GazeSense

User Guide

Last revision: 7 January, 2022

1. Glossary

Point of regard	Visual representation of the user's gaze on the computer display. In other words, the point of regard is the inferred position on the screen where the user is looking at.
Primary screen	The user's main display screen. This can be, but is not limited to, the display used mainly by the user in his or her day-to-day activities. It also happens to be the screen used for the calibration process (see next row).
Calibration	The process of fine-tuning the tracking results by using the software's algorithms to understand the user's gaze.

2. Introduction

GazeSense ™ is a software application which allows to measure a subject's attention in real-time. The eye tracking technology by Eyeware can provide a high-level description of attention, which goes beyond the eye tracking signal. GazeSense can answer the following questions in real-time:

- Is the person looking at the table?
- Is participant A looking at participant B?
- Is the user looking at the robot?
- Where on the screen is the user looking at?

This document provides instructions on how to use the GazeSense software in your specific use case.

GazeSense main features:

- Gaze Tracking on your computer screen.
- Gaze Tracking on geometric targets in an environment setup, such as the layout of a room, or shop, or a car, etc.
- Pre-existing setup templates for different use cases.
- In-software modeling tool to build your own setups for tracking.
- Export of tracking data to text files for offline analysis. The user can choose which information (which fields) are exported.
- Real-time visualization of the gaze *point of regard* on the screen.
- Real-time visualization of gaze intersection in viewer for multiple screen mode.
- Real-time visualization head pose and gaze rays lines over the image stream.
- Simple 9 point calibration process, for increased tracking accuracy.
- Creation of different users, each with their own calibration configurations, that can be stored and reused for future sessions (no need to calibrate every time).

3. Hardware & Installation

This section provides details on the hardware required and on how to install the GazeSense software.

Hardware requirements

- **PC - Minimum requirements:**
  - 6th Generation Intel® Core™ processor, or newer
  - USB 3.0
  - 4GB RAM
  - Windows 10 or Ubuntu 18.04 LTS or higher LTS version
  - 1.2 GB of free hard drive space
  - Note: administrative rights are required to install camera drivers

Cameras Supported	**Windows 10**	**Linux (Ubuntu 18.04/20.04 LTS)**
Intel RealSense D415 (*required minimum firmware) (color/infrared)	✓ (Realsense Driver 2.27.0)	✓ (Kernel 5.0, Realsense Driver 2.25.0)
Intel RealSense D435 (*required minimum firmware) (color/infrared)	✓ (Realsense Driver 2.27.0)	✓ (Kernel 5.0, Realsense Driver 2.25.0)
Intel RealSense D435i (*required minimum firmware) (color/infrared)	✓ (Realsense Driver 2.27.0)	✓ (Kernel 5.0, Realsense Driver 2.25.0)
Webcam (color)	✓	✓
Orbbec Astra Embedded S (color)	✓	✓
Meerecompany S100D S.CUBE (infrared)	✓	✓
Azure Kinect (color/infrared)	✓	✓
Vzense DCAM710 (color/infrared)	✓	✓

Table 1: Cameras supported by GazeSense in Windows and Linux

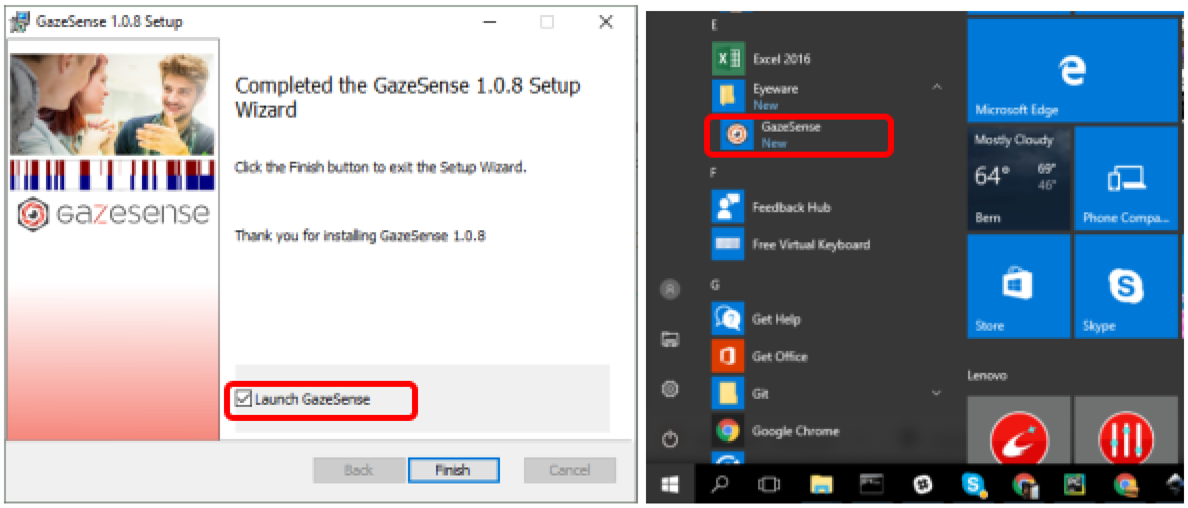
The first time you run the GazeSense software, you will be required to enter a license key. You should have a key available from your purchase of the software.

If you have trouble with your license, or if you would like to purchase more licenses, please contact us at **contact@eyeware.tech**.

Windows - Software installation

To install GazeSense on your Windows PC, please execute the following steps:

1. Uninstall all existing, older versions of GazeSense on your system.
2. Run the installer *GazeSense 2.X.X.exe* [1]
3. Follow the installation wizard instructions
   a. Please note: the wizard will install additional software packages needed for the correct functioning of GazeSense. Please accept the installation of those software packages, even if they have been installed previously.
4. Execute the application by either:
   a. Selecting the option “Launch GazeSense” after installation, or
   b. Clicking the GazeSense icon in the Windows menu, or
   c. Clicking on the GazeSense Desktop shortcut


GazeSense 1.0.8 Setup

Completed the GazeSense 1.0.8 Setup Wizard

Click the Finish button to exit the Setup Wizard.

Thank you for installing GazeSense 1.0.8

☑ Launch GazeSense

Figure 1: Different ways of starting GazeSense.

[1] Version may vary.

Linux - Software installation

1. Download the zip file
2. Extract the zip file on your system, for example in your home directory - the zip will create some directories
3. Enter the directory GazeSense_Linux64_<VERSION_NUMBER>
4. Run the following configuration scripts:
    a.
        - From a Linux terminal, enter the GazeSense directory and execute:
            - *sudo ./install/install.sh*
        - The above script copies USB rules for all supported cameras to your system. If it does not work automatically, then manually copy the *.rules files that you need (for the cameras that you need) to your /etc/udev/rules.d folder. Example for RealSense:
            - *sudo cp install/camera_drivers/realsense/99-realsense-libusb.rules /etc/udev/rules.d*

5. Execute the application by typing GazeSense in the Activities menu. If you cannot find it, open it by typing *./run_GazeSense.sh* in a terminal window that is open in the location of the files.

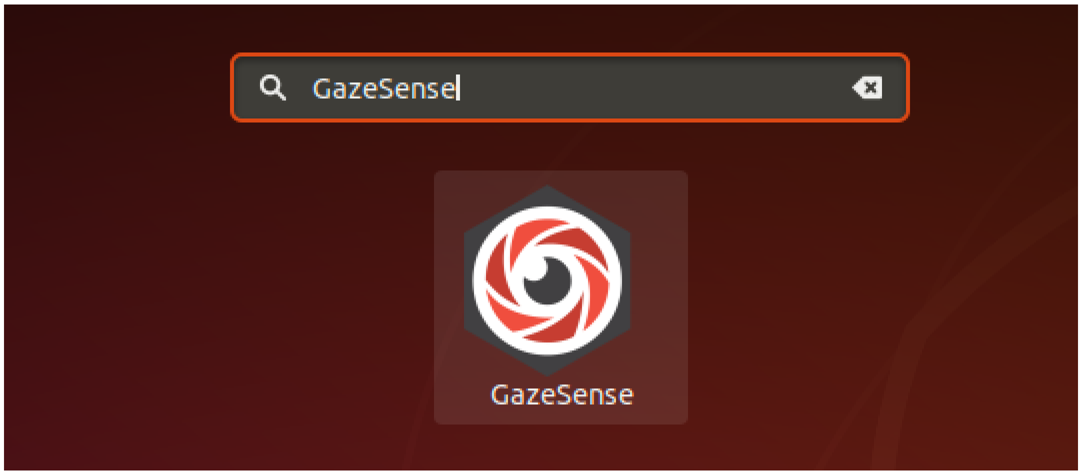
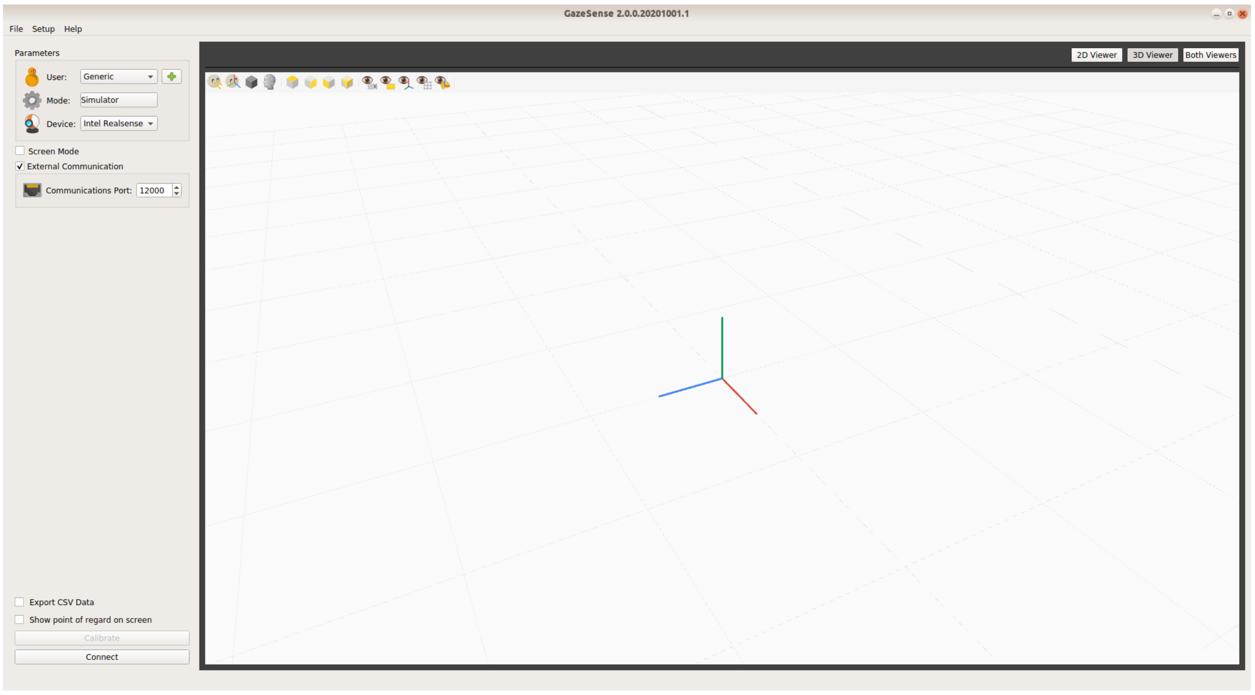


Figure 2: Location of GazeSense executable.

