Look! Blickschulungsbrille
Technical Documentation

Look! ET UG (haftungsbeschränkt)

September 12, 2022

Contents

1 Introduction

This document is intended for professional and scientific users. It gives insights into technical details of the measurement process, programming APIs and data formats. It is not a required read for user intending to record data only.

The *Look! Blickschulungsbrille* is a mobile, head-mounted eye-tracking device. It is powered by modern machine vision methods that perform appearance based gaze estimation (i.e., there is no explicit pupil detection performed nor necessary, but the image of the eye as a whole is utilized). The glasses can be worn in combination with habitually worn eyewear and measures even through complex lenses.

Figure 1: Our Co-founder Elena wearing the Look! Blickschulungsbrille

For real-world studies in challenging conditions, i.e., when pupil detection is difficult (due to eyeglasses, make-up, irregularly shaped pupil or iris, reflections, and so on), neural networks clearly outperform the traditional pupil and glint detection approach.

The *Look! Blickschulungsbrille* consists of three miniature USB cameras. We call the two cameras that produce an image of the left and right eye *eye cams* and the camera that points towards the scenery the wearer is facing the *scene cam*. The scene cam records what is happening and the eye cams help us determine where in the scene the wearer is looking at. They are mounted on a 3D printed frame which also houses near-infrared filters combined with

eye-safe near-infrared illumination. The whole device works via a single USB2.0 plug. You can add a presenter stick to the recording device to enable remote control of the calibration and recording process.

Optionally, we offer a recording unit that can run the Look! Software on a small footprint embedded system and offers a command interface as well as a video live stream via Wifi to any connected web browser (such as a Laptop, Tablet or Smartphone of your choice).

Scene cam	
Resolution	640x480 px
Field of view wide (default)	120x70 °
Field of view pinhole	57x35 °
Frame rate	30 fps

Eye cam	
Resolution	320x240 px
Frame rate	30 fps

The scene cam records a cyclops view from a central perspective located over the wearer's nose. By default it comes with a wide angle lens, but smaller field of view are available. Eye cameras operate at a resolution of 320×240 pixels (they can go 640×480, if required for some reason). For research use-cases the camera firmware (of both scene and eye cameras) can be upgraded to operate at 60 Hz, provided a device with sufficient real-time processing capabilities is connected to handle to additional load (most modern CPUs can do so easily).

2 Gaze estimation

At the core of the Look! software is a neural network that predicts the geometric orientation of both eyeballs. We utilize `Mobilenet v3` building blocks to run them on CPU with minimal latency and CPU load. Eye features are extracted at multiple scales and contribute to an eye model that is latently learned by the neural network. That means we do not need to specify a corna or eyeball site explicitly, but the model learns to make these assumptions within the neural network.

It processes the eye images at a downsized resolution of 180×135 or 224x224 pixels. Depending on the variant of the network configured, we process:

- Both eyes jointly as a two-channel image.

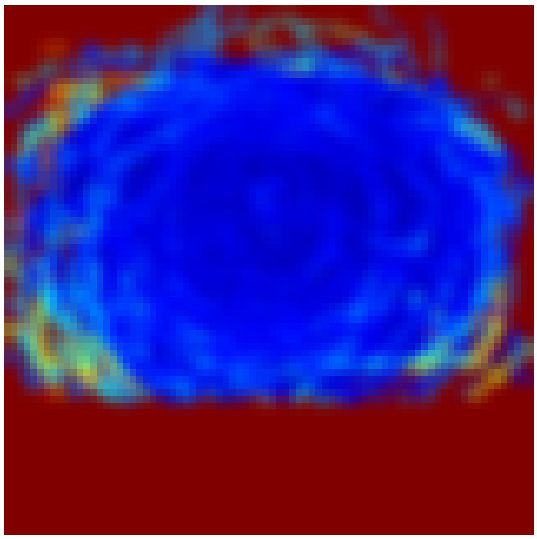


Figure 2: Gaze error distribution in the scene cam's field of view. Gaze estimation is stable within the central area and decreases towards the periphery, most noticeably towards the edges.

- Both eyes separately with the same network weights and a mirrored right eye.

The neural network is executed via OpenVINO (`https://openvino.ai`) on CPU, or as a tensorflow-lite model in case of the recording unit (optimized and quantized for its ARM processor). The Look! recording unit runs a quantized tensorflow-lite model, optionally on a Coral tensor processing unit, if available (`https://coral.ai`). That model is, besides optimizations for the specific architecture, identical to the Desktop version and produces very similar results.

