Warning: A large error during verification can be due to miscalibrated cameras, an inaccurate calibration grid, or wrong grid width or height. Please make sure that the grid is accurate and the entered grid width and height are correct. Otherwise, manual calibration will actually decalibrate the cameras!

Step 3: Performing calibration

The camera's exposure time should be set appropriately before starting the calibration. To achieve good calibration results, the images should be well-exposed and image noise should be avoided. Thus, the maximum auto-exposure time should be great enough to achieve a very small gain factor, ideally 0.0 dB. The gain factor is displayed below the camera images as shown in Fig. 6.6.3.

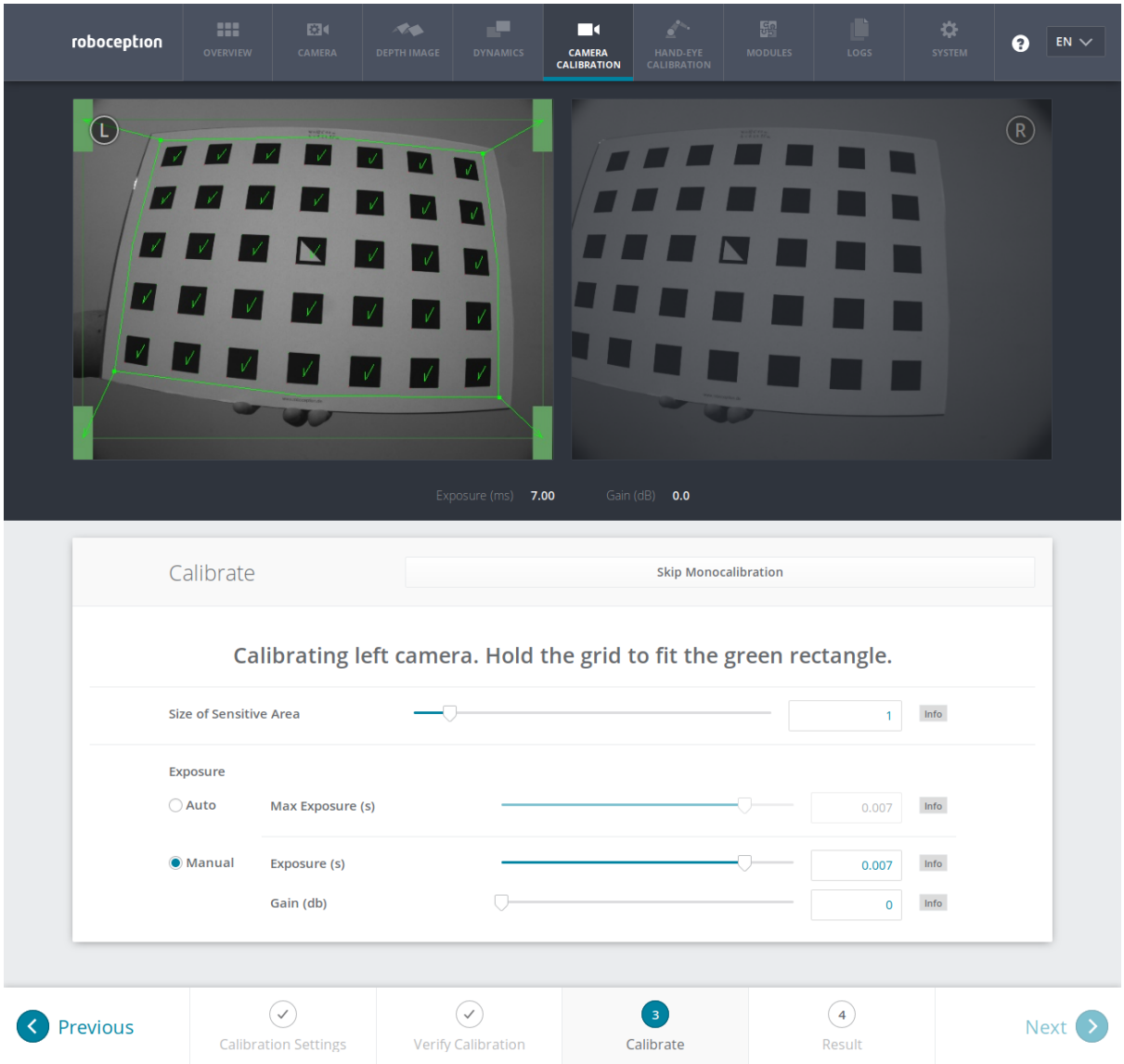

Fig. 6.6.3: Starting the calibration procedure

For calibration, the grid has to be held in certain poses. The arrows from the grid corners to the green areas indicate that all grid corners should be placed inside the green areas. The green areas are called sensitive areas. The *Size of Sensitive Area* slider can control their size to ease calibration as shown in the screen shot in Fig. 6.6.3. However, please be aware that increasing their size too much may result in slightly less calibration accuracy.

Holding the grid upside down is a common mistake made during calibration. Spotting this in this case is easy because the green lines from the grid corners into the green areas will cross each other as shown in Fig. 6.6.4.