4. Main Window

The following is a screenshot of the Main Window from GazeSense.



Parameters

User: Generic

Mode: Simulator

Device: Intel Realsense

☐ Screen Mode

☑ External Communication

Communications Port: 12000

☐ Export CSV Data

☐ Show point of regard on screen

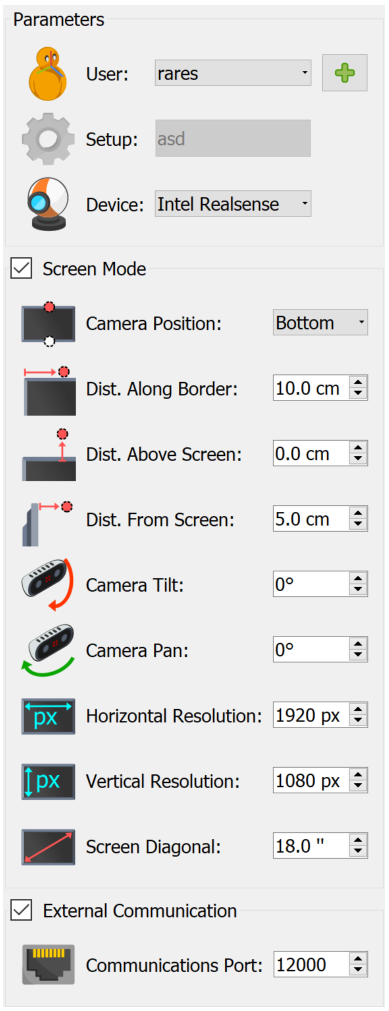
Figure 3: View of GazeSense Main Window. Left side: properties and the buttons for managing the tracking. Right side: 2D and visualization.

5. Run Modes

GazeSense offers two run modes: Screen Mode and Setup Mode.

Screen Mode

We can generally distinguish between two different categories for tracking: single screen, or multiple screens. The main difference is that in single screen mode (abbreviated as *Screen Mode*) we provide the actual physical characteristics of the screen where we are working on. We can activate **Screen Mode** by pressing the respective check box. This will open a previously hidden list of parameters:


Parameters

User:	rares
Setup:	asd
Device:	Intel Realsense

☑ Screen Mode

Camera Position:	Bottom
Dist. Along Border:	10.0 cm
Dist. Above Screen:	0.0 cm
Dist. From Screen:	5.0 cm
Camera Tilt:	0°
Camera Pan:	0°
Horizontal Resolution:	1920 px
Vertical Resolution:	1080 px
Screen Diagonal:	18.0 "

☑ External Communication

Communications Port:	12000

Figure 4: View of the Properties widgets when Screen Mode is activated. These properties accurately describe the physical setup of the user.

By accurately filling each of the parameters we can ensure a **more accurate tracking** as well as a better calibration process, which will be discussed further. All of the parameters are required to make sure that GazeSense knows the physical setup. Typically, you have one of these two configurations:

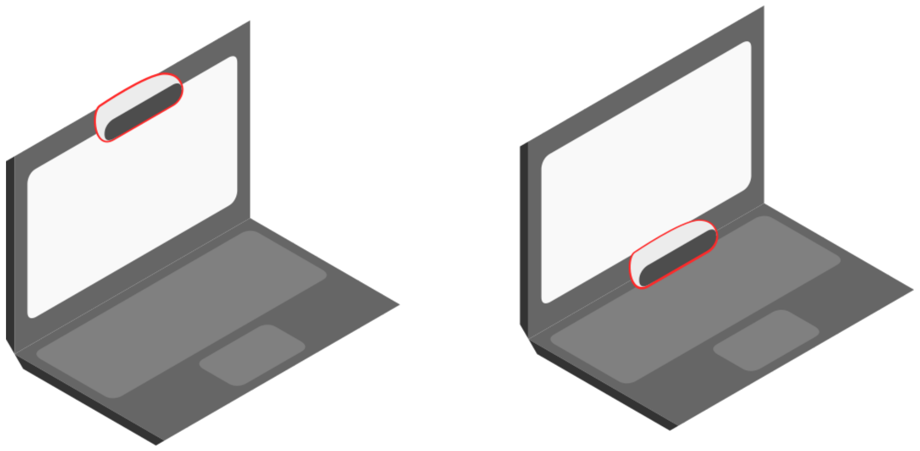


Figure 5: **(left)** Laptop configuration with the camera position on the top. **(right)** Laptop configuration with the camera position on the bottom.

You need to inform GazeSense about where the camera device is physically located. The options available are top or bottom. This can be chosen from the **<u>Camera Position</u>** widget (Figure 4).

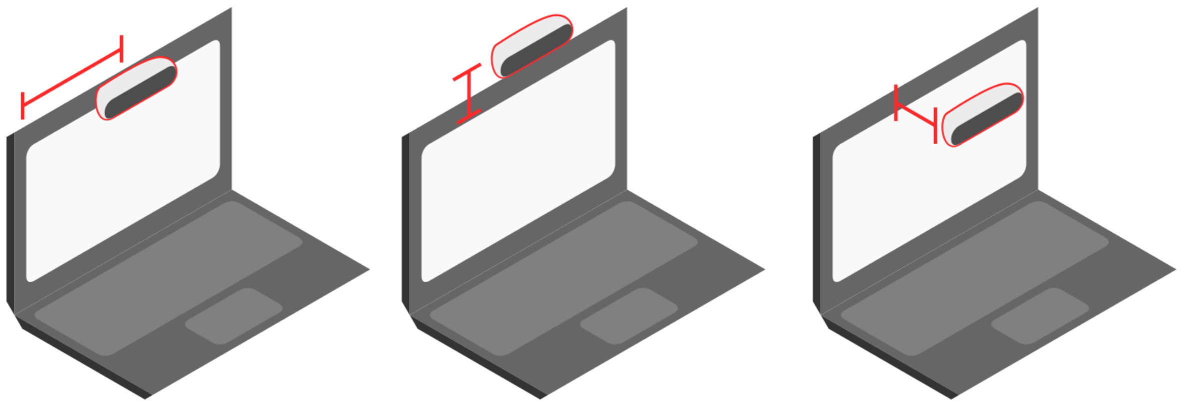


Figure 6: **(left)** "Distance Along Border", which describes the distance between the left border limit of the screen (not considering screen bezel) and the center of the camera. **(center)** "Distance Above Screen", which describes the distance between the top border limit of the screen (not considering screen bezel) and the camera center. **(right)** "Distance From Screen", which describes the distance between the place of the screen and the center of the camera.

We now know the general position of the camera. However, to have a precise location we can input the parameter **<u>Distance Along Border</u>**, which describes the length between the left screen border (do not consider the bezel or encasing of the digital screen) and the center of the device. The parameter **<u>Distance From Screen</u>** relates to the distance between the top limit of the screen and the center of the camera. The final coordinate parameter is named ***Distance Above Screen*** and describes the distance from which the camera is from the plane of the screen (Figure 6).

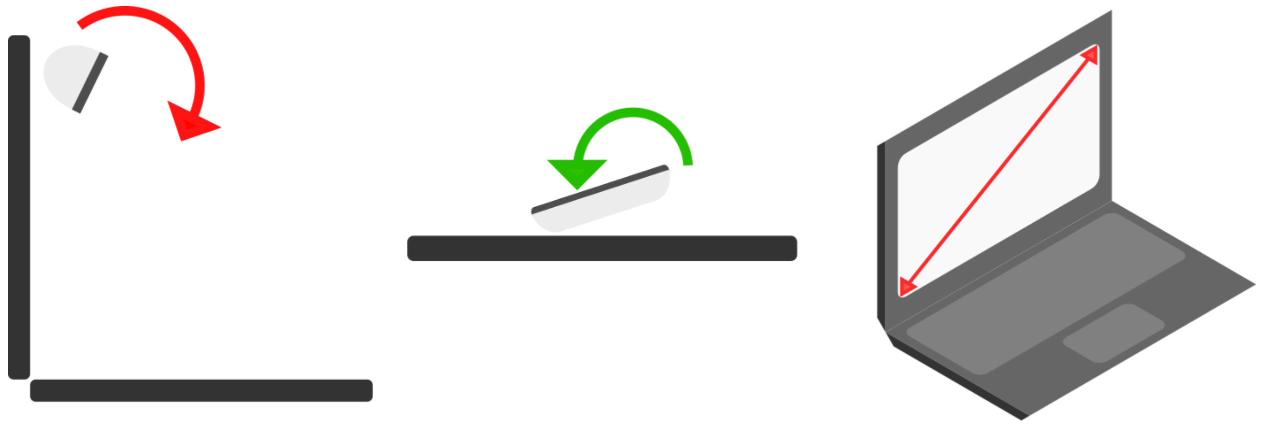
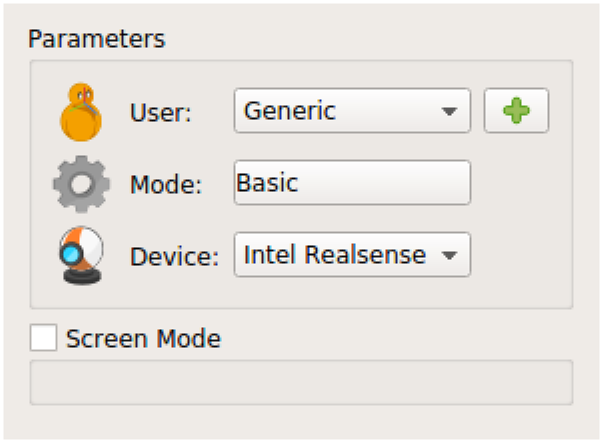


Figure 7: **(left)** "Camera Tilt" which measures, in degrees, the amount of rotation the camera has by comparison with a perfectly flat position. That is how much is it pointing up or down. **(center)** "Camera Pan" which measures, in degrees, the amount of rotation the camera has horizontally. That is: how much is it pointing left or right. **(right)** Diagonal, in inches, of the primary screen. That is the user's physical screen.

