. The curved lines become straight. The field of view becomes slightly smaller.

The "*cameracalibration"* program uses OpenCV for calibration. This calibration process is well documented at
https://docs.opencv.org/4.x/dc/dbb/tutorial_py_calibration.html.

Applications that should use the data generated by this program must do so using OpenCV.

1.1 Applications

Intrinsic calibration is used in:

- Metrology, where measurements become more accurate, including near the edges of the image.
- Stitching images from multiple cameras.
- 3D stereo systems, where both cameras must provide an equally sized, undistorted image region.

2. Programming Example and Results

2.1 Calibration Result

A camera calibration produces an XML file with the following example content:

```
<?xml version="1.0"?>
<opencv_storage>
<image_size>
  2048 1536</image_size>
<Camera_Matrix type_id="opencv-matrix">
  <rows>3</rows>
  <cols>3</cols>
  <dt>d</dt>
  <data>
    823.01087878946305 0. 1039.0799012738162 0. 820.80168824546922
    733.08149016169932 0. 0. 1.</data></Camera_Matrix>
<Distortion_Coefficients type_id="opencv-matrix">
  <rows>5</rows>
  <cols>1</cols>
  <dt>d</dt>
  <data>
    -0.16717370174894627 0.029150863272593131 -0.0026241392403182637
    -8.5319662569887086e-05 -
0.0023071518373398033</data></Distortion_Coefficients>
</opencv_storage>
```

This XML file was created using the OpenCV “FileStorage” class, https://docs.opencv.org/3.4/da/d56/classcv_1_1FileStorage.html, and can easily be loaded into a custom application using this class. If supported by the camera being used, this XML file is also stored in the camera.

The image size that was used to calculate the matrices is also stored in the XML file, allowing an application to ensure that the correct width and height are configured in the camera.

2.2 Application Example (C++)

The following C++ source code shows how a “camera.xml” file is opened and how the data is read from it.

This data is then applied to the camera - specifically the width and height - and subsequently to an image captured by the camera.

```
#include <string.h>
#include <iostream>
#include <ic4/ic4.h>
#include <ic4-interop/interop-OpenCV.h>

#include <opencv2/core.hpp>

int main()
```

```
{
	ic4::InitLibraryConfig cfg;
	cfg.apiLogLevel = ic4::LogLevel::Error;
	cfg.defaultErrorHandlerBehavior = ic4::ErrorHandlerBehavior::Throw;
	ic4::initLibrary(cfg);
	std::atexit(ic4::exitLibrary);
	ic4::Error err;
	ic4::Grabber grabber;
	auto sink = ic4::SnapSink::create(ic4::PixelFormat::BGR8);
	std::cout << "rectify_opencv" << std::endl;

	if (!grabber.deviceOpen("DFK 33UX264", err))
	{
		std::cerr << "Failed to open device: " << err.message() <<
std::endl;
		return -2;
	}

	cv::FileStorage fs("camera.xml, cv::FileStorage::READ |
cv::FileStorage::MEMORY);
	// Extract the calibration matrixes
	cv::Mat CameraMatrix;
	cv::Mat DistortionCoeffs;
	cv::Size image_size;
	fs["image_size"] >> image_size;
	fs["Camera_Matrix"] >> CameraMatrix;
	fs["Distortion_Coefficients"] >> DistortionCoeffs;

	grabber.devicePropertyMap().setValue(ic4::PropId::Width,
image_size.width);
	grabber.devicePropertyMap().setValue(ic4::PropId::Height,
image_size.height);
	grabber.devicePropertyMap().setValue(ic4::PropId::AcquisitionFrameRat
e, 30.0);

    if (!grabber.streamSetup(sink,NULL,
ic4::StreamSetupOption::AcquisitionStart, err))
	{
		std::cerr << "Failed to setup stream: " << err.message() <<
std::endl;
		return -4;
	}

	// Snap an image, save it as original, rectify (undistort) it
	// and save it as "rectified".
	auto buffer = sink->snapSingle(1000);
	ic4::imageBufferSaveAsJpeg(*buffer, "Original.jpg");
	auto dstImage = ic4interop::OpenCV::wrap(*buffer);
	auto srcImage = ic4interop::OpenCV::copy(*buffer);
	cv::undistort(srcImage, dstImage, CameraMatrix, DistortionCoeffs);
	ic4::imageBufferSaveAsJpeg(*buffer, "rectified.jpg");
	grabber.streamStop();
	grabber.deviceClose();

	return 0;
}
```



