Spectacular AI SDK

Calibration manual

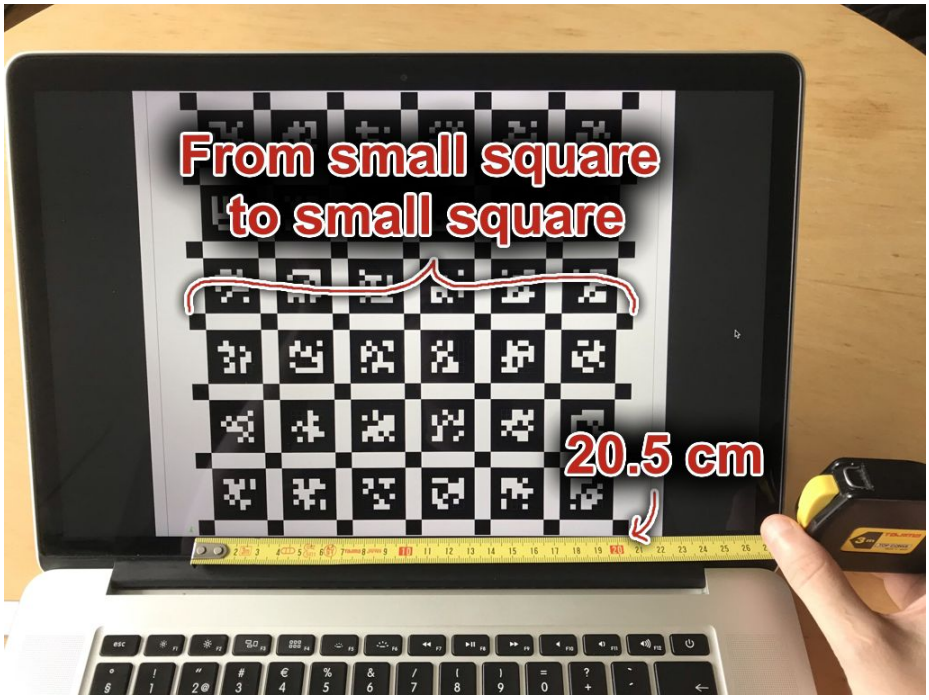
Introduction

This document describes the calibration aspects that are relevant to the Spectacular AI SDK. The target audience are users of the *core SDK* who wish to perform the camera calibration manually. For the *wrapper SDKs* (e.g., OAK-D or RealSense wrappers), this is not required as the factory calibration parameters are provided by the devices and automatically used by the SDK.

To start using manual calibration with Spectacular AI SDK, it is recommended to start with a *calibration test* in collaboration with Spectacular AI engineers. This provides a reference that helps the integration with the customers' camera calibration pipeline. The calibration test is described in the first section of this document.

The second part of the document is a self-contained description of the mathematical calibration models and coordinate systems used by the SDK, intended as a reference for integration.

Calibration test



Example: simple AprilGrid calibration using a computer screen: measure and divide by 8.1 to get `tagSize`.

In the calibration test, a single device individual is calibrated in collaboration with Spectacular AI engineers.

The recommended way to perform the test is using the built-in recording function in Spectacular AI SDK to record a calibration sequence. If the SDK has not yet been integrated to enable this, it is also possible to record camera (and preferably also IMU) data using other software and convert the data to a suitable format in collaboration with Spectacular AI.

A calibration test sequence should view a calibration pattern from multiple angles. The calibration pattern may be displayed on a computer screen or printed on a flat surface. Larger screens may improve accuracy of the calibration. In the calibration test, the IMU is also calibrated from the data and, consequently, the **calibration target must be stationary** and the device should be moved around it.

The following patterns are supported:

- AprilGrid (link to PDF)

- Checkerboard

The size of the calibration pattern should be defined in the YAML format supported by the *Kalibr* software. For example

```
target_type: 'aprilgrid'
tagCols: 6               # number of apriltags
tagRows: 6               # number of apriltags
tagSize: 0.037654321     # size of apriltag, edge to edge [m]
tagSpacing: 0.3          # ratio of space between tags to tagSize
```

The recommended resolution for calibration image data is the maximum sensor resolution. However, if this cannot be achieved, the resolution should be at least 400x400. If IMU data is included, it should be recorded at 50Hz or more (500Hz recommended). If IMU data cannot be recorded, then approximate IMU-to-camera extrinsics (see the sections below for more information) should be provided. If the calibrated system includes stereo camera pairs, the frames recorded from stereo camera pairs must be synchronized.