The last three parameters (Figure 4) are related to camera rotation and physical size of the screen. **<u>Camera Tilt</u>** is the vertical angle describing if the camera is pointing up, down, or flat (0 degrees). **<u>Camera Pan</u>** gives the horizontal rotation of the camera describing a looking left, looking right, or looking at the front (0 degrees) orientation. Finally, by providing the **<u>Screen Diagonal</u>** in inches, we have all the required parameters for *Screen Mode*.

Setup Mode

In Setup Mode (Figure 9), we can build any arbitrary setup with one or more screen targets of attention. The Mode button is the entry point to manage setups.


Parameters

User: Generic

Mode: Basic

Device: Intel Realsense

☐ Screen Mode

Figure 8: General parameters for GazeSense in Setup Mode. By clicking the Mode button, the user can select or create a new setup.

When clicking the Mode button, a new dialog will appear that will let us select a new setup (Figure 4).

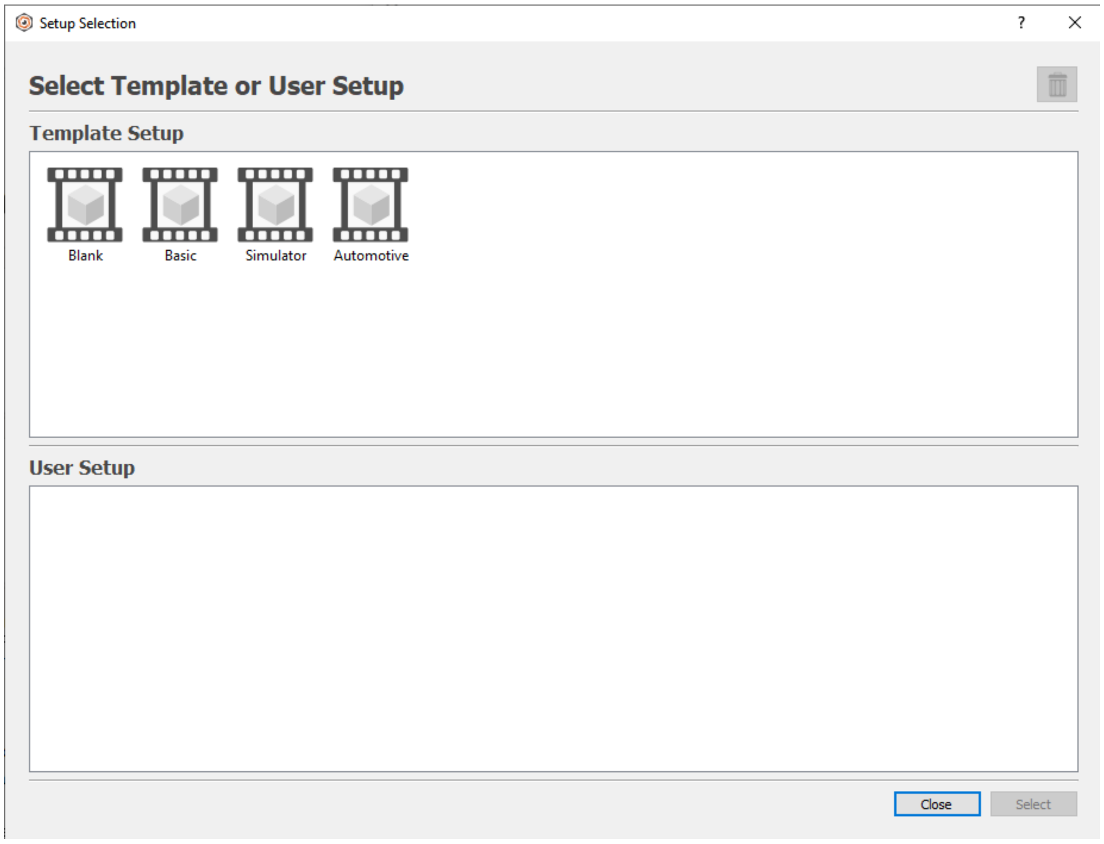
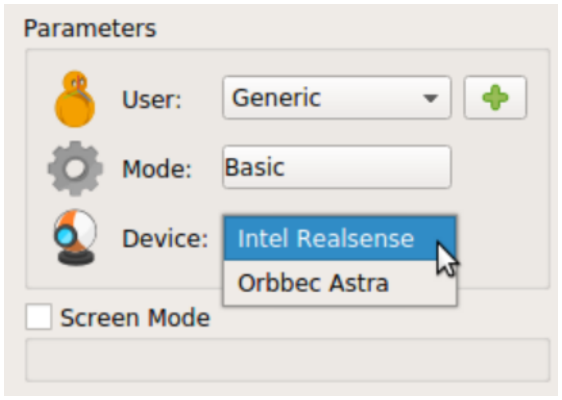


Figure 9: The Mode Selection dialog contains two lists. Templates (top) cannot be modified, but can be used to create new setups. User Setups (bottom) shows all user-defined environments which can be modified or deleted at will. By selecting a setup and clicking on the **Select** button, that setup will become the one being used by GazeSense .

By double-clicking on any of the template and user setups or selecting one and clicking on the Select button, the user will be sent to the editor panel. In the editor panel we can edit/create/delete Screen objects, as well as save setups on the disk for later reuse. If users make modifications, they will be prompted to save the setup when closing the dialog. Whatever setup was selected or created in this dialog, will be the one used by GazeSense. You can see the name of the setup used in the Mode button label (Figure 8). We will discuss further how to create and modify setups in Section 12 (Setup Builder). For now it's important to understand how to choose one.

6. Devices

The device is a medium to provide image streams as input to Eyeware proprietary technology for Gaze Tracking. This is typically a camera capable of producing RGB (color) plus D (depth) data. Notable examples are RealSense or Orbbec Astra cameras. The device can be changed by using the widget related to the **<u>Device</u>** parameter.


Parameters

User: Generic

Mode: Basic

Device: Intel Realsense

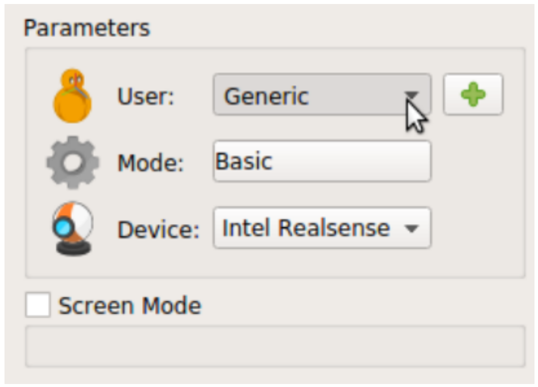
Orbbec Astra

Screen Mode

Figure 10: View of the widget responsible for the selection of the user streaming device in GazeSense.

7. User

The notion of User is associated with the concept of calibration. GazeSense can track better (i.e., provide more precise geometric measurements of attention tracking) if it knows who is being tracked, including but not limited to the user-specific head shape. This is achieved with the process of calibration. You can create different users, each with their own calibration. Or you can just use a Generic user and (optionally) perform the calibration. You can pick the user by clicking on the widget associated with the label **<u>User</u>**.


Parameters

User: Generic

Mode: Basic

Device: Intel Realsense

Screen Mode

Figure 11: By clicking on the User widget we can select which User (calibrated or uncalibrated person) is to be tracked. The plus button opens a dialog to further create, delete, or modify users.

You can also create, modify, or delete users by clicking the plus button at the right, or just modify it using the User choice widget. This will open a dialog (Figure 12) with a list of existing users. By double clicking on one you can change its name. By selecting one and clicking the remove button you can delete it (the “Generic” user is protected and cannot be deleted). The plus button can be used to create a new user. Clicking the OK button when a user is being selected from the list will make that the selected user in the software.

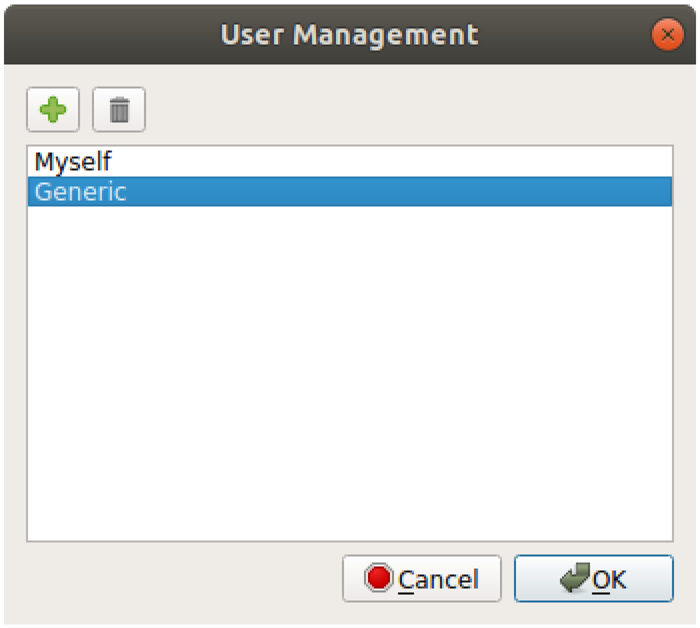


Figure 12: User Management dialog. The list presents all existing users, being “Generic” a protected value (which cannot be deleted or modified). By clicking the plus button a new User is created. Double clicking on a user enables us to change the User name. Furthermore when a user is selected, clicking on the garbage button will delete it.

Notice that for every User that is created, a calibration should be done in Screen Mode. Otherwise this may compromise the performance of the tracking algorithms

8. Tracking Profiles

Choosing one of the tracking profiles below can help optimize tracking performance for your use case.

Desktop (PC)

- Suitable for desktop setups - laptops, PCs and mobile phone scenarios, with tracked people.
- Expected to be close to the camera (less than 1.5 meter) and with moderate range of head movements.

Automotive (Driver Monitoring)

- Tracked people expected to be close to the camera (less than 1.5 meter) and with potentially large range of head movements.

Retail

- Suitable for retail setups, with tracked people expected to be farther from the camera and with a larger range of head movements.

9. Tracking (Connect/Disconnect)

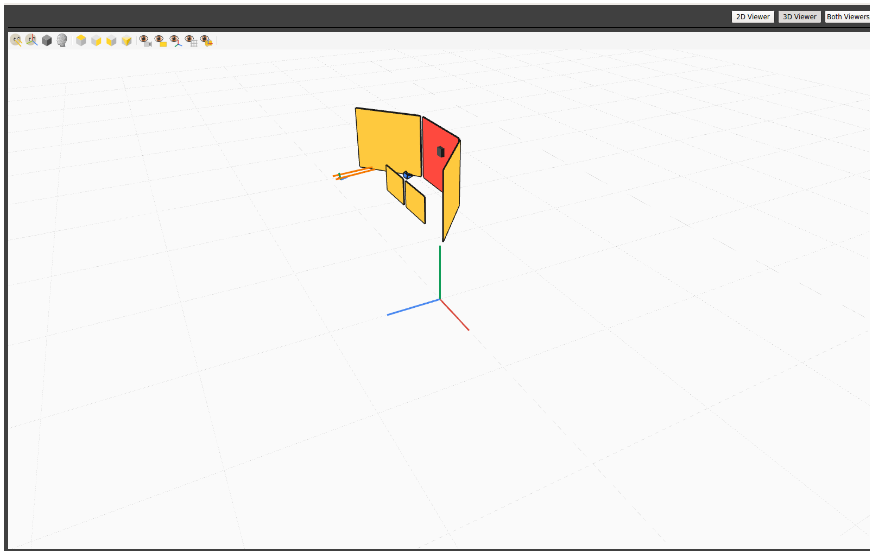


Figure 13: Main Window of GazeSense during tracking. The viewer (and 2D if visible) will be updated with the selected setup and tracking information. Once you connect, the Calibrate button will become active.