Note:

Installing OpenCV for C++ on Windows is not entirely straightforward. It is documented at https://docs.opencv.org/4.x/d3/d52/tutorial_windows_install.html.

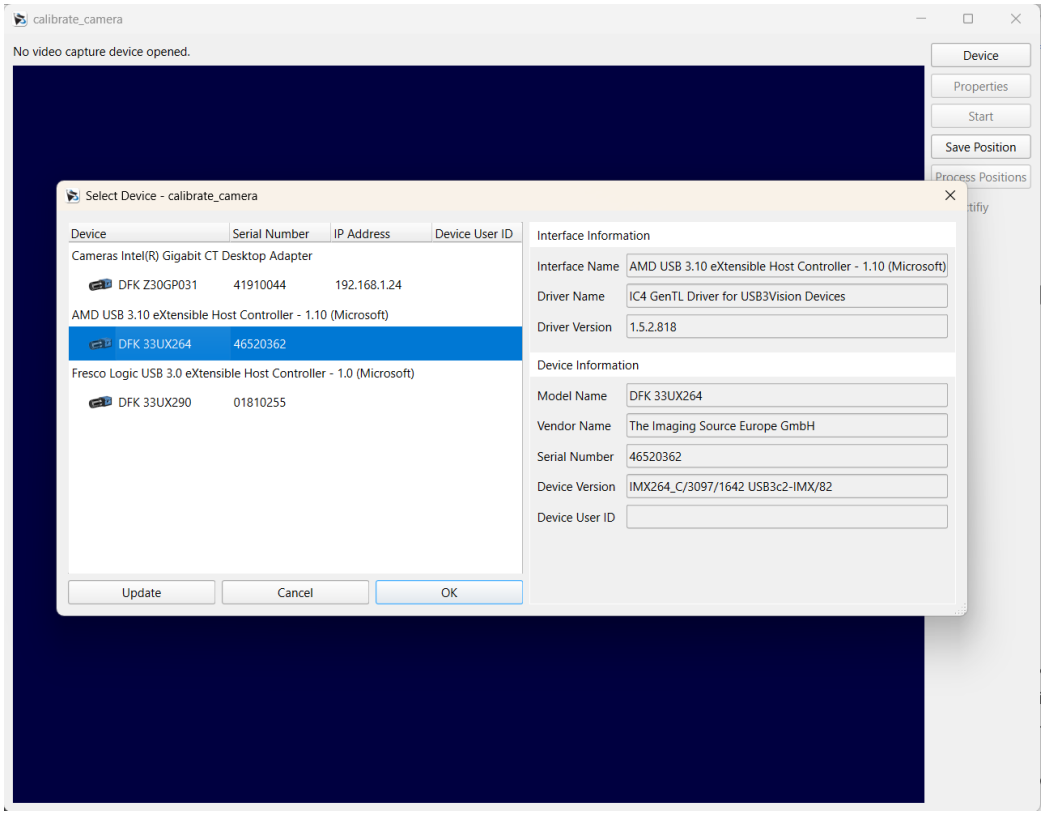
On Linux, OpenCV is available through the package repositories, which makes installation much easier.