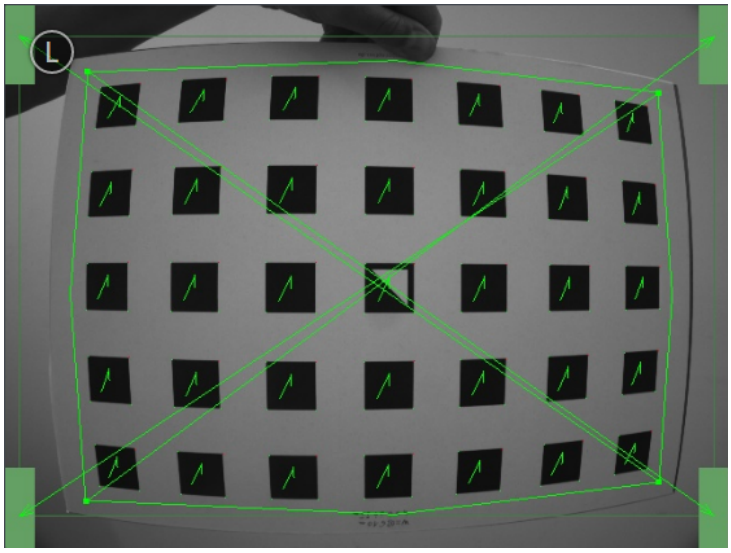

Fig. 6.6.4: Wrongly holding the grid upside down leads to crossed green lines.

Note: Calibration might appear cumbersome as it involves holding the grid in certain predefined poses. However, only this can ensure an unbiased, high-quality calibration result.

Monocalibration

Full calibration consists of calibrating each camera individually and then performing a stereo calibration to determine the relationship between them. In most cases, the intrinsic calibration of each camera does not get corrupted. For this reason, *Skip Monocalibration* in the *Calibrate* tab should be clicked to skip monocalibration during the first recalibration. Continue with the guidelines given in *Stereo calibration*. If stereo calibration yields an unsatisfactory calibration error, then calibration should be repeated without skipping monocalibration.

The monocalibration process involves five poses for each camera as shown in Fig. 6.6.5.

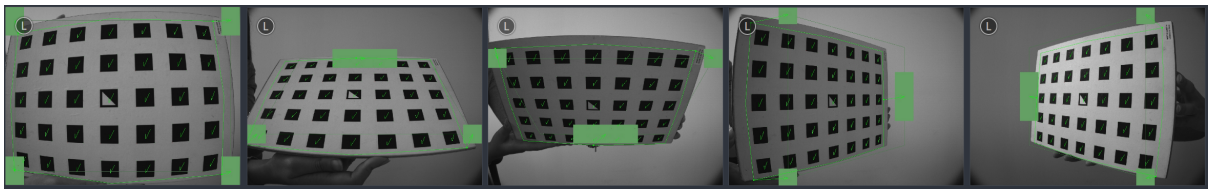

Fig. 6.6.5: Poses required for monocamera calibration

After the corners or sides of the grid are placed on top of the sensitive areas, the process automatically shows the next pose required. When the process is finished for the left camera, the same procedure is repeated for the right one.

Stereo calibration

After monocalibration is completed or has been skipped, the stereo calibration process is started. During stereo calibration, both cameras are calibrated to each other to find their relative rotation and translation.

First, the grid should be held closer than 40 cm from the sensor. It must be fully visible in both images and the cameras should look perpendicularly onto the grid. A green outline that stays in the image indicates the images' acceptance.

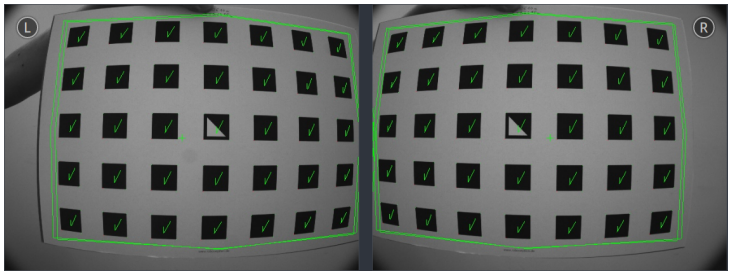

Fig. 6.6.6: Holding the grid closer than 40 cm during stereo calibration

Next, the grid should be held at least 1 m from the cameras. The small cross in the middle of the images should be inside of the grid and the cameras must look perpendicularly onto the grid. A green outline that stays in the image indicates the images' acceptance.

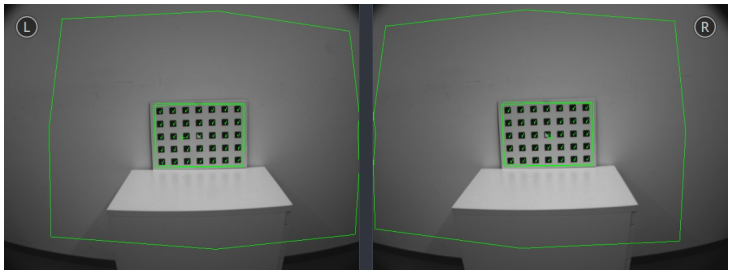

Fig. 6.6.7: Holding the grid farther away than 1 m during stereo calibration

Note: If the check marks on the calibration grid all vanish, then either the camera does not look perpendicularly onto the grid, the green cross in the middle of the images is not inside the grid, or the grid is too far away from the camera.