The glasses can also run traditional pupil detection, and research institutions can directly access the state-of-the-art algorithms implemented in EyeRecToo (Santini et al. 2017), such as *Purest* pupil detection and tracking as well as the *Swirski* temporal eye modeling by ellipse shape.

Accuracy of the uncalibrated eye-tracker is 5.6° on average over the whole field of view. After calibration accuracies $<1°$ can usually be expected. Actual accuracy depends on tracking conditions and the quality of the calibration performed. Down to 0.47° are achievable given ideal conditions (but unlikely during actual in-field usage).

2.1 Calibration

2.1.1 Calibration Markers

The Look! software detects both Aruco markers (as in EyeRecToo) and concentric markers (see Figure 3). We recommend the use of concentric markers as it is easier to explain to subjects where to look at. Also, the marker is considerably smaller and can be detected very reliably. A calibration can be started by clicking the calibrate button or long pressing one of the remote control buttons. The calibration procedure needs to be manually finished the same way. As marker detection is an optical process, make sure the lighting is sufficient or turn on the lights during the calibration procedure.

Figure 3: Concentric calibration marker: Two high contrast concentric circles on a credit card sized marker.

2.1.2 Fitting methods

In hybrid calibration mode (default), Look! will estimate a 2D pixel offset (x- and y-direction) to the gaze pixel coordinates predicted by the neural network. This simple calibration is indicated in the GUI by a yellow calibration marker visualized on top of the detected calibration marker position. Once the indicator turns green (after some seconds of collecting calibration data), an extensive calibration is performed: coefficients of a polynomial of degree 2 are determined via Ridge regression.

While the calibration is running, the calibration function will continuously update itself. It is recommended to calibrate until the marker detection and gaze estimation indicators overlap for all desired gaze directions. Look! has some built-in feature weighting to balance calibration samples for different gaze directions, so there is no need to gaze in each direction for the same amount of time.

Calibration be be applied either on the predicted gaze coordinates, on the predicted per eye orientation vector, or on the slippage robust eye orientation calculated as *instantaneous gaze* via EyeRecToo.

2.2 Caveats

Note that the field of view of our scene camera is very large. It might not be necessary nor feasible to calibrate the complete image. However, gaze estimation quality deteriorates dramatically when leaving the calibrated area. It is thus important to calibrate at least that field of view which is expected to be relevant for the task.

3 Recommended metrics

With the given resolution and sampling rate, we recommend the use of fixation based metrics (dwell time, dwell on AOI, gaze on target,...). It is possible to use time-based metrics (first glance on AOI, time to first fixation).

It is not possible to calculate saccadic velocity profiles (such as saccadic peak velocity, acceleration, and such).

4 Frame

The frame is made of bio-compatible PA12 Nylon. Its flexibility allows fitting many different head shapes. Therefore, it presses lightly against the side of the head, fixing itself in position and avoiding slippage. The area where the pressure is produced (near the ears) is thickened to distribute the force on a larger area and make wearing more comfortable. When worn correctly, the eye cameras should be at the bottom of the field of view of the wearer and not obstruct the important central field of vision. If this is not the case, the position of the frame on the nose or the height of the nose piece can be adjusted.

Where the earpieces meet the front piece (where regular glasses have a joint), stability is of special importance to avoid that the frame bends

diagonally. This would lead to a skewed positioning of the glasses on the nose and an oddly looking fit (after calibration, the glasses are still functional though). When adapting the frame, this area should therefore be handled with special care (e.g., no holes drilled there).

Two NIR-LEDs (QBLP630-IR3) are placed on each eye camera. The LEDs are connected to pins on the camera chip for power supply, so no extensive wiring is required. The NIR illumination used is comparatively weak - in fact often much weaker than natural sunlight. That is because we are working with methods that do not require a glint to be visible. The illumination is only required for situations in which ambient illumination is dim. For accurate eye modelling based on glints it might be necessary to add more and stronger LEDs.

4.1 Safety considerations

Look! Blickschulungsbrille bears the CE mark and Look! ET UG (haftungsbeschränkt) declares conformity with the relevant regulations:

1. NIR illumination is classified as safe for continuous exposure following IEC/EN 62471: 2008 Photobiological safety of lams and lamp systems. The maximum intensity of the LEDs is stated as 1.6 mW/sR. We advise to keep a distance of at least 1 cm between LEDs and eye. Undamaged devices ensure an even larger distance through their geometry. Please do not use damaged devices that may allow the LEDs to get closer to the eye than intended. Broken camera arms can be repaired professionally by us!