By choosing an existing Device (section 5), and a mode (section 3 and 4) you can click on the **Connect** button. This will automatically call your chosen setup into the main viewer (and 2D if this is visible) and the tracking will commence. The label on the button will also change to **Disconnect** to enable us to end the tracking session (Figure 13).

While the tracking is running, you can manipulate the viewer by rotating or panning over the perspective. The Gaze Rays will appear in orange. The head pose as a small red-green-blue orientation mark. Finally, the dark cubic marker represents the intersection between gaze and screen. As for the setup model, Screens appear as yellow rectangles, the camera as a blue pyramid.

On the viewer toolbar there are several visualization options for toggling on or off the components of the visualization.

Rotate the view	Click and drag left mouse button
Pan the view	Click and drag middle mouse button (wheel) or Shift + click and drag left mouse button
Zoom in and out	Roll middle mouse button (wheel)

10. Calibration

The calibration process is associated with the chosen User (see section 6). The calibration can be run whenever a connection is active. When this option is chosen a new window will appear, showing an orange circle (Figure 14).

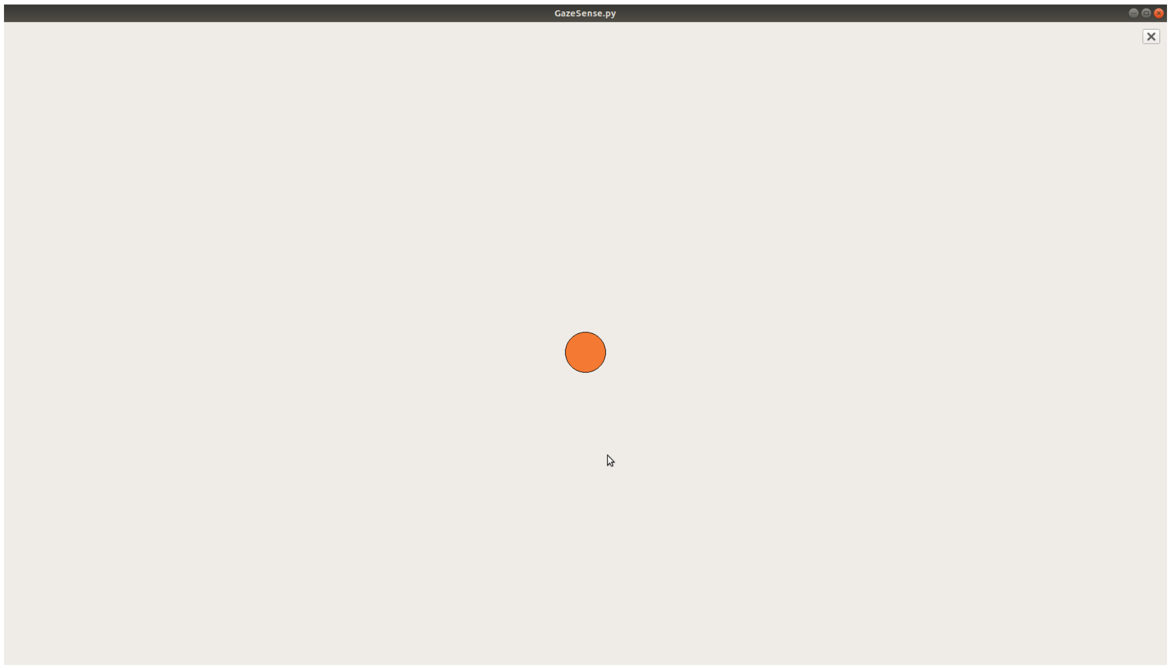
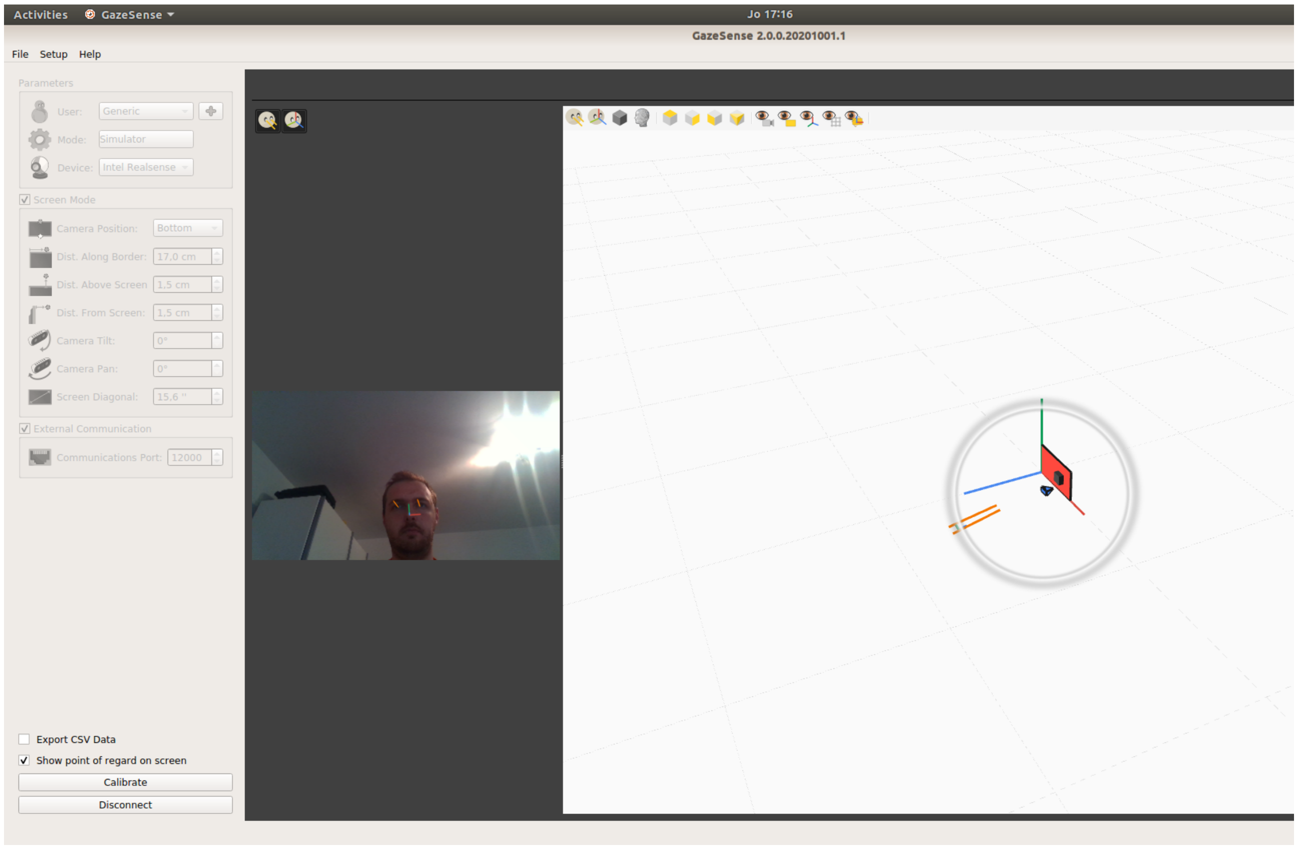


Figure 14: The calibration window showing orange circles that the User must gaze into and confirm (space key or left mouse button) for nine times, for nine different circle locations. At the end of the process the user will be calibrated.

Gaze at the orange circle and press the left mouse button or the space key on your keyboard. By doing so you'll be confirming that you were looking at the target and requesting a new one. By doing this procedure nine times for nine different circles you'll be completing the calibration process. The user's face must be frontal, and he must be gazing at the point while clicking. Also make sure that the user does not blink while clicking. *It's important to calibrate users, so that the quality of the tracking is as good as it can be*. Often, if the quality of the tracking is not good for a calibrated user, repeating this process might help greatly.

11. Point of Regard on Screen

The Screen Point of Regard is an optional feature when running in Screen Mode that allows us to see a visual representation of the user's gaze on the computer display (Figure 15). It will appear as a circle directly on the screen. This can be toggled on or off at any time during a connection, by using the checkbox **Show Point of Regard on Screen**.

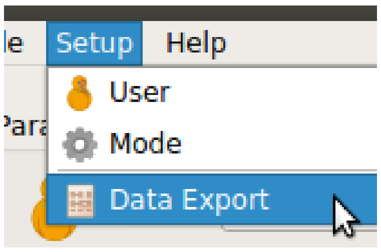


Parameters
User: Generic
Mode: Simulator
Device: Intel Realsense
[✓] Screen Mode
Camera Position: Bottom
Dist. Along Border: 17,0 cm
Dist. Above Screen: 1,5 cm
Dist. From Screen: 1,5 cm
Camera Tilt: 0°
Camera Pan: 0°
Screen Diagonal: 15,6 "
[✓] External Communication
Communications Port: 12000
[ ] Export CSV Data
[✓] Show point of regard on screen

Figure 15: Connection using a visual representation of the point of regard in Screen Mode. In this image the user is looking at the model of the setup being run

12. Export Data

Eye tracking and attention data can automatically be exported into a text file (in .csv format but using space as a column separator) in a location of your choice. This allows the post-processing of exported data, such as gaze and head tracking results, with any kind of software including Excel. You can choose which information (i.e., which fields) should be written in the exported file. The way to define these parameters is via the **Settings** option in the **Setup** at the top menu (Figure 16, left). The way to instruct the software to save a tracking session to file is by selecting the checkbox on the properties at the left side of the main window (Figure 16, right).