Step 4: Storing the calibration result

Clicking the *Compute Calibration* button finishes the process and displays the final result. The presented result is the mean reprojection error of all calibration points. It is given in pixels and typically has a value around 0.3.

Note: The given result is the minimum error left after calibration. The real error is definitely not less than this, but could in theory be larger. This is true for every camera-calibration algorithm and the reason why we enforce holding the grid in very specific poses. Doing so ensures that the real calibration error cannot significantly exceed the reported error.

Pressing *Save Calibration* applies the calibration and saves it to the sensor.

Warning: If a hand-eye calibration was stored on the *rc_visard* before camera calibration, the hand-eye calibration values could have become invalid. Please repeat the hand-eye calibration procedure.

6.6.3 Parameters

The component is called `rc_stereocalib` in the REST-API.

Note: The camera calibration component's available parameters and status values are for internal use only and may change in the future without further notice. Calibration should only be performed through the Web GUI as described above.

6.6.4 Services

Note: The camera calibration component's available service calls are for internal use only and may change in the future without further notice. Calibration should only be performed through the Web GUI as described above.

6.7 Hand-eye calibration

For applications, in which the *rc_visard* is integrated into one or more robot systems, it needs to be calibrated w.r.t. some robot reference frames. For this purpose, the *rc_visard* is shipped with an on-board calibration routine called the *hand-eye calibration* component.

Note: The implemented calibration routine is completely agnostic about the user-defined robot frame to which the *rc_visard* is calibrated. It might be a robot's end-effector (e.g., flange or tool center point) or any point on the robot structure. The method's only requirement is that the pose (i.e., translation and rotation) of this robot frame w.r.t. a user-defined external reference frame (e.g., world or robot mounting point) is exactly observable by the robot controller and can be reported to the calibration component.

The *Calibration routine* (Section 6.7.3) itself is an easy-to-use three-step procedure using a calibration grid. Calibration grids for the *rc_visard* can be obtained from Roboception.

6.7.1 Calibration interfaces

The following two interfaces are offered to conduct hand-eye calibration:

1. All services and parameters of this component required to conduct the hand-eye calibration **programmatically** are exposed by the *rc_visard*'s *REST-API interface* (Section 8.2). The respective node name of this component is `rc_hand_eye_calibration` and the respective service calls are documented *Services* (Section 6.7.5).

   Note: The described approach requires a network connection between the *rc_visard* and the robot controller to pass robot poses from the controller to the sensor's calibration component.

2. For use cases where robot poses cannot be passed programmatically to the *rc_visard*'s hand-eye calibration component, the *Web GUI*'s *Hand-Eye Calibration* tab (Section 4.5) offers a guided process to conduct the calibration routine **manually**.

   Note: During the process, the described approach requires the user to manually enter into the Web GUI robot poses, which need to be accessed from the respective robot-teaching device or handheld.

6.7.2 Sensor mounting

As illustrated in Fig. 6.7.1 and Fig. 6.7.2, two different use cases w.r.t. to the mounting of the *rc_visard* generally have to be considered:

a. The *rc_visard* is **mounted on the robot**, i.e., it is mechanically connected at its *mounting points* (Section 3.6) to a robot link (e.g., at its flange or a flange-mounted tool), and hence moves with the robot.

b. The *rc_visard* is not mounted on the robot but is fixed to a table or other place in the robot's vicinity and remains at a **static** position w.r.t. the robot.

While the general *Calibration routine* (Section 6.7.3) is very similar in both use cases, the calibration process's output, i.e., the resulting calibration transform, will be semantically different, and the fixture of the calibration grid will also differ.

Calibration with a robot-mounted sensor When calibrating a robot-mounted *rc_visard* with the robot, the calibration grid has to be secured in a static position w.r.t. the robot, e.g., on a table or some other fixed-base coordinate system as sketched in Fig. 6.7.1.

> **Warning:** It is extremely important that the calibration grid does not move during step 2 of the *Calibration routine* (Section 6.7.3). Securely fixing its position to prevent unintended movements such as those caused by vibrations, moving cables, or the like is therefore strongly recommended.

