Calibration

In order to calculate gaze data with high accuracy and precision, the eye tracker firmware needs to adapt the algorithms to the person sitting in front of the tracker. This adaptation is done during the calibration process when the user is looking at points located at known coordinates. The calibration is initiated and controlled by the client application.

The calibration procedure

The calibration of the eye tracker would typically be done as follows:

1. A small animated object is displayed, to catch the user's attention.
2. When it arrives at the calibration point location, the object rests for about 0.5 seconds to give the user a chance to focus on it. Good practice is to shrink the object to help the user focus the gaze on its center, i.e. the calibration point location.
3. When the user has focused his or her gaze on the calibration point, the eye tracker is told to start collecting data for that specific calibration point.
4. The eye tracker collects data for the calibration point and sends a notification to the client application when the data collection is completed.
5. If the object has been shrunk, it is now returned to its original size.
6. The object is moved to the next calibration point location, usually by using an animation.
7. Repeat steps 2-6 for all desired calibration points.
8. The calibration result is computed and reviewed in a calibration plot. (For screen based eye trackers only)
9. If the data for one or more calibration point is missing or with low accuracy or precision, steps 2-6 and 8 are repeated for those points.

10. Once the calibration result is satisfactory, the calibration procedure is concluded.
11. (Optional) After the participant is calibrated, perform a validation of the estimated performance. Read more about Calibration Validation below.

The animation in step 1 should not be too fast, nor should the shrinking in step 2 be too fast. Otherwise the user may not be able to get a good calibration result due to the fact that he or she has no time to focus the gaze on the target before the eye tracker starts collecting calibration data.

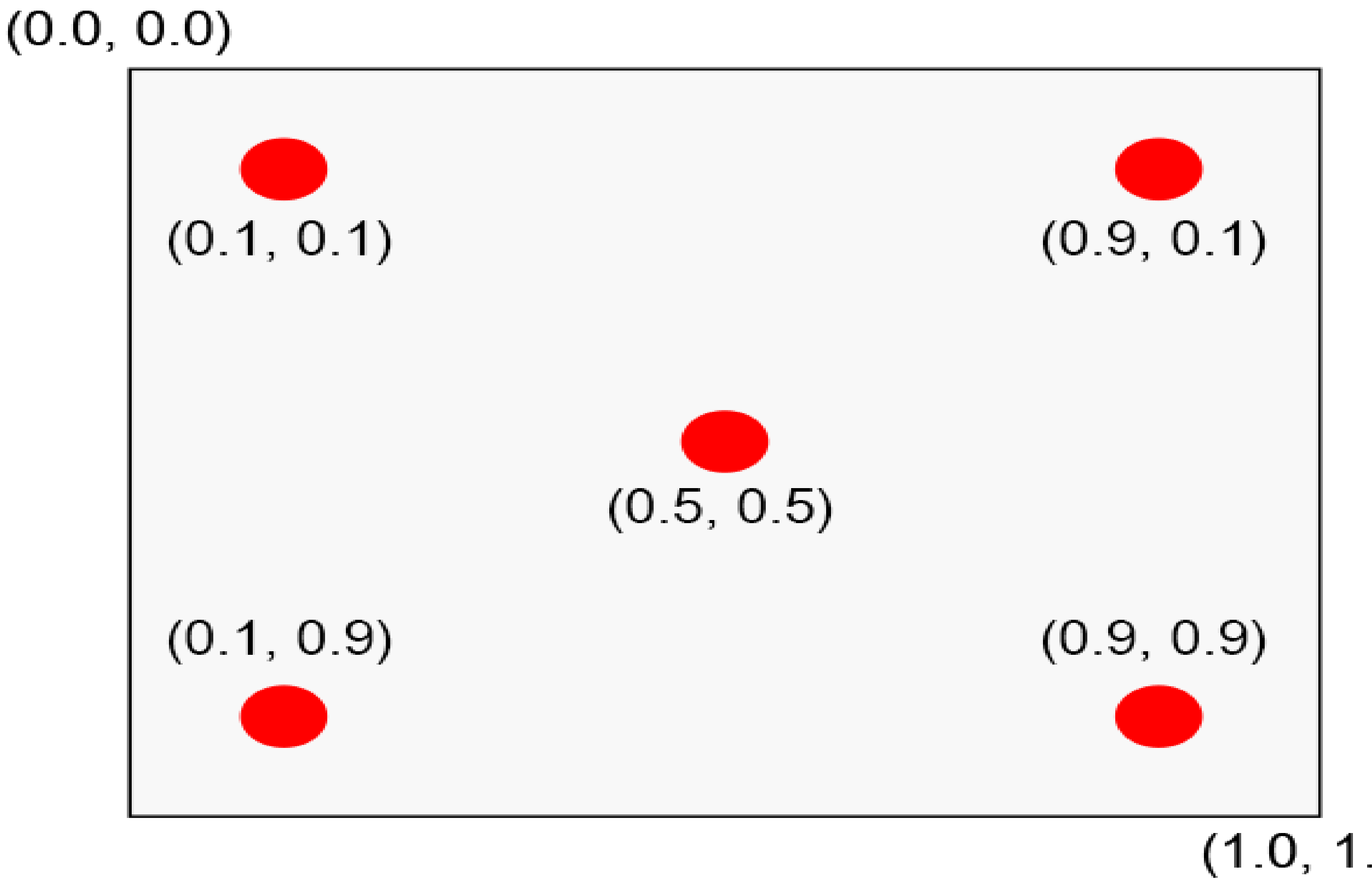
The normal number of calibration points is 2, 5, or 9. More points (up to 14 point for ET5) can be used but the calibration quality will not increase significantly for more than 9 points. Usually 5 points yields a very good result and is not experienced as too intrusive by the user.