3. Calibration Procedure

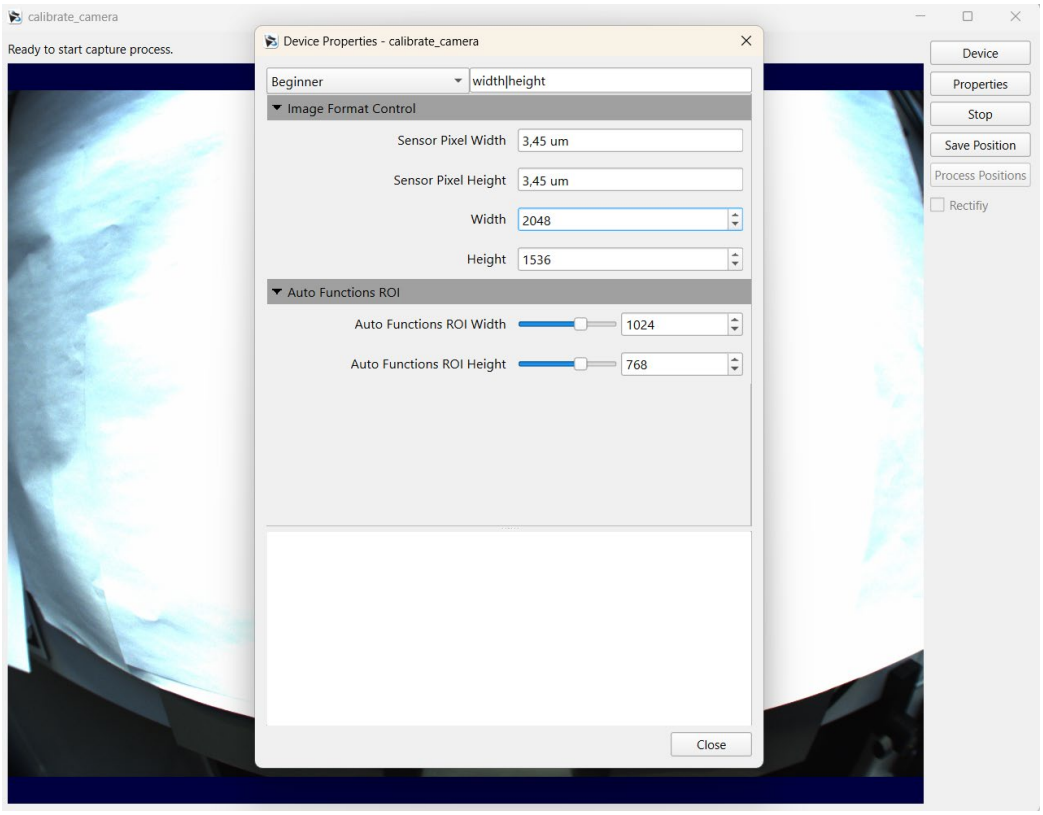
Calibration is performed using the “*cameracalibration*” program.

First, a 7×7 point grid is required. A suitable HTML file is provided in the point-grid directory. This file can be scaled appropriately when printing. It should be printed large enough so that it fits into the camera image approximately six times. The printout may also be smaller.