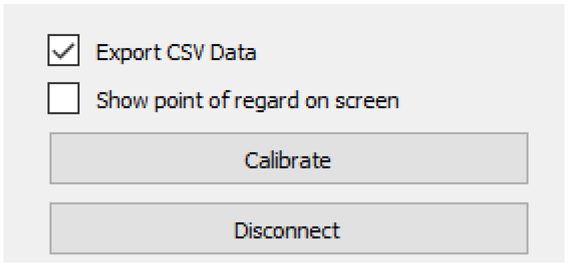
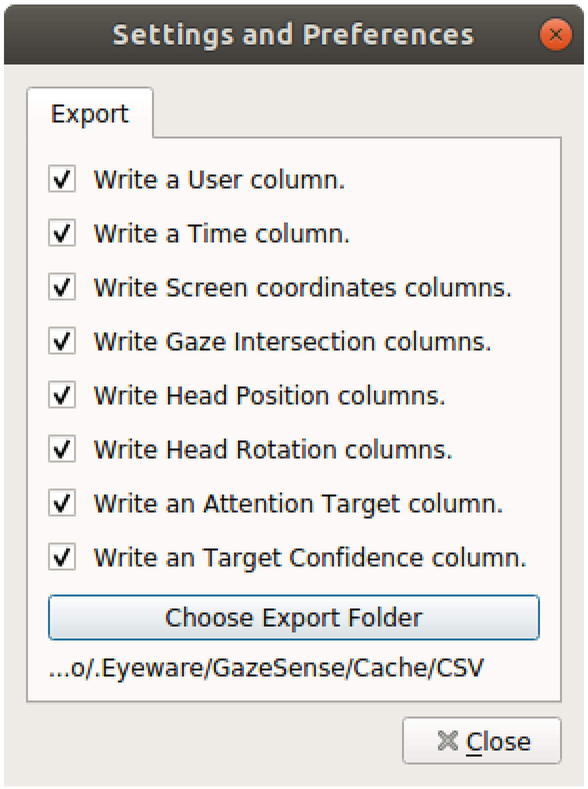


Figure 16: (**left**) The settings option at the Setup top menu will open the Data Export dialog where it is possible to choose the directory for the export data and which information is to be exported. (**right**) If the "Export CSV Data" checkbox is activated the data will be automatically written at the start of a tracking session.

When opening the Data Export dialog, a panel will be shown. In this panel you can choose which information (fields) is to be written to file. Furthermore, by clicking the button "Choose Export Folder" we can choose the output directory. Each of the saved files will be named with the tag "Session", followed by the date and hour of the start of the respective tracking session.



Settings and Preferences

Export

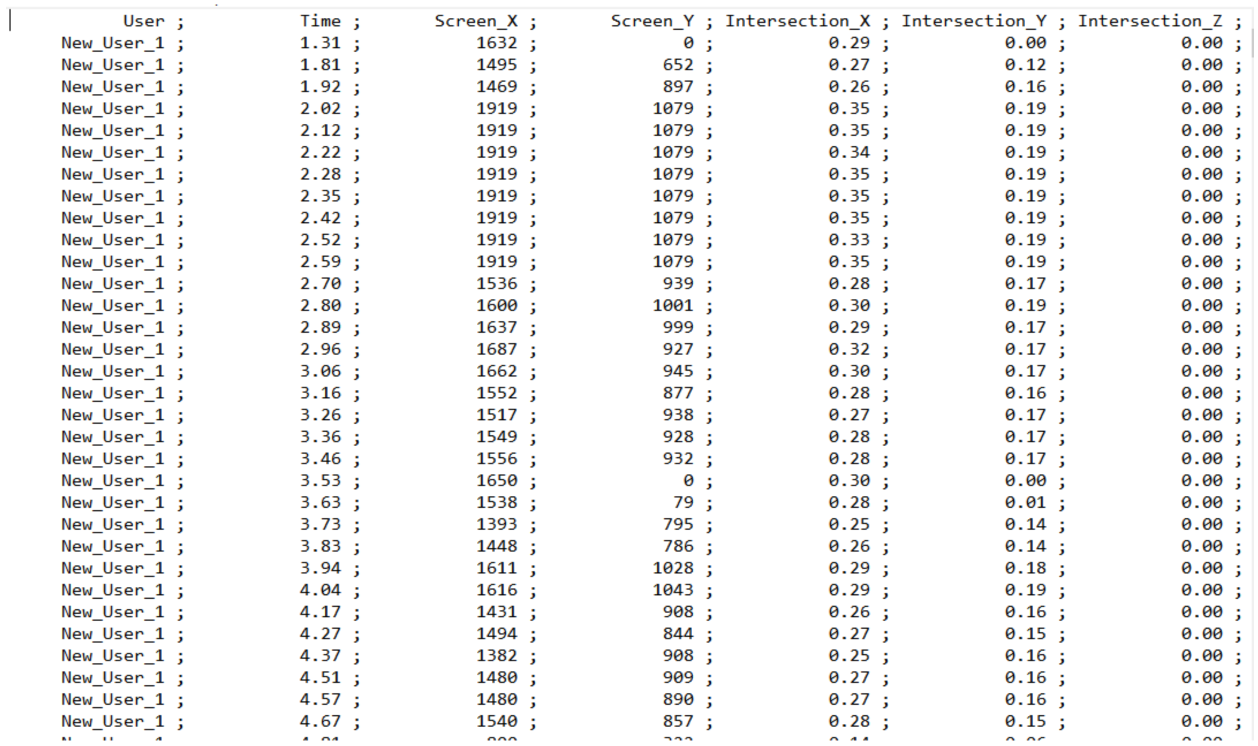
- [✓] Write a User column.
- [✓] Write a Time column.
- [✓] Write Screen coordinates columns.
- [✓] Write Gaze Intersection columns.
- [✓] Write Head Position columns.
- [✓] Write Head Rotation columns.
- [✓] Write an Attention Target column.
- [✓] Write an Target Confidence column.

Choose Export Folder

...o/.Eyeware/GazeSense/Cache/CSV

Close

Figure 17: View of the Settings dialog. Each of the check boxes represent a possible field to be exported to text files when a tracking session occurs. By clicking on the "Choose Export Folder" button, the user can choose the folder where data files will be saved. Below you can see an example of a .csv file.



```
        User ;        Time ;    Screen_X ;    Screen_Y ; Intersection_X ; Intersection_Y ; Intersection_Z ;
  New_User_1 ;        1.31 ;        1632 ;           0 ;           0.29 ;           0.00 ;           0.00 ;
  New_User_1 ;        1.81 ;        1495 ;         652 ;           0.27 ;           0.12 ;           0.00 ;
  New_User_1 ;        1.92 ;        1469 ;         897 ;           0.26 ;           0.16 ;           0.00 ;
  New_User_1 ;        2.02 ;        1919 ;        1079 ;           0.35 ;           0.19 ;           0.00 ;
  New_User_1 ;        2.12 ;        1919 ;        1079 ;           0.35 ;           0.19 ;           0.00 ;
  New_User_1 ;        2.22 ;        1919 ;        1079 ;           0.34 ;           0.19 ;           0.00 ;
  New_User_1 ;        2.28 ;        1919 ;        1079 ;           0.35 ;           0.19 ;           0.00 ;
  New_User_1 ;        2.35 ;        1919 ;        1079 ;           0.35 ;           0.19 ;           0.00 ;
  New_User_1 ;        2.42 ;        1919 ;        1079 ;           0.35 ;           0.19 ;           0.00 ;
  New_User_1 ;        2.52 ;        1919 ;        1079 ;           0.33 ;           0.19 ;           0.00 ;
  New_User_1 ;        2.59 ;        1919 ;        1079 ;           0.35 ;           0.19 ;           0.00 ;
  New_User_1 ;        2.70 ;        1536 ;         939 ;           0.28 ;           0.17 ;           0.00 ;
  New_User_1 ;        2.80 ;        1600 ;        1001 ;           0.30 ;           0.19 ;           0.00 ;
  New_User_1 ;        2.89 ;        1637 ;         999 ;           0.29 ;           0.17 ;           0.00 ;
  New_User_1 ;        2.96 ;        1687 ;         927 ;           0.32 ;           0.17 ;           0.00 ;
  New_User_1 ;        3.06 ;        1662 ;         945 ;           0.30 ;           0.17 ;           0.00 ;
  New_User_1 ;        3.16 ;        1552 ;         877 ;           0.28 ;           0.16 ;           0.00 ;
  New_User_1 ;        3.26 ;        1517 ;         938 ;           0.27 ;           0.17 ;           0.00 ;
  New_User_1 ;        3.36 ;        1549 ;         928 ;           0.28 ;           0.17 ;           0.00 ;
  New_User_1 ;        3.46 ;        1556 ;         932 ;           0.28 ;           0.17 ;           0.00 ;
  New_User_1 ;        3.53 ;        1650 ;           0 ;           0.30 ;           0.00 ;           0.00 ;
  New_User_1 ;        3.63 ;        1538 ;          79 ;           0.28 ;           0.01 ;           0.00 ;
  New_User_1 ;        3.73 ;        1393 ;         795 ;           0.25 ;           0.14 ;           0.00 ;
  New_User_1 ;        3.83 ;        1448 ;         786 ;           0.26 ;           0.14 ;           0.00 ;
  New_User_1 ;        3.94 ;        1611 ;        1028 ;           0.29 ;           0.18 ;           0.00 ;
  New_User_1 ;        4.04 ;        1616 ;        1043 ;           0.29 ;           0.19 ;           0.00 ;
  New_User_1 ;        4.17 ;        1431 ;         908 ;           0.26 ;           0.16 ;           0.00 ;
  New_User_1 ;        4.27 ;        1494 ;         844 ;           0.27 ;           0.15 ;           0.00 ;
  New_User_1 ;        4.37 ;        1382 ;         908 ;           0.25 ;           0.16 ;           0.00 ;
  New_User_1 ;        4.51 ;        1480 ;         909 ;           0.27 ;           0.16 ;           0.00 ;
  New_User_1 ;        4.57 ;        1480 ;         890 ;           0.27 ;           0.16 ;           0.00 ;
  New_User_1 ;        4.67 ;        1540 ;         857 ;           0.28 ;           0.15 ;           0.00 ;
```

13. Main Viewers

In the Main Window of GazeSense we can see a visualization area on the right of the layout. By default the viewer is occupying the whole area, but the three buttons at the top right allow for a choice about the visualization.

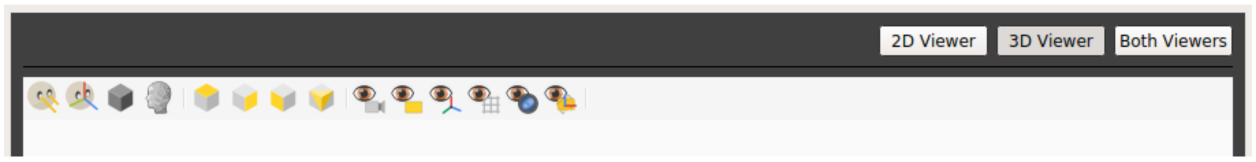


Figure 18: Top area for GazeSense viewer in the Main Window. By clicking the 2D viewer, button only the texture or color device stream will be displayed. Clicking on the Viewer will enable only the viewer. Clicking on Both Viewers will put both viewers side by side.