Configuring the calibration points

Screen based calibration

For screen based eye trackers the calibration points are given in normalized screen coordinates (0,0) is in the top left corner, (1,1) is in the lower right corner. It is the responsibility of the SDK client to display a marker at the corresponding coordinate on screen.

The location of the calibration points is decided by the client application. A typical calibration pattern for 5 points can be seen below. The coordinates illustrates common locations of the calibration points as expressed in the Active Display Coordinate System.



NOTE: All points are given in normalized coordinates in such a way that (0.0, 0.0) corresponds to the upper left corner and (1.0, 1.0) corresponds to the lower right corner of the screen. When choosing the calibration points it is important to consider the following:

- The calibration points should span an area that is as large or larger than the area where the gaze is of interest in order to ensure good data.
- The calibration points must be positioned within the area that is trackable by the eye tracker and be within the Active Display Area.

The calibration state

To be able to perform a calibration the client application must first enter the calibration state. The calibration state is an exclusive state which can only be held by one client at a time. It is entered by calling `tobii_calibration_start` function and is left by calling the `tobii_calibration_stop` function. Whenever a client enters or leaves the calibration state, a `TOBII_NOTIFICATION_TYPE_CALIBRATION_STATE_CHANGED` notification is sent to all clients connected to the same eye tracker, including the calibrating client. These notifications are mostly meant for user interface purposes, like graying out a "Calibrate" button etc. Since the communication with the eye tracker is asynchronous, it is considered to be best practice to use the `tobii_calibration_start` function to check whether it reports that another client is currently calibrating before initiating the calibration procedure rather than caching the result of the calibration events.

Some operations can only be performed when in the calibration state, e.g. `tobii_calibration_collect_data_2d`, `tobii_calibration_discard_data_2d` as well as `tobii_calibration_compute_and_apply`. Other operations such as `tobii_calibration_apply` or `tobii_calibration_retrieve` can be used at any time. However, if the eye tracker is in calibration mode, only the client who set it in that mode can apply a calibration on it.

Applying a calibration

The `tobii_calibration_compute_and_apply` function should be called once all calibration points have been shown and data collected. It uses the collected data to calculate an eye model based on the person in front of the eye tracker. The calibration can be recalculated with new input from calibration points until the calibration state is left.