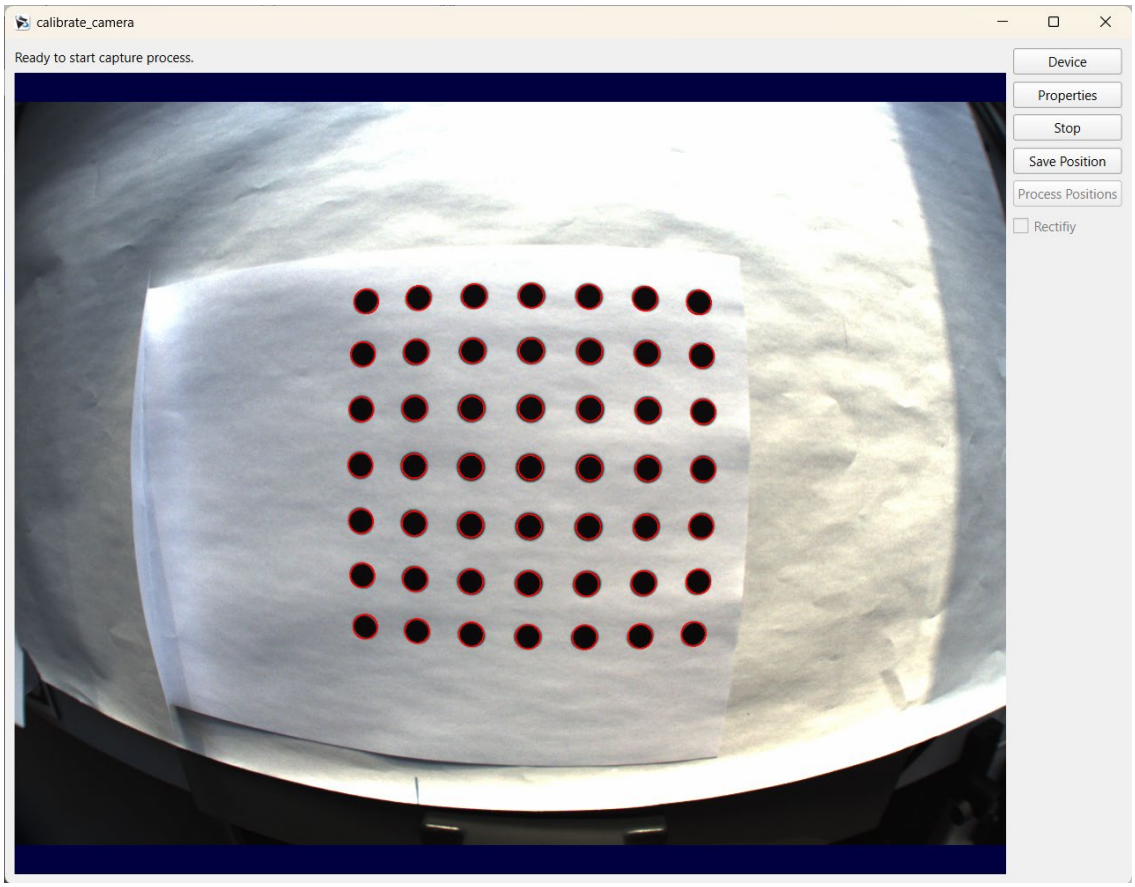
After starting the “*cameracalibration*” program, the camera is selected:



Then the desired resolution is set:

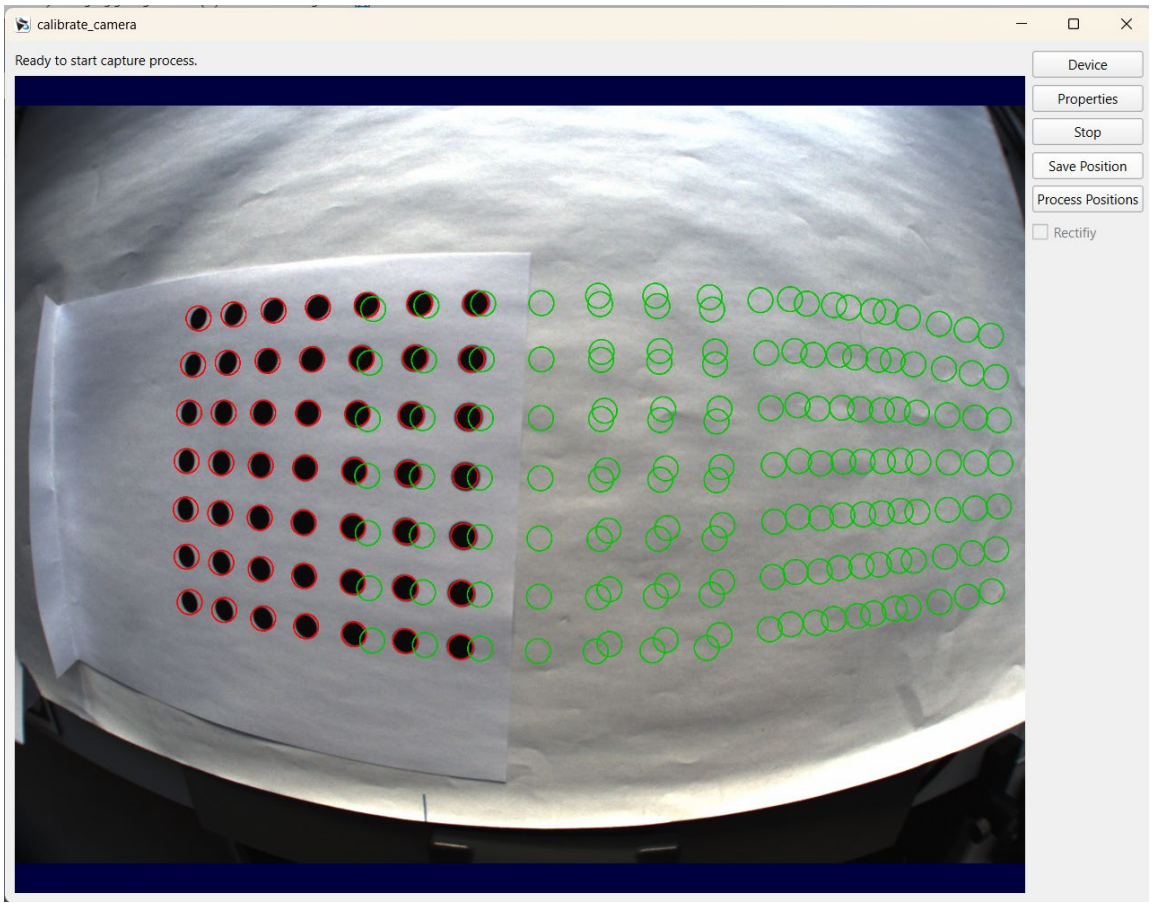


The point grid must be positioned approximately in the center of the image for the first position. In addition, the background should be as uniform as possible; otherwise, the grid may not be detected.