The recorded sequence should be sent to Spectacular AI, who will calibrate the device based on the data and send a calibration file, which can be used as a reference and example for the subsequent steps in SDK integration.

Calibration specification

File format

The calibration is provided to the SDK in JSON format. Spectacular AI will provide a reference as the results of a successful *calibration test*. An example calibration file for a stereo-camera-IMU-system is given below.

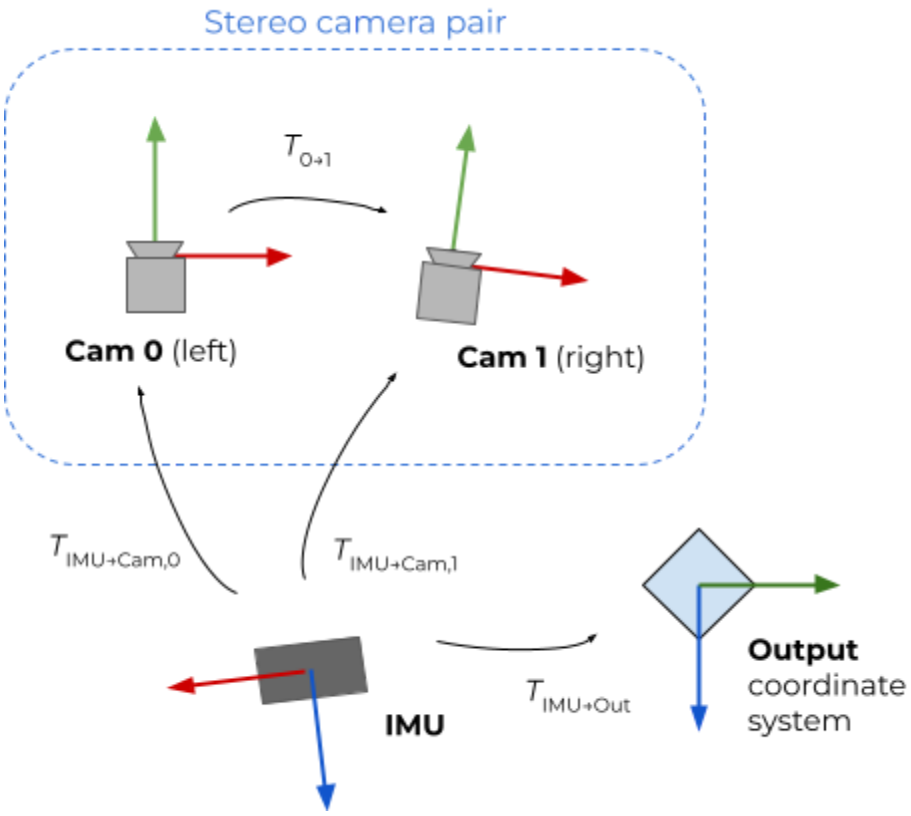

```
{
  "cameras": [
    {
      "imageWidth": 1280,
      "imageHeight": 800,
      "focalLengthX": 689.9600212721717,
      "focalLengthY": 689.7791814512566,
      "principalPointX": 625.7728119663589,
      "principalPointY": 406.30847173743695,
      "model": "kannala-brandt4",
      "distortionCoefficients": [-0.042199872,-0.0024873,-0.0156296,0.008040966],
      "imuToCamera": [

[-0.007597321889990516,-0.9999685028233531,-0.0022965324560711986,0.003925088167884679],

[-0.028027852548307086,-0.0020827542464345594,0.9996049727848895,-0.002080025490845079],
       [-0.9995782711632094,0.007658687614133075,-0.028011146395656494,-0.06311860979590438],
       [0.0,0.0,0.0,1.0]
      ]
    },
    {
      "imageWidth": 1280,
      "imageHeight": 800,
      "focalLengthX": 689.6159071698686,
      "focalLengthY": 689.3776100206506,
      "principalPointX": 637.155260132079,
      "principalPointY": 410.031637138216,
      "model": "kannala-brandt4",
      "distortionCoefficients": [-0.0381701,-0.015025785,0.0042020,-0.0005575143],
      "imuToCamera": [

[-0.008900660156368811,-0.9999585078949196,-0.0019392620624486545,-0.12881566945954037],

[-0.014472758680460995,-0.0018103137813196835,0.9998936253523123,-0.0004115181854138228],
       [-0.9998556483337772,0.008927779823628468,-0.014456045187493105,-0.06294064508330674],
       [0.0,0.0,0.0,1.0]
      ]
    }
  ],
  "imuToOutput": [
    [0.06859197197751811,-0.9973466692339874,-0.024387758160742287,-0.0],
    [0.995903321632435,0.06700802688203558,0.06071654053764488,0.0],
    [-0.05892126391820337,-0.028452516606581855,0.9978570734113344,0.04],
    [0.0,0.0,0.0,1.0]
  ]
}
```