The result of the calibration (step 3 of the *Calibration routine*, Section 6.7.3) is a pose $\mathbf{T}_{\text{camera}}^{\text{robot}}$ describing the (previoulsy unknown) relative positional and rotational transformation between the *rc_visard*'s *camera* frame and the user-selected *robot* frame such that

$$\mathbf{p}_{\text{robot}} = \mathbf{R}\left(\mathbf{T}_{\text{camera}}^{\text{robot}}\right) \cdot \mathbf{p}_{\text{camera}} + \mathbf{t}\left(\mathbf{T}_{\text{camera}}^{\text{robot}}\right) \ , \tag{6.7.1}$$

where $\mathbf{p}_{\text{robot}} = (x, y, z)^T$ is a 3D-point with its coordinates expressed in the *robot* frame, $\mathbf{p}_{\text{camera}}$ is the same point represented in the *camera* coordinate frame, and $\mathbf{R}(\mathbf{T})$ as well as $\mathbf{t}(\mathbf{T})$ are the corresponding 3×3 rotation matrix and 3×1 translation vector to a pose $\mathbf{T}$, respectively.

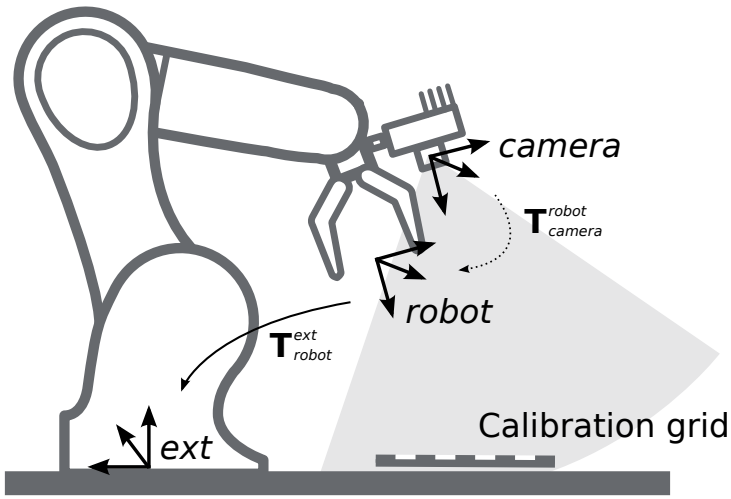

Fig. 6.7.1: Important frames and transformations for calibrating a robot-mounted *rc_visard*: The sensor is mounted with a fixed relative position to a user-defined *robot* frame (e.g., flange or TCP). It is important that the pose $\mathbf{T}_{\text{robot}}^{\text{ext}}$ of this *robot* frame w.r.t. a user-defined external reference frame *ext* is observable during the calibration routine. The result of the calibration process is the desired calibration transformation $\mathbf{T}_{\text{camera}}^{\text{robot}}$, i.e., the pose of the *camera* frame within the user-defined *robot* frame.

Calibration with a statically-mounted sensor In use cases where the *rc_visard* is positioned statically w.r.t. the robot, the calibration grid needs to be mounted to the robot as shown for example in Fig. 6.7.2 and Fig. 6.7.3.

> **Note:** The hand-eye calibration component is completely agnostic about the exact mounting and positioning of the calibration grid w.r.t. the user-defined *robot* frame. That is, the relative positioning of the calibration grid to that frame neither needs to be known, nor it is relevant for the calibration routine, as shown in Fig. 6.7.3.

Warning: It is extremely important that the calibration grid is attached securely to the robot such that it does not change its relative position w.r.t. the user-defined *robot* frame during step 2 of the *Calibration routine* (Section 6.7.3).

Securely preventing unintended position changes such as those caused by vibrations, for example by mounting the calibration grid itself on a wooden support (suggested thickness min. 1 cm), which can then be screwed to the robot structure, e.g., its flange or tool, is therefore strongly recommended.

In this use case, the result of the calibration (step 3 of the *Calibration routine*, Section 6.7.3) is the pose $\mathbf{T}^{\text{ext}}_{\text{camera}}$ describing the (previoulsy unknown) relative positional and rotational transformation between the *rc_visard*'s *camera* frame and the user-selected external reference frame *ext* such that

$$\mathbf{p}_{\text{ext}} = \mathbf{R}\left(\mathbf{T}^{\text{ext}}_{\text{camera}}\right) \cdot \mathbf{p}_{\text{camera}} + \mathbf{t}\left(\mathbf{T}^{\text{ext}}_{\text{camera}}\right) \ , \tag{6.7.2}$$

where $\mathbf{p}_{\text{ext}} = (x, y, z)^T$ is a 3D point with its coordinates expressed in the external reference frame *ext*, $\mathbf{p}_{\text{camera}}$ is the same point represented in the *camera* coordinate frame, and $\mathbf{R}(\mathbf{T})$ as well as $\mathbf{t}(\mathbf{T})$ are the corresponding 3×3 rotation matrix and 3×1 translation vector to a pose $\mathbf{T}$, respectively.