Once a connection has started (see section 7), the viewers will be updated with their respective content. The 2D viewer will show the stream from the device plus the lines representing the gaze lines and the head pose. You can toggle their visibility by using the buttons on the toolbar of this viewer. The viewer will show the setup being run plus the gaze lines, head pose, and gaze intersection. There will also be other symbols to facilitate orientation and analysis while navigating the setup. Those include the center axis mark, the grid floor, and several other toggles for the visibility of the different elements (e.g., showing the head lines, showing the gaze lines, etc.). The view can be rotated with the left mouse button, and dragged (or panned) with the middle mouse button. Rolling the middle mouse button will zoom in/out.

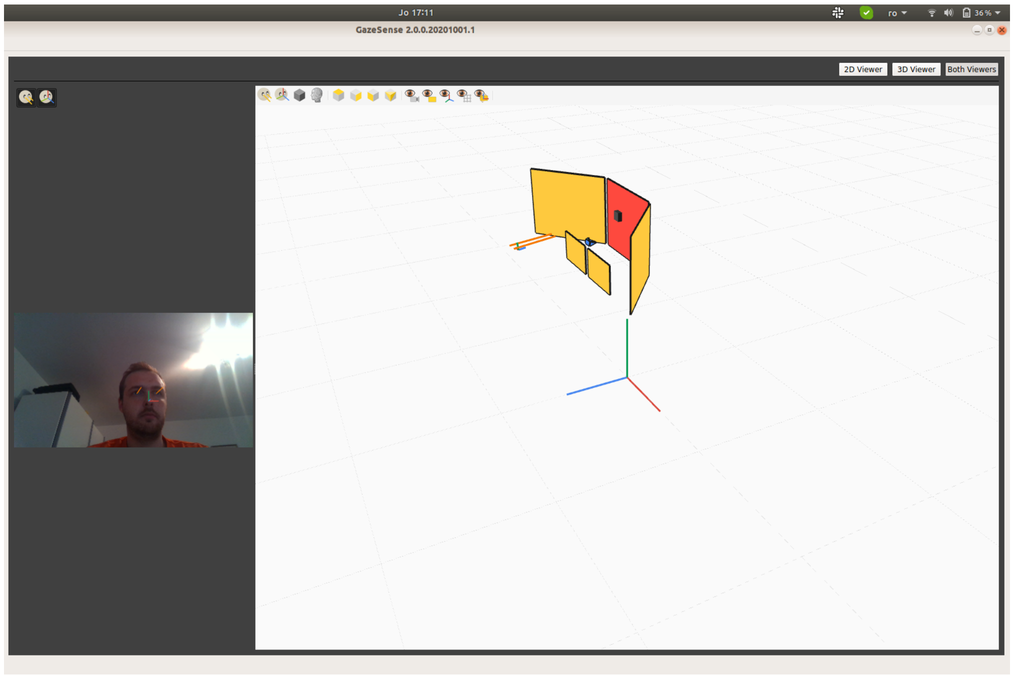
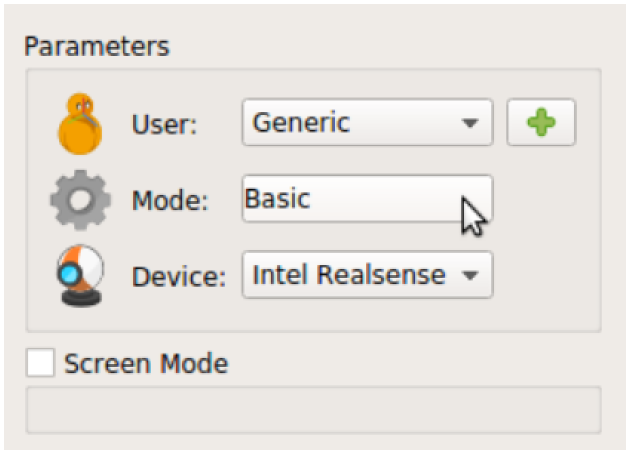


Figure 19: Main viewer of GazeSense while a connection is on. Right side: model with gaze, head pose, gaze intersection, among other elements from the setup. Both viewers are interactive in real time, either by toggling on/off the visibility of visual elements, or by changing the perspective (viewer).

Once a disconnection occurs, both viewers will be cleaned of content. They will refresh if a new connection is made.

14. Setup Builder

As mentioned in Section 4.1 (Setup Mode), you can choose, create, or edit a setup by clicking the button associated with the **Mode** option (Figure 20).



User: Generic
Mode: Basic
Device: Intel Realsense
Screen Mode

Figure 20: View of widget responsible for calling the dialog where a setup can be created, modified, deleted, and selected. Notice that this will only be available if the Screen Mode is unchecked.

The Mode Selection dialog will, by default, appear in such a way that you can choose what setup to use. By double clicking on any of the setups, or by a simple click followed by the Select button, you can select a setup (Figure 21).

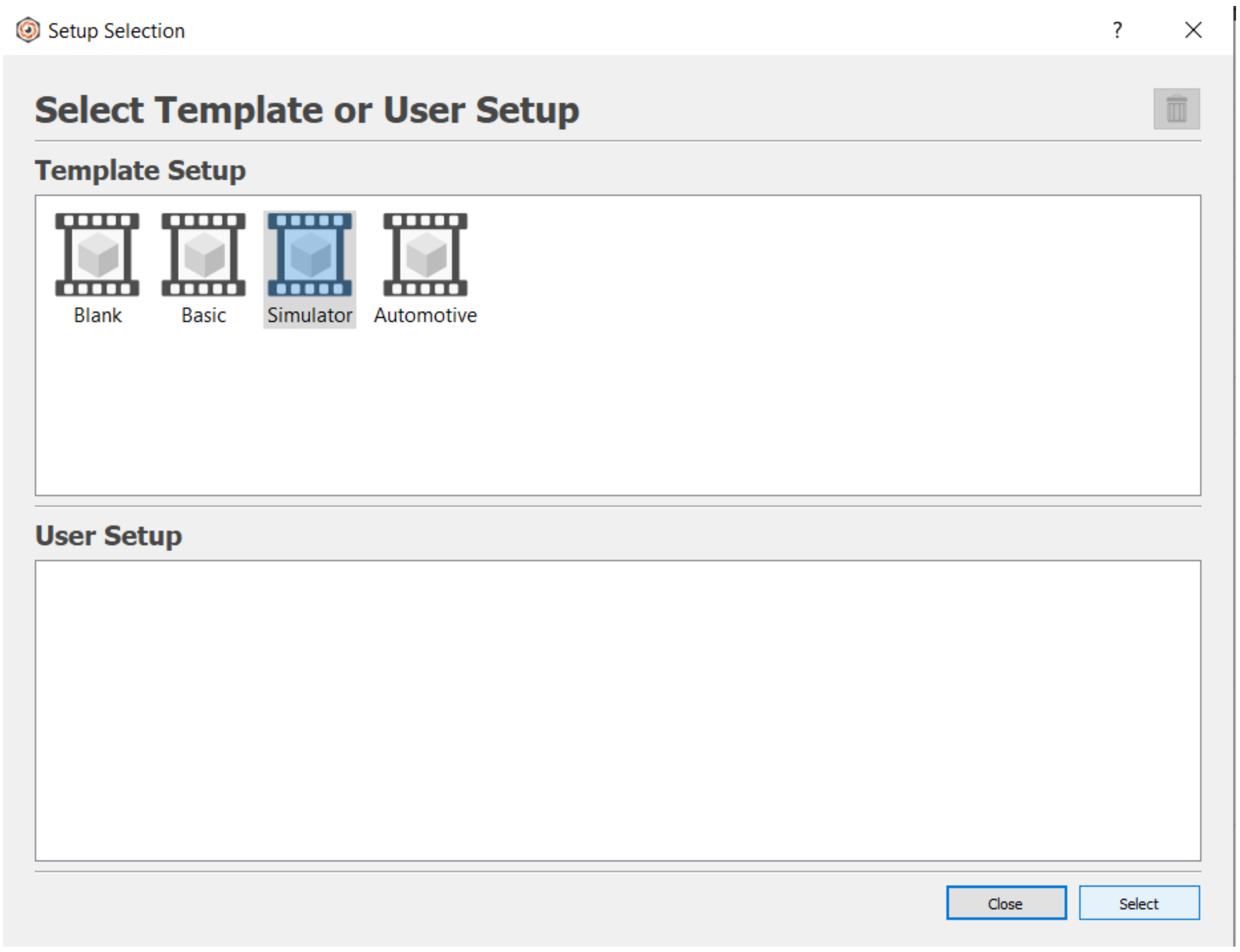


Figure 21: Mode Selection setup selection panel. The list on top refers to template setups which can be used to create customized scenes. The list on the bottom are the user setups, which can be modified or deleted using the Rubbish button on the top-right corner. To select a setup, just double click on it, or do a single click followed by the Select button.

If you don't yet have a setup (a User Setup) you can use a template to create your first one. If you do have an unwanted User Setup you can click on it and use the Rubbish button on the top-right corner to delete it. When selecting a setup the editor panel will appear (Figure 22).