The colors in the above markup listing highlight the role of the different parameters as follows:

- Intrinsic camera calibration
  - left camera
  - right camera
- Extrinsic calibration parameters
  - Left IMU-to-camera matrix ($T_{IMU \to Cam,0}$)
  - Right IMU-to-camera matrix ($T_{IMU \to Cam,1}$)
  - IMU to output reference matrix ($T_{IMU \to Out}$)

The meaning of each parameter is explained in more detail in the following sections.

Extrinsic coordinate systems



The extrinsic calibration models the geometry of the device, from the point of view of the IMU-camera system. In Spectacular AI SDK, the system is assumed to consist of an IMU sensor and one or more cameras. The first two cameras in the system may form a stereo camera pair.

The extrinsic calibration is specified as 4x4 homogeneous coordinate transformation matrices. For a system with one stereo camera pair, one must specify:

- $T_{\text{IMU}\to\text{Cam},0}$: IMU-to-camera matrix for the left camera
- $T_{\text{IMU}\to\text{Cam},1}$: IMU-to-camera matrix for the right camera

Optionally, a matrix $T_{\text{IMU}\to\text{Out}}$ (IMU-to-output matrix) can be given to specify the reference point and local coordinate axis directions that the SDK will use for its output.

The IMU-to-output can be convenient in a case where the SDK output is compared to an external reference system, such as VIVE trackers. Specifying the VIVE tracker reference point and orientation using $T_{\text{IMU}\to\text{Out}}$ allows direct comparison of the poses returned by the two systems with SE(3) or SE(2)-aligned metrics.

Spectacular AI uses the **OpenCV camera coordinate convention** (x = left, y = down, z = forward). This affects the form of the IMU-to-camera matrices. See Appendix for more details.

Intrinsic calibration

Intrinsic calibration models the optics of the cameras and possible digital deformations of the image in the camera pipeline. Spectacular AI SDK uses a small-aperture approach that is typical for computer vision. Depth of field or chromatic aberration effects are not modeled (see Appendix for more details).

All models have four common parameters: (f_x, f_y, c_x ,c_y), which describe the “focal length” (exact interpretation depends on the camera model) and principal point. They should be given as separate JSON fields (see JSON format description above), e.g., `focalLengthX`. The rest of the model parameters are given in a `distortionCoefficients` array and the type of the camera model in the `model` field.

The SDK currently supports the following types of calibration models:

1. Brown-Conrady radial-tangential models without thin-prism terms
    a. Undistorted pinhole: no distortion coefficients (only (f_x, f_y, c_x ,c_y))
    `"model": "pinhole"`
    b. 3-coefficient radial variant:
    `"model": "pinhole",`
    `"distortionCoefficients": [`k_1, k_2, k_3`]`
    c. 8-coefficient variant (Brown-Conrady / Inverse Brown-Conrady)
    `"model": "brown-conrady",`
    `"distortionCoefficients": [`k_1, k_2, p_1, p_2, k_3, k_4, k_5, k_6`]`
    d. 5-coefficient variant, a.k.a. “pinhole-radtan”
    `"model": "brown-conrady",`
    `"distortionCoefficients": [`k_1, k_2, p_1, p_2, k_3`,0,0,0]`

2. Kannala-Brandt
    a. Kannala-Brandt-4, radially symmetric
    `"model": "kannala-brandt4",`
    `"distortionCoefficients": [`k_0, k_1, k_2, k_3`]`
    b. Full Kannala-Brandt-18 (ask us for a reference implementation)

3. OpenCV “omnidir” camera model (Mei & Rivers, 2007):
    `"model": "omnidir",`
    `"distortionCoefficients": [`k_1, k_2, s, ξ, p_1, p_2`]`

The values of the calibration parameters depend on the manner the image is rescaled, cropped or undistorted by the camera pipeline before being input to the SDK. If these aspects are changed, the calibration needs to be modified accordingly.