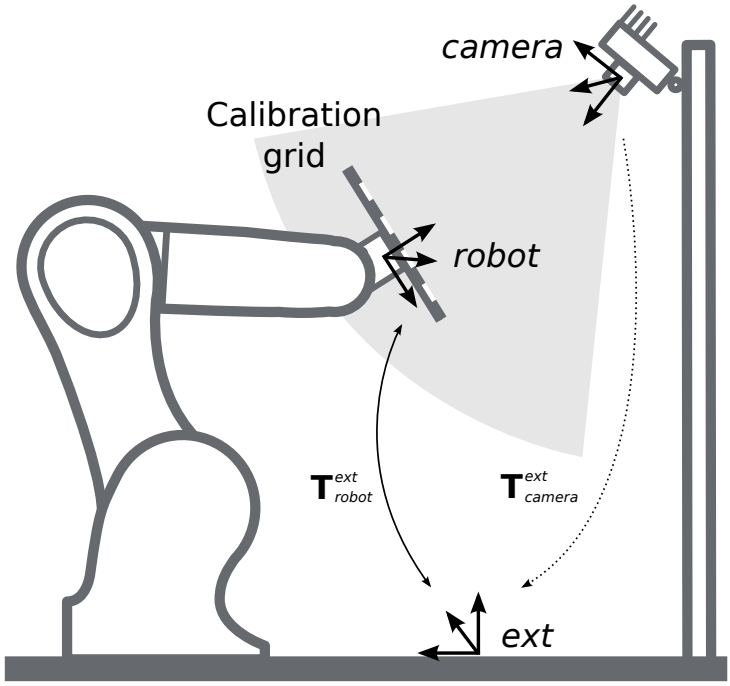

Fig. 6.7.2: Important frames and transformations for calibrating a statically mounted *rc_visard*: The sensor is mounted with a fixed position relative to a user-defined external reference frame *ext* (e.g., the world coordinate frame or the robot's mounting point). It is important that the pose $\mathbf{T}^{\text{ext}}_{\text{robot}}$ of the user-defined *robot* frame w.r.t. this frame is observable during the calibration routine. The result of the calibration process is the desired calibration transformation $\mathbf{T}^{\text{ext}}_{\text{camera}}$, i.e., the pose of the *camera* frame in the user-defined external reference frame *ext*.

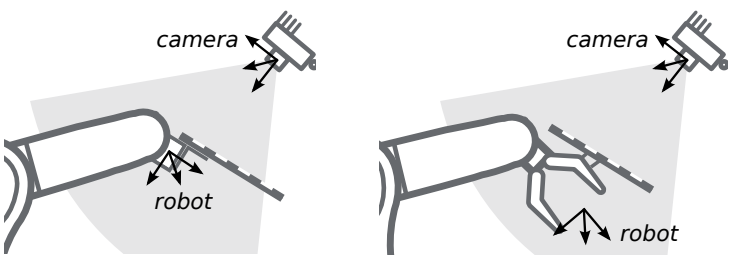

Fig. 6.7.3: Alternate mounting options for attaching the calibration grid to the robot

6.7.3 Calibration routine

The general hand-eye calibration routine consists of three steps, which are illustrated in Fig. 6.7.4. These three steps are also represented in the *Web GUI*'s guided hand-eye-calibration process (Section 4.5).

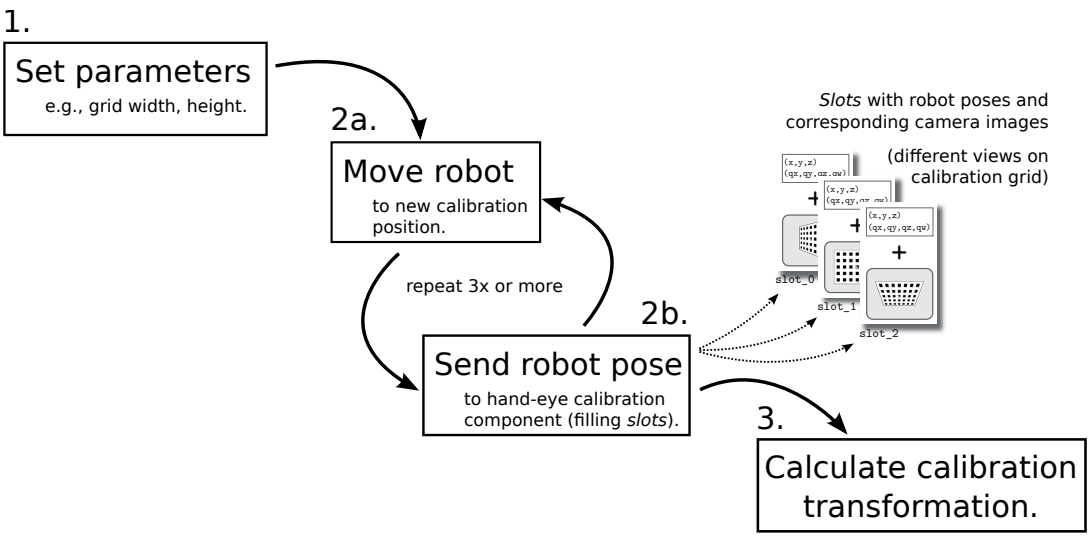

Fig. 6.7.4: Illustration of the three different steps involved in the hand-eye calibration routine

Step 1: Setting parameters

Before starting the actual calibration routine, the grid size and sensor-mounting parameters have to be set to the component. As for the REST-API, the respective parameters are listed in *Parameters* (Section 6.7.4).