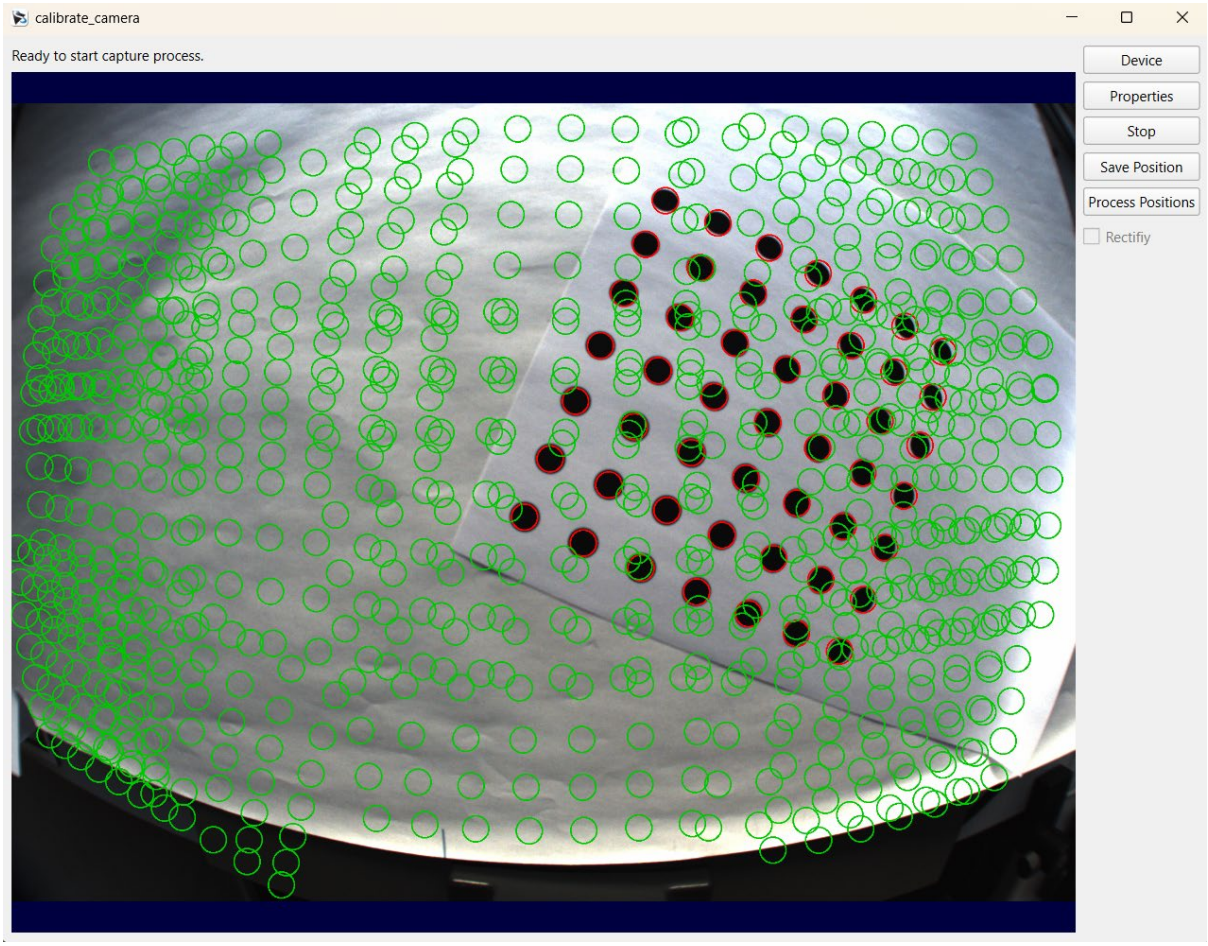


Next, the *Save Position* button is pressed to store the first point positions. The grid must be centered in the image, because internally the pixel distance between two points is calculated at the center of the grid.

The point grid is then moved within the camera's field of view. After each movement, the *Save Position* button should be clicked. The positions of the point grid in the image must overlap.



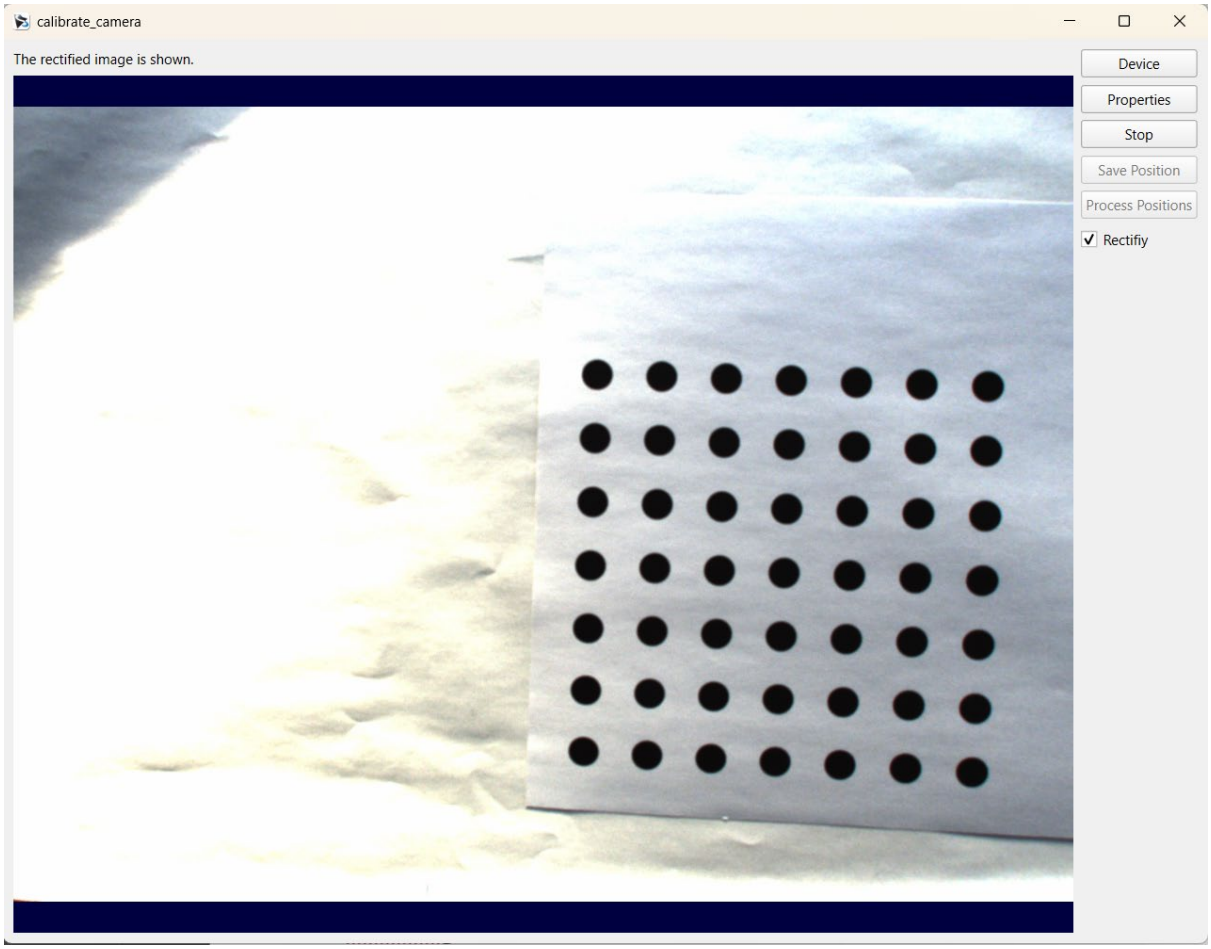
This process is repeated until the entire image is filled with green circles. In the corners, it may happen that the point grid is not detected. In that case, the grid should be moved into the corners as far as it can still be detected.