One possible non-trivial modification to the images is undistortion, which may be applied to the images before they are given as input to the SDK. After undistortion, the images are typically assumed to obey the simple undistorted pinhole camera model. This may also be performed simultaneously with *stereo rectification*, which can also effectively rotate the camera coordinate systems. Extra care should be taken in this case to ensure that the extrinsic calibration parameters remain consistent with the rectified image data (see Appendix for more details).

Accuracy requirements

Extrinsics and intrinsics

The transformation between the two stereo cameras, $T_{0\to1}$, must be accurate and consistent with the intrinsic camera calibration parameters. It is recommended to compute this quantity with a calibration system that jointly optimizes these parameters (see the section below for more information). The average residual reprojection error should be less than 0.3 pixels (RMSE).

The transformation between IMU and the stereo camera system is not required to be highly accurate. The error in IMU-to-camera matrix should be

- less than 3 degrees in orientation
- less than 10% in translation (or less than 3 millimeters, whichever is greater)

The as-designed extrinsics from a CAD model can typically be assumed to be sufficiently accurate. The camera calibration, including intrinsic calibration and the extrinsic calibration within stereo camera pairs must be performed per device individual.

In practice, a less accurate IMU-to-camera extrinsic information should be combined with more accurate stereo extrinsics by using the mathematical identities described in the Appendix.

Optics and 3A

The use of auto-focus or variable-focus lenses is not recommended as changing the focus affects the calibration parameters in a hard-to-predict manner. Optical or electronic image stabilization (OIS/EIS) should not be used.

Auto-exposure and auto-white-balance (in case of color data) may be used. The pixel data that is fed to the SDK can be linear or gamma-corrected.

Timing

The IMU and camera timestamps should be synchronized to a precision of 1 millisecond. However, delays up to 30 ms can be tolerated by enabling an online time-shift calibration feature in the SDK.

The timestamps of the camera frames given to the SDK should represent the midpoint of exposure.

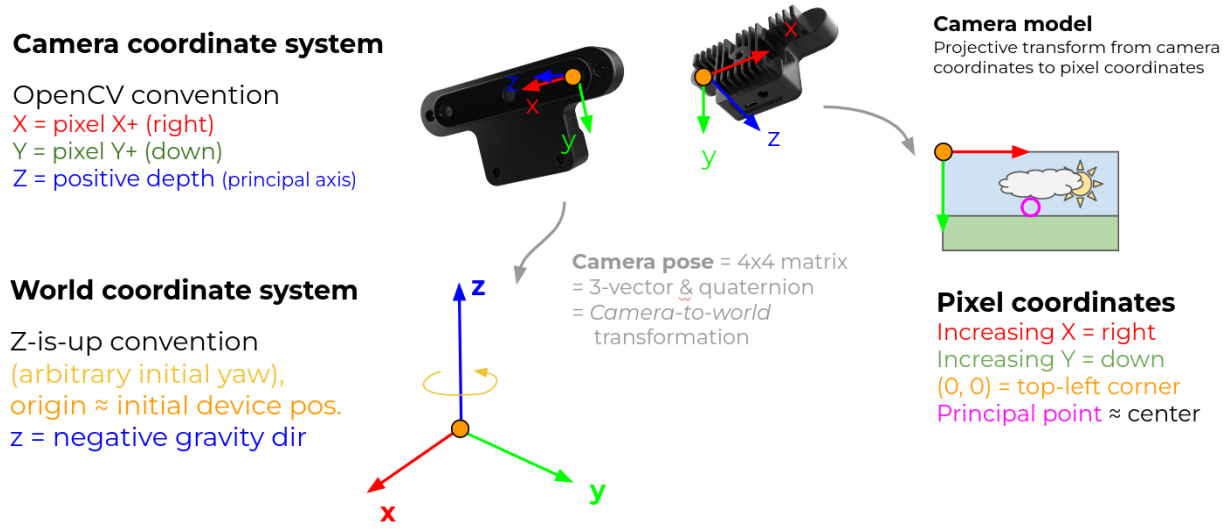
Appendix - mathematical details

This appendix elaborates the mathematical details of some of the concepts referred to in the main document.