2. RoHS2 Directive 2002/95/EC

3. Electronics passed EMC/LVD tests following EN62368-1:2014+A11:2017

4. PA12 Nylon is safe for skin contact.

Look! Recording unit bears the CE mark, documentation on request.

5 API

Access to the API is required in case you intend to remote control the Look! recording software or mobile recording unit. It is also required to achieve live gaze data access. It is not required for normal operation (i.e., recording a video and gaze data). Users of the API are expected to have knowledge in programming. The API interfaces are network based and programming language agnostic.

Look offers three kinds of APIs:

- A TCP based API server that can remote control the eye-tracker. A client can subscribe to topics such as real-time gaze data or status updates. Recommended for most cases. An example client in python is available.
- A HTTP REST-based API well suited for controlling the eye-tracker via http. It allows sending commands to the eye-tracker. No real-time access to the data is possible. The Look! mobile recording unit utilizes this interface. Access to the video is possible via the MJPG stream.
- A WebRTC Datachannel can be used when you transmit the video via WebRTC. As the overhead is quite large, it is not recommended to use this API in case you do not wish to access the video in real time.

5.1 Definitions

Scene camera the forward directed camera recording the scenery around the user.
Eye camera one of the cameras directed at the user's eye.
Gaze The 2D gaze pixel coordinate in the eye-tracker's scene camera view.
GazeMapped The 2D gaze pixel coordinate mapped to an object of interest (e.g., a screen).

This document assumes the Look! software to run locally on the device that the API access happens from. If this is not the case, you need to access the correct device via its IP address and make sure port 80 of the device running Look! is available and not used by another service. You can access the Look! mobile recording unit via Wifi by its IP 10.0.0.1, or its DNS address look.box.

The mobile recording unit provides a fully functional reference implementation of the API in Javascript. Follow the instructions to work with the recording unit to access its web interface.

5.2 License agreement

The software license agreement you consent with during installation of the Look! software is also applicable when accessing it via the API. Specifically, usage of the API must not circumvent a valid license for usage of the Look! Software.

The Look! Software and eye-tracking signal must not be used to drive safety-critical applications. Look! ET UG (haftungsbeschränkt9 does not make any guarantees with regard to reliability and accuracy of the gaze signal in these conditions. Any kind of military use cases require separate permission on a case-by-case base and are not covered by the standard license.

5.3 TCP

The TCP client can identify the Look! Server via Zeroconf. Look! will offer the service `_lookapi._tcp.local.` by default on port 7071. The actual port number can be gained from the Zeroconf properties on the offered service. Using Zeroconf is not required in case you know the IP address ad port on which Look! is running.

`I subject name`	set the subject name
`i`	get the subject name
`R`	start recording with the current subject
`R subject name`	start recording after changing the subject name
`r`	stop recording
`C`	start calibration
`c`	stop calibration
`v`	get the Look! version number
`S T`	subscribe to topic T [S,G,V,A]
`s T`	unsubscribe from topic T

`subject name` needs to contain at least 3 characters, followed by a space, followed by another at least 3 characters.
`T` defines the topic name and can be chosen from:

- **S**tatus - get eye-tracker status updates
- **G**aze - get live gaze data
- **V**ideo - get live video data
- **A**nalysis - get a gaze statistics report whenever one is produced

- **B**roadcast - everyone gets them, no need to subscribe, not possible to unsubscribe

You will not receive a response, but should receive a status update where appropriate, if you previously subscribed to status updates. A gaze data packet looks as following:
`G 0.00 0.00`
where G is the topic identifier, followed by two %.2f formatted x and y coordinates of calibrated gaze in the eye-tracker's scene camera image.

5.4 HTTP

The Look! Software offers a HTTP GET based API that receives simple commands and responds with JSON objects. This method can be used to send commends to the device (record, calibrate, playback) as well as to query its current state. The video stream can be received in real-time (with a certain delay depending on the mode of transmission) as a MJPG stream.

An implementation *example* of the API can be found in the `FlaskProtocol` interface. Implementing the defined functions can provide a fully functional GUI interface to the eye-tracker.