Web GUI example: The Web GUI offers an interface for entering these parameters during the first step of the calibration routine as shown in Fig. 6.7.5. In addition to grid size and sensor mounting, the Web GUI also offers a *Pose* setting to be defined by the user. It specifies the format used for reporting the robot poses in the upcoming step 2 of the calibration process, either as *XYZABC* for positions and Euler angles, or *XYZ+quaternion* for positions plus quaternions for representing rotations. See *Pose formats* (Section 13.1) for the exact definitions.

> **Note:** The *Pose* parameter is added to the Web GUI as a convenience option only. For reporting poses programmatically via REST-API, the *XYZ+quaternion* format is mandatory.

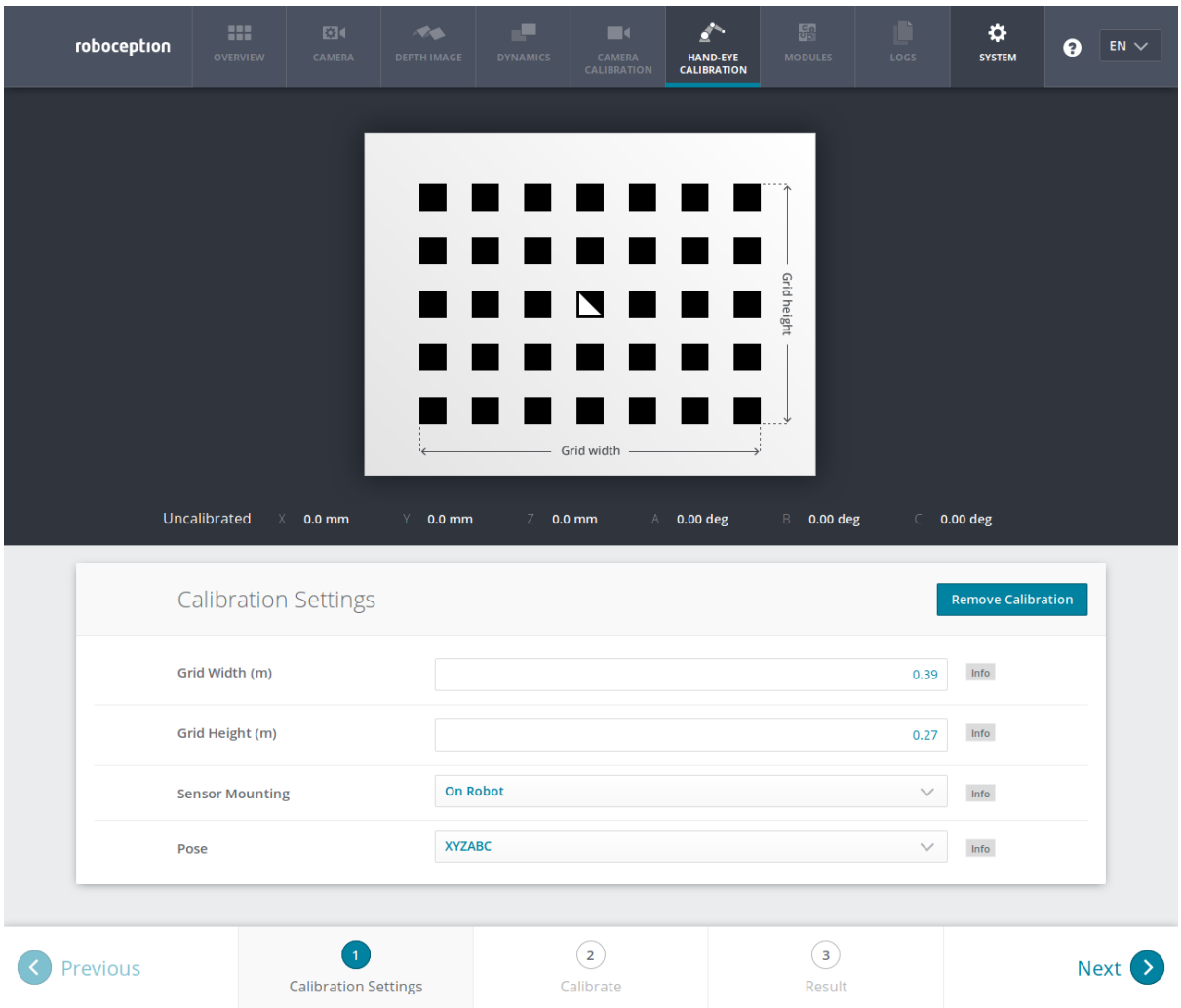

Fig. 6.7.5: Defining hand-eye calibration settings via the *rc_visard*'s Web GUI

Step 2: Selecting and reporting robot calibration positions

In this step (2a.), the user defines several calibration positions for the robot to approach. These positions must each ensure that the calibration grid is completely visible in *rc_visard*'s left camera image. Furthermore, the robot positions need to be selected properly to achieve a variety of different perspectives for the *rc_visard* to perceive the calibration grid. Fig. 6.7.6 shows a schematic recommendation of four different view points.