Spectacular AI SDK coordinate system conventions

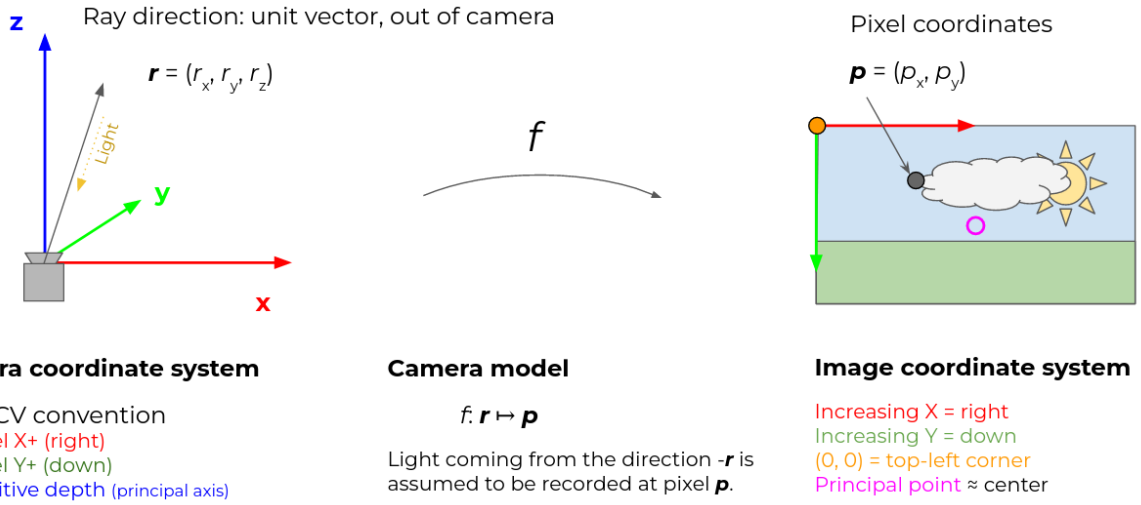
Spectacular AI SDK uses the following coordinate conventions, which are also illustrated in the image below

- **World coordinate system**: Right-handed Z-is-up
- **Camera coordinate system**: OpenCV convention:
  X = left, Y = down, Z = forward.
- **IMU coordinate system**: assumed to be right-handed (and use SI units)
- **Local output coordinate system** ("reference point"). Can be specified in the calibration data. Default: IMU coordinates



Camera models

Spectacular AI SDK utilizes a simplified camera model typical for computer vision: the origin of the camera coordinate system acts as a single point-like center of projection. The light captured by the camera is thought to arrive directly at this point from different directions. The incident direction of a ray of light is assumed to determine the (fractional) coordinates of the pixel it contributes to.



The intrinsic calibration model is a function that maps an outbound ray direction $\boldsymbol{r} = (r_x, r_y, r_z)$ to pixel $\boldsymbol{p}$, that is, light coming from the direction opposite to $\boldsymbol{r}$ is assumed to be contributing to the pixel with coordinates $\boldsymbol{p} = (p_x, p_y)$.

The camera model function $f_{\boldsymbol{\theta}}: (r_x, r_y, r_z) \mapsto (p_x, p_y)$ depends on a vector of parameters $\boldsymbol{\theta}$, which are determined by a calibration procedure. One of the simplest widely used models is the undistorted pinhole camera model with 4 parameters $\boldsymbol{\theta} = (f_x, f_y, c_y, c_x)$, defined as

$$p_x = r_x / r_z \cdot f_x + c_x$$
$$p_y = r_y / r_z \cdot f_y + c_y$$

In a stereo camera system, the calibration typically involves simultaneously optimizing, $T_{L \to R}$, the extrinsic transformation between the cameras and the calibration parameters $\boldsymbol{\theta}_L$, $\boldsymbol{\theta}_R$ for each camera in the stereo camera pair.