The eye-tracking software offers the following functionality, which can be requested via `http://IP_ADDRESS/cmd?COMMAND_STRING` (use 127.0.0.1 as IP_ADDRESS for a local connection). Where COMMAND_STRING is one of:

- **`c=status&a=get`**
  `Get eye-tracker status information`
  `response:` *`STATUSREPORT`* `{"c": "status", "a": "report", "p": {"subjectid": int, "recordingid": int, "tracker_mode": 'live', 'replay', 'analysis', "calibration_state": "uncalibrated", "calibrating", "calibrated", "recording_state": bool, "playback_state": "replay", "analysis", "playing_state": bool}}`
- **`c=rec&a=start`**
  `Start a new recording for the currently selected subject.`
  `response:` `STATUSREPORT`
- **`c=rec&a=stop`**
  `Stop a currently running recording.`
  `response:` `STATUSREPORT`

- **c=rec&a=marker**
  Place a marker in the recording (e.g., for synchronization).
  response: HTTP 200

- **c=cal&a=start**
  Start capturing calibration samples.
  response: STATUSREPORT

- **c=cal&a=stop**
  Stop capturing calibration samples and apply the new calibration from now on.
  response: STATUSREPORT

- **c=sbj&a=list**
  List all subjects currently in the database.
  response: SUBJECTLIST {"c": "sbj", "a": "list", "p": ["id": int, "forename": str, "familyname": str]}

- **c=sbj&a=select**
  Select a subject name for the recording. The name needs to contain at least 3 letters, followed by a whitespace, followed by 3 more letters. In case the subject is not found in the database, it will be created.
  response: SUBJECTLIST

- **c=status&a=seeksync**
  Position of the playback slider within a recorded video stream.
  response: {"c": "status", "a": "seeksync", "p": {"syncpos": float}}

  More undocumented functions are available, e.g. for recording playback and analysis.

5.5 WebRTC

We also offer an WebRTC interface that uses h264 or VP9 encoded video as well as a data channel to transfer commands. The http interface is however usually easier to use.

6 Optional extension modules

6.1 Streaming via WebRTC and MJPG

When streaming via MJPG, a web server provides access to the MJPG stream. An API allows to remote control the device. This way of streaming is recommended. Performance requirements for video compression are critical even for powerful systems and JPEG compression can often be done in hardware. The Look! Recording unit utilizes this interface.

It is possible to configure the software to provide a WebRTC stream of the video as well as a stream of the data. Please note that h264 or VP9 video compression might put substantial additional load on the recording device. In this configuration a WebRTC data channel can be used to remote control the software.

7 AI training

You should never be required to retrain our gaze estimation model during regular usage, yet we provide some information on the training process here. The built-in domain-specific data augmentation allows to perform training with relatively few recordings. For recording the training data, we made participants look at the center of a calibration marker. These can easily be detected in the scene camera image and tuples of eye images and gaze targets in the scene camera image can be acquired.

Other approaches abstract over the scene camera's calibration. We chose to predict scene camera pixel coordinates directly as well as an abstract representation of gaze direction relative to each eye camera's coordinate system. To achieve this, we decouple the gaze target pixel coordinates in the scene cam image from the eye image embedding generated by the neural network via a calibration network. That way the important gaze embedding network can learn its own bottleneck representation of a gaze orientation. All subject-specific parameters can be distilled to a calibration embedding generated by a separate neural network that is only used during training of the network. The resulting regression network can backprophagate its loss to optimize the embedding and the calibration network jointly.

The consequence of this gaze direction representation is that the device requires a calibration. As we recommend a personal calibration anyways, both steps can simply be merged. The gaze estimate is adjusted

to the subject's specific eye as well as to the specific scene camera's intrinsics. We implemented a calibration procedure following the CalibMe approach. This calibration is applied on top of the neural network gaze estimate. That way we achieve similar accuracy as when using EyeRecToo, but the network is not as susceptible to losing track of the pupil as the pupil detection methods employed in EyeRecToo are.

8 Demonstration of flexibility

Researchers are unlikely to fiddle around with their expensive eye-tracking gear, cut off pieces, replace modules or change their hardware. In this section we want to give an impression on how we utilize the ability to build our own, relatively inexpensive eye-tracking devices as well as the ability to easily modify them and what kinds of problems such modifications could possibly solve.

8.1 Angle of the scene camera

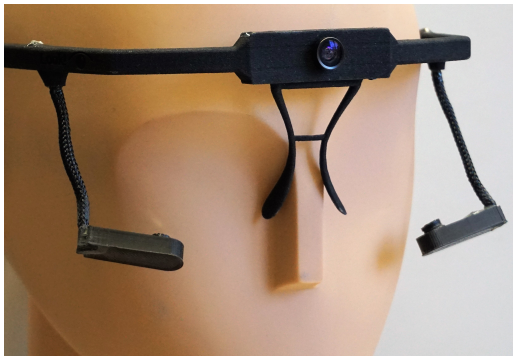
When recording tasks where the participants have to manipulate something with their hands, many scene cameras reach the limits of their field of view. We found some scene cameras not to cover that area at all. Generally, gaze accuracy deteriorates towards that region (as calibration is hard to perform, glints disappear on the sclera and camera viewing angle on the pupil is extreme). For these recordings we found it necessary to slightly tilt the scene camera of the Blickschulungsbrille towards the ground. Doing so is relatively easy with the provided 3D sketches. Additionally, we produced a variant with adjustable scene camera angle (figure 4b).