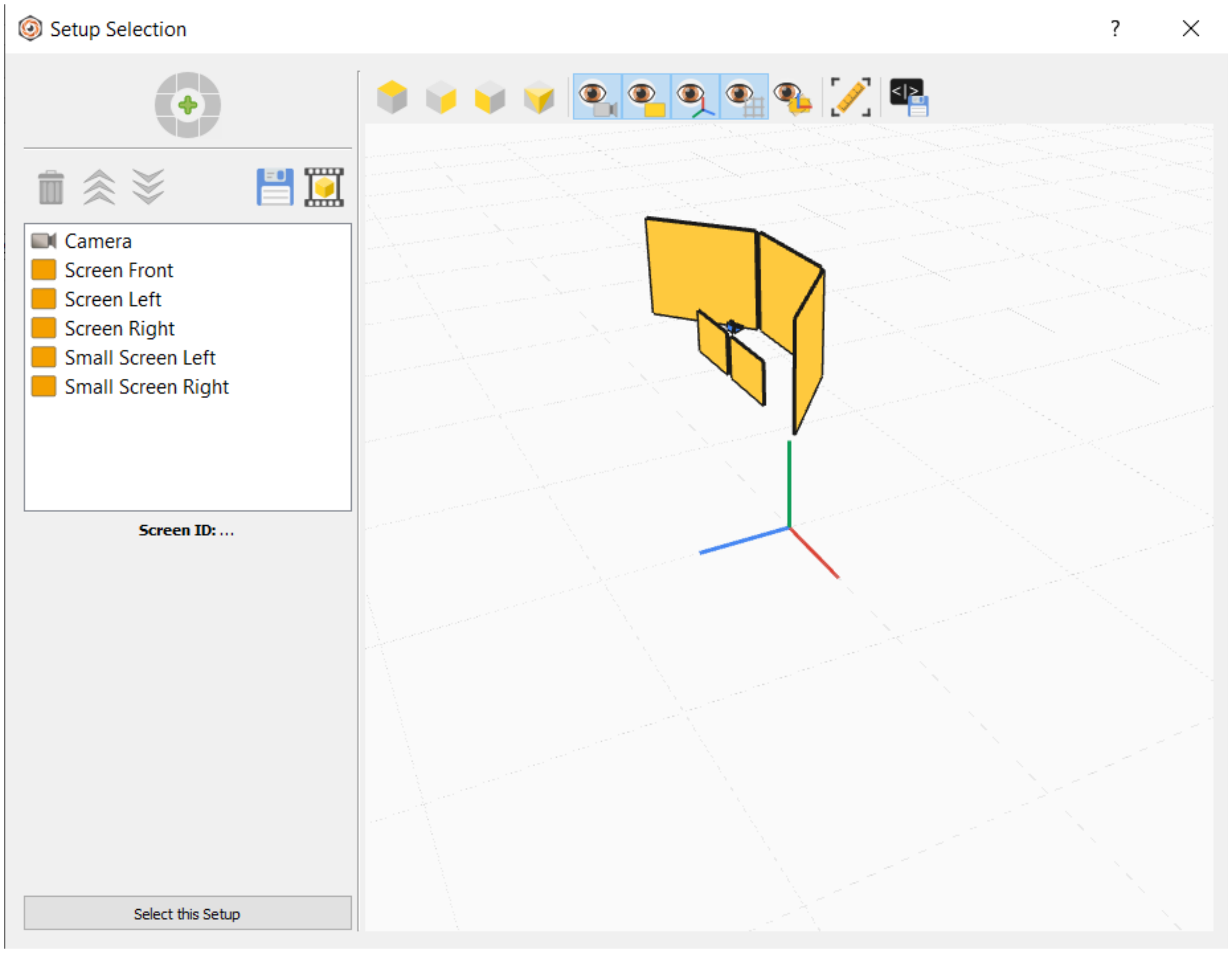
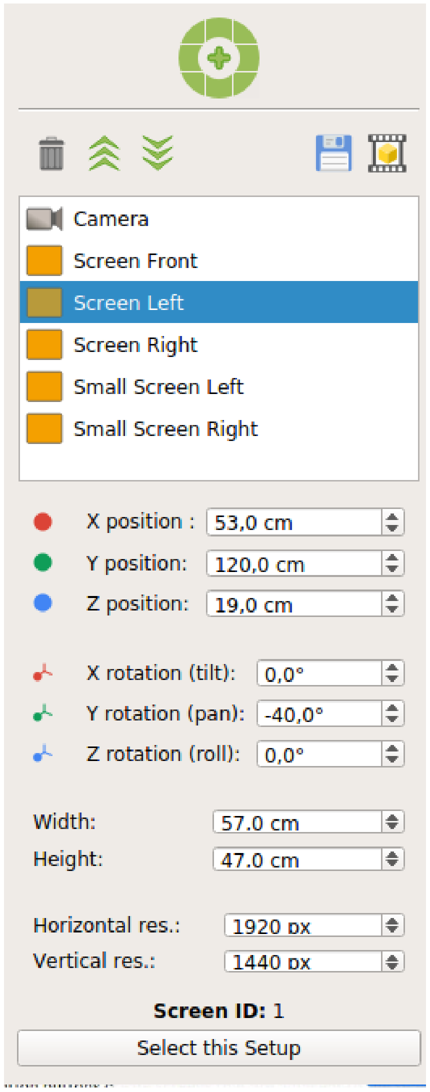


Figure 22: View of the Editor panel on the Setup Selection dialog. Right side: visualization of the setup. Left side: list of available objects. By clicking on one of the objects, a window will appear to allow you to manipulate its geometric properties such as the position, rotation or size.

If you click in any of the objects on the list, the property widgets that are specific to that object will appear. For example, for an object such as a Screen (yellow rectangle) you'll be able to change the rotation, position, width, height, and resolution. Any change that happens in the widgets automatically updates the viewer. The rotation is always relative to the center of the object and the position is given in cm (Figure 23).



Camera
Screen Front
Screen Left
Screen Right
Small Screen Left
Small Screen Right

X position :	53,0 cm
Y position:	120,0 cm
Z position:	19,0 cm
X rotation (tilt):	0,0°
Y rotation (pan):	-40,0°
Z rotation (roll):	0,0°
Width:	57.0 cm
Height:	47.0 cm
Horizontal res.:	1920 px
Vertical res.:	1440 px

Screen ID: 1

Figure 23: Properties of the selected Screen object. You'll notice the widgets to enable changes on the position, rotation, width, and height of the screen. Any change on these widgets will cause an immediate update on the viewer so you can adjust your scenario by eye.

Unlike Screens, the Camera object does not have width or height, or resolution. But you still need to provide position and rotation. If you wish to create a new Screen you can use the **plus button** at the top. This will create a screen position in the origin. However if you select an existing Screen and click on a **directional button** (i.e., the small circle sections around the plus) you can create Screens adjacent to the selected one. Moreover by clicking on the floppy disk button you'll be able to save your current setup. With the yellow scene button you can return to the Select Setup panel (Figure 24).

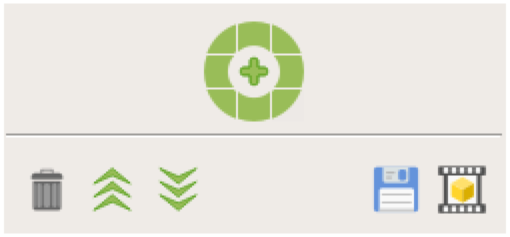


Figure 24: View of the Screen creation button and the objects toolbar. The plus button creates a Screen in the origin position. The adjacent creation buttons create screens that are adjacent to the selected screen in the same direction of the button. The garbage button deletes the selected objects. And the move buttons allows moving the objects on the list. The disk is used to save the setup, and the yellow scene to return to the Select Scene panel.

In the viewer you can toggle the visibility of multiple elements on the scene. You can also select an object with the right mouse button. The left mouse button is used to rotate the perspective, and the middle mouse button to pan it (drag). By rolling the middle mouse button you can also zoom in or zoom out.

15. Import/Export Setups

You can both load or save your setups locally. If you have a setup available you can use the option Export Setup to save to file the setup that you have currently selected. This file can then be shared with your colleagues or other GazeSense users to be loaded into their system. The way to do this is by using the option Import Setup. You can select a valid GazeSense setup file to be loaded into your system (Figure 25).

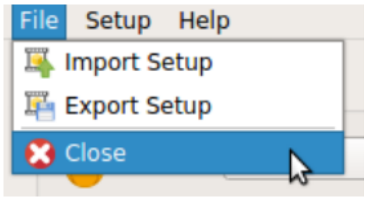


Figure 25: View of the top menu options to Import Setup and Export Setup. The first option allows the user to import an external setup from a file, loading it into GazeSense . The second option allows the export of the current selected setup to be saved into a shareable file.

16. Network API

GazeSense can be used together with third-party software by using a Network API. This is done with a Python script that listens to "tracking events" coming from GazeSense. For each event, the script does something with the tracking data, such as printing the geometric values in a terminal, or playing a sound, etc. This Network API is thus a valuable and simple way of building powerful, interactive demos with a few lines of Python code.

The process to create and run a demo is the following:

- Create a script (see example 1) that allows you to read processed frames from GazeSense.
- In a Command Prompt, set up an environment to run the script as explained below.
- Open GazeSense.
- Make sure to specify the **External Communication Listener Port** (see figure 26).
- Press Connect in GazeSense, in order to start tracking.
- Execute the script. This will start receiving the tracking information and perform the demo actions (e.g., printing in a terminal, playing a sound, etc.).

Creating a script

In order to create a script, take a look at the example below. In this minimal viable example you can see how we create a *Client*, specifying the host and the port, that will listen to the setup being run by GazeSense. The listener class is a derivation of the base class *TrackerListener* that contains a callback method called *on_track_ready*. This method will be called every time a new *TrackingEvent* is produced. Inside this tracking event, we have access to all the information regarding gaze direction, head pose and other tracking data. The following is a complete functional example of a script made to print to terminal the pixel coordinates of the gaze of the user.

```python
from gazesense_client import Client, TrackerListener, TrackingEvent
import time

class MyListener(TrackerListener):
    def __init__(self):
        super().__init__()
    def on_track_ready(self, event: TrackingEvent) -> None:
        print("No of people tracked: ", len(event.people))
        if len(event.people) < 1:
            return
        screen_gaze_is_lost = event.people[0].tracking_info.screen_gaze.is_lost
        print("      - Lost track:       ", screen_gaze_is_lost)
        screen_gaze = event.people[0].tracking_info.screen_gaze
        if not screen_gaze_is_lost:
            print("      - Screen Id:        ", screen_gaze.screen_id)
            print("      - X Coordinate:     ", screen_gaze.screen_gaze.x)
            print("      - Y Coordinate:     ", screen_gaze.screen_gaze.y)

        print("======================================================")

# Global Parameters
host = "localhost"
listener_port = 12000
# Building client and listener
client = Client(host, listener_port)
listener = MyListener()
client.register_tracker_listener(listener)
# Making sure the script remains open for a long time
time.sleep(100000)
```

Example 1. Executable script for GazeSense .

But let us analyze each of the relevant code blocks to build a customized sample. We start by defining the fundamental imports to make the connection possible.

```
from gazesense_client import Client, TrackerListener, TrackingEvent
import time
```

The first import refers to the objects provided by Eyeware to build the client and the listener. The second module, *time*, we use as an optional command to prevent the script from closing right after being run. It's the reason as to why we insert this instruction at the end of the script:

```
# Making sure the script remains open for a long time
time.sleep(100000)
```

The Client itself requires a host, typically the local host, and a port which must be the same specified in the "External Communication" group in the main window of GazeSense (Figure 26).

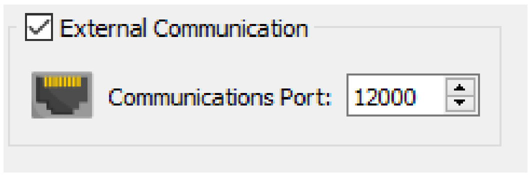


Figure 26 - View of the External Communication group as can be seen in the main window of GazeSense. The port value inserted here must be the same as the one specified in the script.