Brown-Conrady

Applied in practice by first computing $x = p_x$ and $y = p_y$ using the pinhole model above and then applying the distortion coefficients $(p_1, p_2, k_1, k_2, k_3, k_4, k_5, k_6)$ as follows

$$p_x' = x\,C + 2\,p_1\,x\,y + p_2(r^2 + 2\,x^2)$$
$$p_y' = y\,C + p_1(r^2 + 2\,y^2) + 2\,p_2\,x\,y$$

where $r^2 = x^2 + y^2$ and $C = (1 + k_1 r^2 + k_2 r^4 + k_3 r^6) / (1 + k_4 r^2 + k_5 r^4 + k_6 r^6)$.

Kannala-Brandt

Applied by first converting the ray direction $\boldsymbol{r}$ to polar coordinates θ, φ, then applying

$$(x', y') = (r(\theta) + \Delta_r(\theta, \varphi))\ \boldsymbol{u}_r(\varphi) + \Delta_t(\theta, \varphi)\ \boldsymbol{u}_\varphi(\varphi)$$

the radial and tangential directions are

$$\boldsymbol{u}_r(\varphi) = (\cos(\varphi), \sin(\varphi)) \qquad \text{and} \qquad \boldsymbol{u}_\varphi(\varphi) = (-\sin(\varphi), \cos(\varphi)),$$

and the distortion terms are defined from the 18 coefficients (k_0, k_1, k_2, k_3, l_1, l_2, l_3, i_1, i_2, i_3, i_4, m_1, m_2, m_3, j_1, j_2, j_3, j_4) as

$$\begin{aligned}
r(\theta) &= \theta\,(1 + k_0\theta^2 + k_1\theta^4 + k_2\theta^6 + k_3\theta^8) \\
\Delta_r(\theta, \varphi) &= (l_1\theta + l_2\theta^3 + l_3\theta^5) \cdot (i_1 \cos\varphi + i_2 \sin\varphi + i_3 \cos 2\varphi + i_4 \sin 2\varphi) \\
\Delta_t(\theta, \varphi) &= (m_1\theta + m_2\theta^3 + m_3\theta^5) \cdot (j_1 \cos\varphi + j_2 \sin\varphi + j_3 \cos 2\varphi + j_4 \sin 2\varphi).
\end{aligned}$$

In the Kannala-Brandt-4 model, the other terms vanish and the model simplifies to

$$(x', y') = r(\theta)\ \boldsymbol{u}_r(\varphi)$$

Finally, the pinhole intrinsics are applied as

$$\begin{aligned}
p_x &= x' \cdot f_x + c_x \\
p_y &= y' \cdot f_y + c_y.
\end{aligned}$$

Implementation note: Trigonometric functions can mostly be avoided in the computations. The full Kannala-Brandt model can be applied as

$$\begin{aligned}
\theta &= \cos^{-1}(r_z / \operatorname{sqrt}(r_x^2 + r_y^2 + r_z^2)) \\
(c, s) &= \texttt{safeNormalize}(r_x, r_y) && = (r_x, r_y) / \operatorname{sqrt}(r_x^2 + r_y^2) = (\cos(\varphi), \sin(\varphi)) \\
c' &= 1 - 2s^2 && = \cos(2\varphi) \\
s' &= 2\,s\,c && = \sin(2\varphi) \\
t &= \theta^2 \\
r(\theta) &= \theta\,(1 + t(k_0 + t(k_1 + t(k_2 + tk_3)))) \\
\Delta_r(\theta, \varphi) &= \theta\,(l_1 + t(l_2 + tl_3)) \cdot (i_1 c + i_2 s + i_3 c' + i_4 s') \\
\Delta_t(\theta, \varphi) &= \theta\,(m_1 + t(m_2 + tm_3)) \cdot (j_1 c + j_2 s + j_3 c' + j_4 s') \\
\boldsymbol{u}_r(\varphi) &= (c, s) \\
\boldsymbol{u}_\varphi(\varphi) &= (-s, c)
\end{aligned}$$

Formally, $(\cos(\varphi), \sin(\varphi)) = (r_x, r_y) / \operatorname{sqrt}(r_x^2 + r_y^2)$ but some care needs to be taken to avoid the singularity at $(r_x, r_y) = (0, 0)$.

Stereo rectification

If the images are stereo rectified before being given to the SDK, special care needs to be taken to ensure that the image plane rotations used in the rectification process are correctly applied to the extrinsic matrices.