8.2 Flexible eye cameras

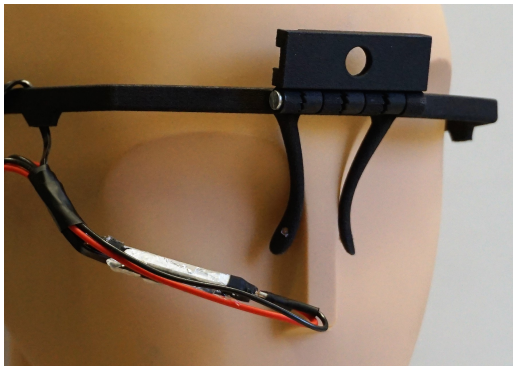
Having a rigid frame reduces mechanical wearing and makes the device easy to use. With a fixed eye camera position deep learning based methods can be trained on a smaller subset of images and a bad camera adjustment becomes unlikely. However, the flexibility gained by being able to move the eye cameras might be important in some cases. The frame of habitual eyeglasses might be in the way, the negative effect of eyelashes and dropping eyelids or reflections on eyeglasses can be reduced or completely avoided by a good, user-specific camera positioning. This can be achieved either by the use of multiple cameras, an adjustable nose piece that preserves the geometry between eye and scene cameras , or adjustable eye cameras.

To gain more flexibility in eye camera placement we replaced the connector between the eye camera and the frame with a flexible aluminium wire (Figure 4a). This setup allows for full freedom in the adjustment of the eye cameras without joint mechanics limitations. That makes finding a good camera position and orientation comparatively easy. We

found that the wire can withstand a reasonable number of readjustments without breaking when tightly fixed to the eye camera with epoxy and not bent too far in a single spot.



(a) Look! variant with flexible eye cameras that can be adjusted to work well with eyeglasses and the most variance in head shape.



(b) Look! variant with an acrylic diffuser as illumination pattern as well as an adjustable scene camera to capture the hand action space.

Figure 4

8.3 Diffuse illumination

As of now, head-mounted eye-tracking devices mostly utilize direct illumination via near-infrared LEDs. This wavelength is invisible to the human eye, yet provides a relatively constant illumination of the eye and therefore solid contrast between pupil, iris and sclera. Often the resulting glints are used to construct a geometrical model of the eyeball (or to compensate for device slippage), however current methods can work without them and utilize the LEDs only as a source of illumination.

Less explored is the use of diffuse illumination, i.e., scattering the source of light over a larger area. Doing so might reduce shadows due to eyelashes and increase the ease of adjusting cameras and illumination to individual eyes.

At first glance such an approach seems to make pupil detection even harder. In our experiments we found that even edge based methods perform similar for both approaches. We expect additional potential of this approach when utilizing deep learning based gaze estimation methods as similar illumination approaches are applied during cornea topography mapping. It might be possible to build more accurate cornea

models by combining the illuminator projection over multiple eye orientations. The curvature of the reflected illuminator bar could also be a helpful indication of cornea shape that could potentially be utilized by deep learning gaze estimation approaches. We did however not test this.

8.4 Line scanning sensors

Besides cameras, which are expensive and energy hungry, photodiode arrays are an interesting way to infer gaze locatio. Here, we demonstrate how the Blickschulungsbrille frame can be modified to house three line-scanning sensors (TSL1401CL), NIR-pass filters as well as their cabling. We placed the line scanners diagonally in order to try to overcome the accuracy issues of other photodiode based devices in vertical gaze estimation.

We evaluated the device on 6 subjects. Two of them participated both with and without eyeglasses, leading to 8 recordings that we consider as separate subjects in the following. Gaze prediction was performed with a fully connected neural network of 3 layers with 96/24/2 neurons and an input vector dimension of 384 that contained the concatenated readings of the three sensors. The network was trained on 20 calibration points within 34° field of view range and evaluated on another 12 points. A chin rest was used in order to fix the gaze targets relative to the subject's head. Average gaze estimation error on the evaluation points was 7.6° when trained on all subjects jointly. With networks trained specifically for one subject, we reached a mean error of 3.4-12.0° (mean 6.3°). These results are preliminary and the number of subjects very low, however they demonstrate the potential of similar approaches for future generations of mobile eye-tracking devices.