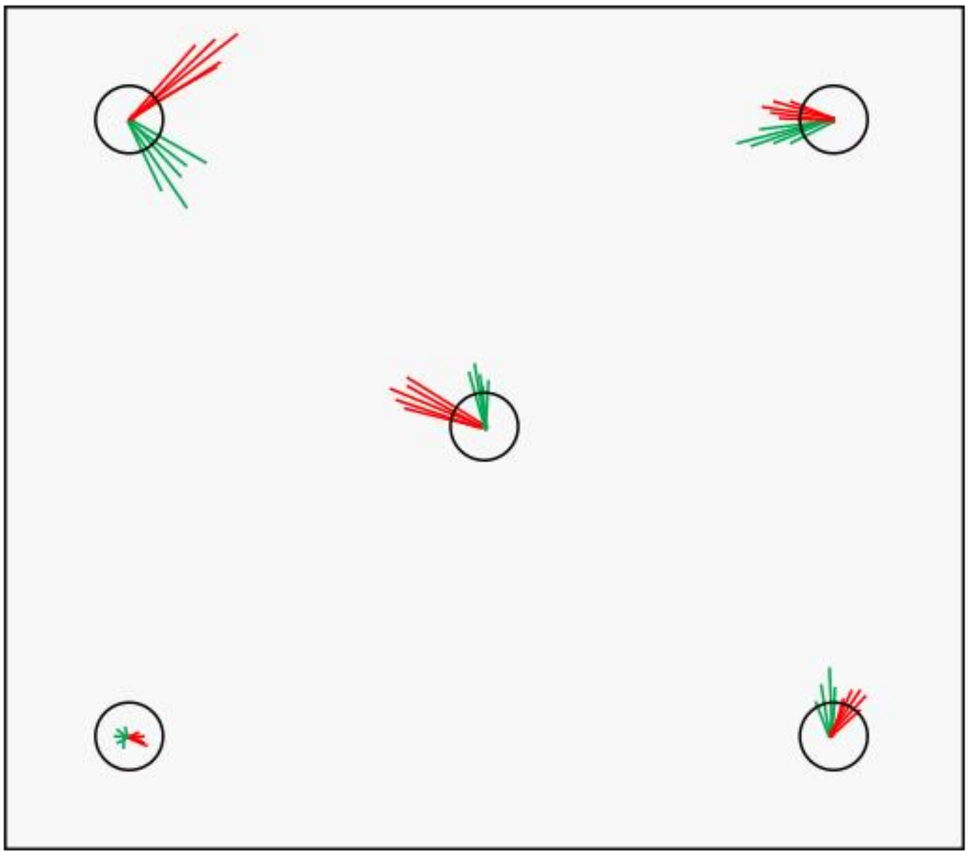
It is possible to save a calibration for a person locally and reapply it at a later time. This is useful if the same person will be using the eye tracker again as you then don't have to go through the entire calibration procedure each time. To get an already active and applied calibration from the eye tracker, call the `tobii_calibration_retrieve` function. To apply a saved calibration, call to the `tobii_calibration_apply` function.

NOTE: Before a calibration has been completed successfully, gaze data is already available from the eye tracker. However, the mapping of the gaze data can then be based either on a default eye model or a previous calibration depending on eye tracker model. Hence, this data should only be used as an indication of where a person is looking or for eye position in the track box.

The table below gives the name in the different languages for the concepts introduced and highlighted in italics in the text above.

Calibration plots

If you have previous experiences with any of Tobii's eyetracking products it is very likely that you have seen a calibration plot which is supposed to illustrate the calibration results. The calibration plot is a simple yet concise representation of a performed calibration and it usually looks something like what is shown below. However, the presentation design can vary.



The calibration plot shows the offset between the mapped gaze samples and the calibration points based on the best possible adaptation of the eye model to the collected values done by the eye tracker during calibration. In the image above, the red and green lines represent the offset between the mapped sample points (red for left eye and green for right eye) and the center of where the calibration points were shown. The circles are the actual calibration points. The data displayed in the plot is made available to client applications through the `tobii_calibration_parse` function which, returns contains a collection of `tobii_calibration_point_data_t` which in turn contains coordinates for where the calibration point was displayed and where the collected gaze points were estimated after calibration. This allows for implementation of alternative visualizations of calibration results as well as the traditional visualization as seen above. Keep in mind that actual accuracy/precision can differ from the results since the result is based on the same data that was used to optimize the calibration parameters.

Calibration validation

When performing an eye tracking study, it can be very useful to perform a validation of the estimated performance after the calibration. This step is often referred to as *Calibration validation*. The common procedure of doing a *Calibration validation* is to show a new set of stimuli points for the participant, collect calibrated gaze data during the stimuli presentation, and calculate values for accuracy and precision based on the gaze data's position in relation to the stimuli point (which the participant was expected to look at).