The *Process Positions* button should now be clicked to start the process of generating the camera matrix and the distortion matrix. The processing time depends on the number of saved point grid positions. After clicking *Process Positions*, the name of the XML file in which the data should be stored is requested first; only then does the processing start. If *Process Positions* is clicked accidentally, *Cancel* can be selected in the *Save File* dialog. This closes the dialog and allows additional positions to be saved.

The image width and height, the camera matrix, and the distortion matrix are stored in the specified XML file. If supported by the camera, this XML file is also saved directly to the camera.

Once the calculation is complete, the undistorted image is displayed:



Programming examples for applying the calibration can be found at:

https://github.com/TheImagingSource/ic4-examples

Revision History

Document Number	Date	Changes
AN20260123-01	23 January 2026	Initial release version of this document.

North and South America: The Imaging Source, LLC
4600 Park Road, Suite 470, Charlotte, NC 28209, USA
Phone: +1 704-370-0110

Asia Pacific: The Imaging Source Asia Co. Ltd.
3F., No. 43-7/8, Zhongxing Road,
New Taipei City, Xizhi District 221012, Taiwan
Phone: +886 2-2792-3153

www.theimagingsource.com

All product and company names in this document may be trademarks and tradenames of their respective owners and are hereby acknowledged.

The Imaging Source Europe GmbH cannot and does not take any responsibility or liability for any information contained in this document. The source code presented in this document is exclusively used for didactic purposes. The Imaging Source does not assume any kind of warranty expressed or implied, resulting from the use of the content of this document or the source code.

The Imaging Source Company reserves the right to make changes in specifications, function or design at any time and without prior notice.

Copyright ©2026 The Imaging Source Europe GmbH, AppNotes_GettingStartedWithCameraCalibration_en_US.pdf

All rights reserved. Reprint, also in parts, only allowed with permission of The Imaging Source Europe GmbH.

All weights and dimensions are approximate. Unless otherwise specified, lenses shown in camera product images are not included with the cameras.

Headquarters: The Imaging Source Europe GmbH
Überseetor 18, 28217 Bremen, Germany
Phone: +49 421 33591-0