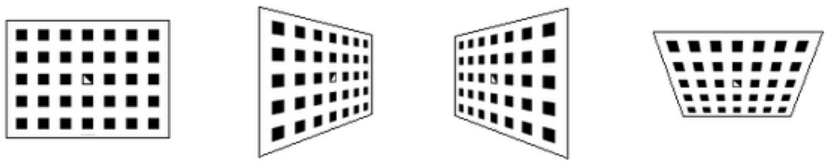

Fig. 6.7.6: Recommended views on the calibration grid during the calibration procedure

Warning: Calibration quality, i.e., the accuracy of the calculated calibration result, depends on the calibration-grid views provided. The more diverse the perspectives are, the better is the calibration. Choosing very similar views, i.e., varying the robot positions only slightly between different repetitions of step 2a., may lead to inaccurate estimation of the desired calibration transformation.

After the robot reaches each calibration position, the corresponding pose $\mathbf{T}^{\text{ext}}_{\text{robot}}$ of the user-defined *robot* frame in the user-defined external reference frame *ext* needs to be reported to the hand-eye calibration component (2b.). For this purpose, the component offers different *slots* to store the reported poses and the *rc_visard*'s corresponding left camera images. All filled slots will then be used to calculate the desired calibration transformation between the *rc_visard*'s *camera* frame and either the user-defined *robot* frame (robot-mounted sensor) or the user-defined external reference frame *ext* (static sensor).

Note: To successfully calculate the hand-eye calibration transformation, at least three different robot calibration poses need to be reported and stored in slots. However, to prevent errors induced by possible inaccurate measurements, at least **four calibration poses are recommended**.

To transmit the poses programmatically, the component's REST-API offers the `set_pose` service call (see *Services*, Section 6.7.5).

Web GUI example: After completing the calibration settings in step 1 and clicking *Next*, the Web GUI offers four different slots (*First View*, *Second View*, etc.) for the user to fill manually with robot poses. At the very top, a live stream from the camera is shown indicating whether the calibration grid is currently detected or not. Next to each slot, a figure suggests a respective dedicated viewpoint on the grid. For each slot, the robot must be operated to achieve the suggested view.