Before rectification, the calibrated stereo-camera-IMU system consists of the extrinsic matrices $T_{\text{IMU}\to\text{Cam0}}$, $T_{\text{IMU}\to\text{Cam1}}$ and the camera models f_0, f_1.

The rectification operation outputs the image plane rotation matrices R_0, R_1 and a pinhole camera model f_{pin}. The pixels in the rectified images then obey the camera model

$$f_i' : \boldsymbol{r} \mapsto \boldsymbol{p}' = f_{\text{pin}}(R_i \cdot \boldsymbol{r}), \qquad i=0 \text{ or } i=1$$

It is then possible to define the undistortion function as

$$h_i : \boldsymbol{p}' \mapsto \boldsymbol{p} = f_i(\boldsymbol{r}) = f_i(R_i^{\mathsf{T}} \cdot f_{\text{pin}}^{-1}(\boldsymbol{p}')),$$

where the inverse projection can be defined as

$$f_{\text{pin}}^{-1} : (p_x, p_y) \mapsto ((p_x - c_x)f_x, (p_y - c_y)f_y, 1).$$

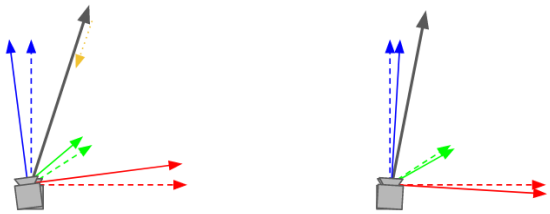
The rectified images I_i' are formed from the original image I_i as

$$I_i'(\boldsymbol{p}') = I_i(\boldsymbol{p}) = I(h_i(\boldsymbol{p}'))$$

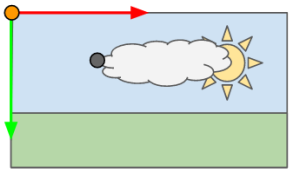
If the rectified images I' are used as input to the SDK, the calibration data should define:

- Undistorted pinhole camera model f_{pin} for both cameras
- IMU-to-camera matrices modified as $T_{\text{IMU}\to\text{Cam,i}}' = g(R_i) \cdot T_{\text{IMU}\to\text{Cam,i}}$

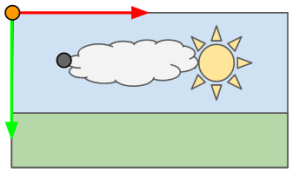
where g creates a 4x4 transformation matrix from a 3x3 rotation matrix as described in the figure below.



Original (dashed) vs rectified (solid) IMU-to-camera matrices



Left



Right

Rectification rotation

$R: \boldsymbol{r} \mapsto \boldsymbol{r'}$

Stereo rectification induces a rotation that is applied to the ray direction before the pinhole camera model f_{pin}.

The full model for the *rectified* pixel coordinates $\boldsymbol{p}$' is

$f': \boldsymbol{r} \mapsto \boldsymbol{p'} = f_{pin}(R \cdot \boldsymbol{r})$

if using the original IMU-to-cam. matrix and

$f_{rect}: \boldsymbol{r} \mapsto \boldsymbol{p'} = f_{pin}(\boldsymbol{r})$

with the rectified matrix T' = g(R) · T, where

$$g(R) = \begin{pmatrix} R & \\ & 1 \end{pmatrix}$$

Relationships between the extrinsic matrices

To combine an accurate stereo extrinsic matrix $T_{0\to1}$ with "as-designed" or otherwise approximate IMU-to-camera extrinsics, first determine the IMU-to-camera extrinsics for the left camera, $T_{0\to1} \cdot T_{\text{IMU}\to\text{Cam}}$, and then compute

$$T_{\text{IMU}\to\text{Cam},1} = T_{0\to1} \cdot T_{\text{IMU}\to\text{Cam},0}$$

where the dot symbol denotes matrix multiplication.

The following identities may also be useful in other similar scenarios (the superscript "-1" denotes matrix inversion):

- $T_{\text{IMU}\to\text{Cam},0} = T_{0\to1}^{-1} \cdot T_{\text{IMU}\to\text{Cam},1}$
- $T_{0\to1} = T_{\text{IMU}\to\text{Cam},1} \cdot T_{\text{IMU}\to\text{Cam},0}^{-1}$
- $T_{1\to0} = T_{0\to1}^{-1}$

.