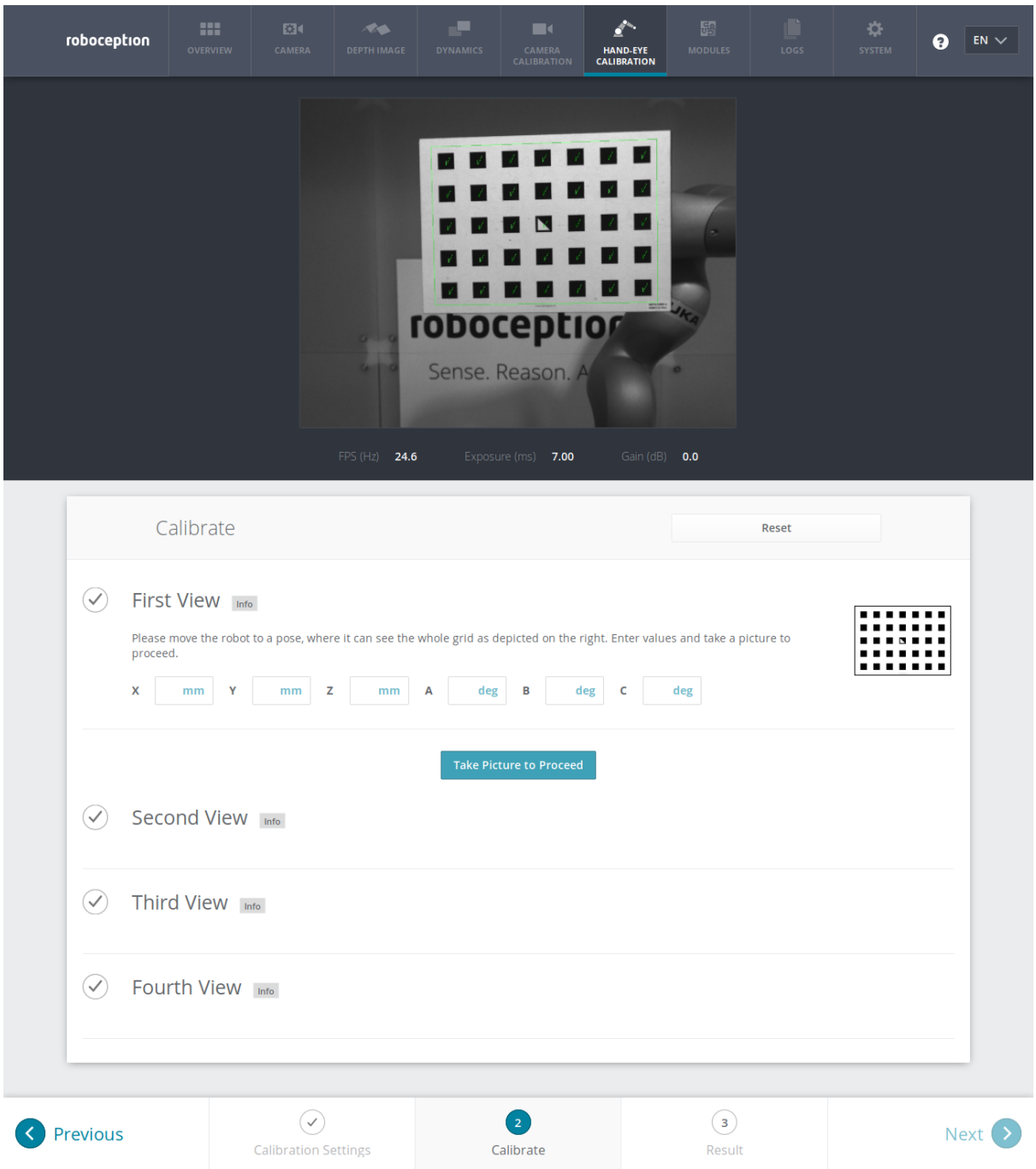

Fig. 6.7.7: First sample image in the hand-eye calibration process for a statically mounted *rc_visard*

Once the suggested view is achieved, the user-defined *robot* frame's pose needs to be entered manually into the respective text fields, and the corresponding camera image is captured using the *Take Picture to Proceed* button.

Note: The user's acquisition of robot pose data depends on the robot model and manufacturer – it might be read from a teaching or handheld device, which is shipped with the robot.

Warning: Please be careful to correctly and accurately enter the values; even small variations or typos may lead to calibration-process failure.

This procedure is repeated four times in total. Complying to the suggestions to observe the grid from above, left, front, and right, as sketched in Fig. 6.7.6, in this example the following corresponding camera images have been sent to the hand-eye calibration component with their associated robot pose:

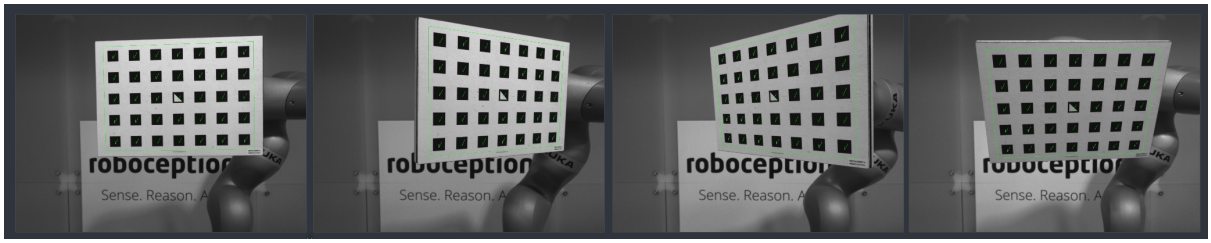

Fig. 6.7.8: Recorded camera images as input for the calibration procedure

Step 3: Calculating and saving the calibration transformation

The final step in the hand-eye calibration routine consists of issuing the desired calibration transformation to be computed from the collected poses and camera images. The REST-API offers this functionality via the `calibrate` service call (see *Services*, Section 6.7.5). Depending on the way the *rc_visard* is mounted, this service computes and returns the transformation (i.e., the pose) between the *camera* frame and either the user-defined *robot* frame (robot-mounted sensor) or the user-defined external reference frame *ext* (statically mounted sensor); see *Sensor mounting* (Section 6.7.2).

To enable users to judge the quality of the resulting calibration transformation, the component also reports a calibration error. This value is measured in pixels and denotes the root mean square of the *reprojection error* averaged over all calibration slots and all corners of the calibration grid. However for a more intuitive understanding, this measurement might be normalized by utilizing *rc_visard*'s focal length f in pixels:

$$E = \frac{E_{\text{camera}}}{f} \; .$$

> **Note:** The *rc_visard* reports a focal length factor via its various interfaces. It relates to the image width for supporting different image resolutions. The focal length f in pixels can be easily obtained by multiplying the focal length factor by the image width in pixels.

The value E can now be interpreted as an object-related error in meters in the 3D-world. Given that the distance between the calibration grid and the *rc_visard* is one meter, the *average* accuracy associated with transforming the grid's coordinates from the *camera* frame to the target frame is $1 \cdot E$ m; assuming a distance of 0.5 meters, it measures $0.5 \cdot E$ m, etc.

Web GUI example: The Web GUI automatically triggers computation of the calibration result immediately after taking the last of the four pictures. The user just needs to click the *Next* button to proceed to the result. In this example with a statically mounted *rc_visard*, the resulting output is the pose of the sensor's left camera in the world coordinate system of the robot – represented in the *pose* format as specified in step 1 of the calibration routine.

The reported error of $E_{\text{camera}} = 0.4$ pixels in Fig. 6.7.9 transforms into a calibration accuracy of $E = \frac{E_{\text{camera}}}{f} \approx \frac{0.4}{1081.46} \approx 0.00036$, which is 0.36 mm at 1 meter distance – a submillimeter accuracy for this calibration run.