We also need to register a listener to the *Client* using the method *register_tracker_listener*.

```
# Global Parameters
host = "localhost"
listener_port = 12000
# Building client and listener
client = Client(host, listener_port)
listener = MyListener()
client.register_tracker_listener(listener)
```

You'll notice the creation and use of the listener, named in our example code sample as *MyListener*. This is optional for the user, but we must make sure that the class that we use as a listener uses the *TrackerListener* as base class.

```
class MyListener(TrackerListener):
    def __init__(self):
        super().__init__()
    def on_track_ready(self, event: TrackingEvent) -> None:
        print("No of people tracked: ", len(event.people))
```

The construction of this class can be done consistently throughout many different scripts, but the contents of the *on_track_ready* method will depend on the purpose for the script. In our example we output the screen coordinate in pixels for the gaze, but we could as easily play a sound when a target is being observed, animate an avatar or communicate with some third-party software. The fields available in each of the events can be seen in the following table.

Note: Our standard for a left or right eye is from the point of view of the camera (as visualized on the screen), not of the tracked subject.

Call	Type	Description
TrackingEvent.people	List[TrackedUser]	A list of all tracked users. A single user can be accessed as *event.people[0]*
TrackingEvent.timestamp	float	The value of time in seconds with origin in the beginning of the stream.
TrackedUser.tracking_info	TrackedPersonInfo	A container of all person related tracked information.
TrackedPersonInfo.head_pose	HeadPoseInfo	A container of all head pose related information.
HeadPoseInfo.transform	AffineTransform	A container for the rotation and translation for head pose.
AffineTransform.rotation	numpy.array	A 3x3 rotation matrix, with respect to the World Coordinate System (see Section 16 for more information).
AffineTransform.translation	numpy.array	A vector with components X, Y, and Z of the tip of the tracked user's nose, with respect to the World Coordinate System (see Section 16 for more information)
HeadPoseInfo.is_lost	bool	A boolean flag set to True if no person is being detected.
TrackedPersonInfo.gaze	GazeInfo	A container of all gaze related information.
GazeInfo.left_eye_ray	Ray	A container of origin and gaze direction for the left eye.
GazeInfo.right_eye_ray	Ray	A container of origin and gaze direction for the right eye.
Ray.origin	numpy.array	A vector with components X, Y, Z of the iris of the eye, with respect to the World Coordinate System (see Section 16 for more information).
Ray.direction	numpy.array	A vector with components X, Y, Z and unitary norm, containing the direction of the eye gaze, with respect to the World Coordinate System (see Section 16 for more information).
GazeInfo.confidence_left	TrackingConfidence, str	A string value with the confidence of the measures made for the left eye gaze: UNRELIABLE, LOW, MEDIUM, HIGH
GazeInfo.confidence_right	TrackingConfidence, str	A string value with the confidence of the measures made for the right eye gaze: UNRELIABLE, LOW, MEDIUM, HIGH
TrackedPersonInfo.screen_gaze	ScreenGazeInfo	A container of information regarding the target being gazed at.
ScreenGazeInfo.screen_id	int	A unique integer to the target screen being observed. These unique integers can be seen in the Setup Builder window.
ScreenGazeInfo.x	int	Horizontal pixel coordinate of the gaze intersection on screen.
ScreenGazeInfo.y	int	Vertical pixel coordinate of the gaze intersection on screen.
ScreenGazeInfo.confidence	TrackingConfidence, str	A string value with the confidence of the measures made for gaze intersection: UNRELIABLE, LOW, MEDIUM, HIGH
TrackedPersonInfo.blink	BlinkInfo	A container of information related to blinking.

BlinkInfo.is_lost	bool	A boolean flag set to True when the person's blinks are not being tracked, set to False when they are being tracked.
BlinkInfo.left_eye_close	bool	A boolean flag set to True if the left eye is closed.
BlinkInfo.right_eye_close	bool	A boolean flag set to True if the right eye is closed.
BlinkInfo.confidence_left	TrackingConfidence, str	A string value with the confidence of the measures made for the left eye blink: UNRELIABLE, LOW, MEDIUM, HIGH
BlinkInfo.confidence_right	TrackingConfidence, str	A string value with the confidence of the measures made for the right eye blink: UNRELIABLE, LOW, MEDIUM, HIGH

Table 1 - Description of the fields exposed by the Network API, and accessible by external scripts.

The file has to be a .py file. Please make sure to follow guidelines before saving your script. For any further assistance, please contact us at support@eyeware.tech

Putting the files together & setting up the environment

After you've created your script, all that is left to do is putting everything together. This can easily be done by following the the steps below:

1. Create a `.bat/.sh` file with the following variables:

Windows Setup (.bat) file:

```
"""
  Copyright (c) 2020 Eyeware Tech SA http://www.eyeware.tech
"""

@echo ON
setlocal enabledelayedexpansion

rem Define Conda environment name
set env_name=demo

rem Clean previous Conda environment
conda.exe env remove -n %env_name% -y

rem Create Conda environment
conda.exe create -n %env_name% conda-forge::python=3.6 numpy -y

rem Activate Conda environment
call conda activate %env_name%
```

Linux Setup (.sh) file:

```
"""
  Copyright (c) 2020 Eyeware Tech SA http://www.eyeware.tech
"""
env_name=demo
# Clean previous Conda environment
conda env remove -n $env_name -y
# Create Conda environment
conda create -n $env_name conda-forge::python=3.6 numpy -y
# Activate Conda environment
conda activate $env_name
```

2. Open a Command Prompt or a Terminal
    a. Windows, click *Windows* button and type **`cmd`**
    b. Linux, type CTRL+ALT+T
3. Go to the folder containing the file created
    a. Windows: for example, **`cd c:\Users\User\Desktop\Script`**
    b. Linux: for example, **cd /home/[user]/Desktop/Script**
4. Execute the following file by typing its name in:
    a. Windows Command Prompt: **`setup_conda_env_demo.bat`**
    b. Linux Terminal: **`./setup_conda_env_demo.sh`**
        i. If you are using Linux, you may need to make the file executable. This can be done by using the command **`chmod +x setup_conda_env_demo.sh`** before running it.

Run the script

1. Run GazeSense. Enable the **External Communication** feature by checking the box in the application (figure 26).
2. After starting GazeSense, connect to your device by pressing the **Connect** button.
3. Go to the folder containing the demo files:
    a. Windows: open Windows Command Prompt, then for example **`cd c:\Users\User\Desktop\Script`**
    b. Linux: open terminal, then for example **cd /home/[user]/Desktop/Script**
4. In the same prompt, type: **`conda activate demo`** and hit enter.
5. In the same prompt, type: **`python [scriptname].py`** and hit enter.

17. Advanced Information about Units and 3D Geometry

This chapter will focus on explaining all the transformations done in GazeSense to define the spatialization of objects in a Setup.

Units and Coordinate System

The spatialization of an object in a tridimensional space can be described by rotation and a translation (position). An object, as defined within the scope of this technology, is a screen. That is, a plane with a given size and resolution. The units that we use are listed in Table 1.

Purpose	Type	Unit
Translation/Screen Size	Length	Meter (*m*)
Rotation	Angle	Radian (*rad*)
Screen Coordinate/Resolution	Measurement	Pixel (*px*)

Table 1 - Units used for the spatialization of all objects in a Setup.

We use a right-handed Cartesian coordinate system to describe the placement of all objects in space. We call this frame of reference the **World Coordinate System (WCS)**. In general, but not as a rule, models are often defined using the Y axis as the vertical axis, leaving the plane described by XZ (y=0) as a virtual floor (Figure 27).

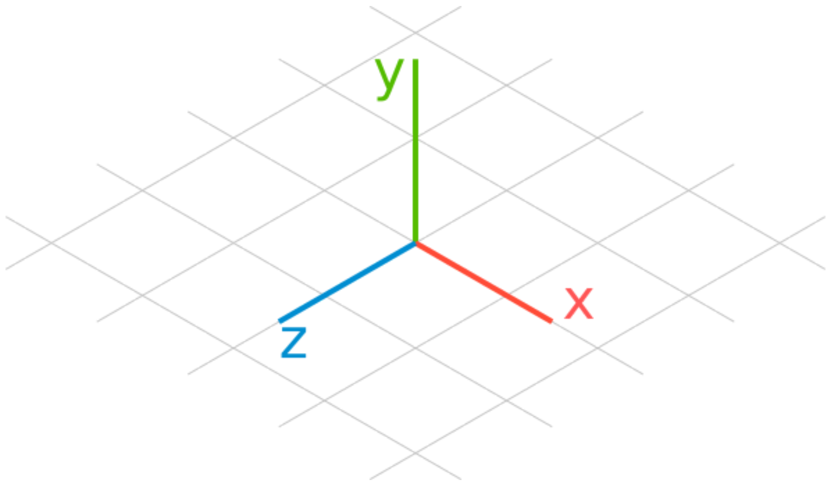


Figure 27 - An illustration of the right handed cartesian system used by Eyeware technology. Although not obligatory, models often use the XZ plane (y=0) as a virtual floor, and Y as the vertical axis.

There are two types of objects available: Camera and Screen(s). Both rely on a rotation and translation to be adequately characterized, but only a Screen object needs a size and a resolution in its definition.

Screen Object Definition

A Screen object requires several characteristics for its full characterization. It is represented by a rectangle whose origin is its center, and a physical size defined by a height in its vertical axis, and width on its horizontal axis (Figure 28).

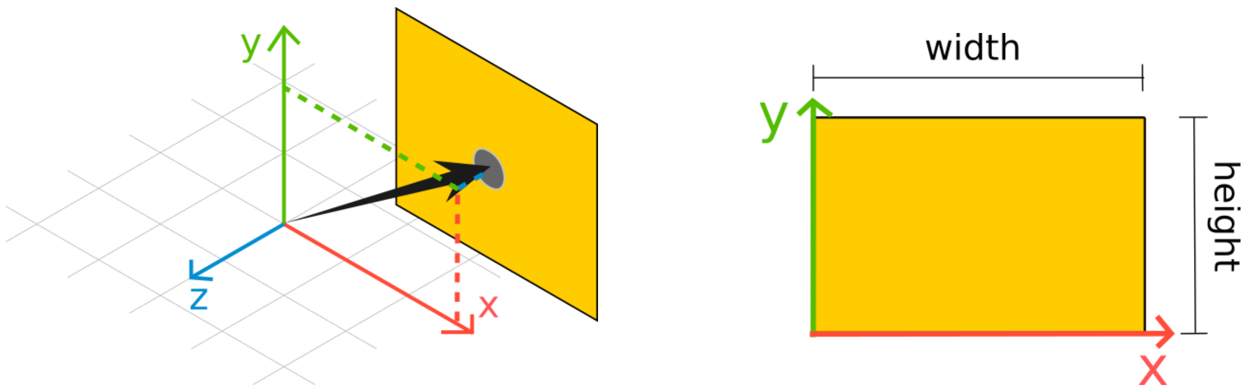


Figure 28 - **(left)** Operation of the translation of a Screen. **(right)** Size of a Screen (in pixels) as defined in GazeSense.

By providing a screen resolution, GazeSense automatically converts between physical units (width and height in meters) and digital units (horizontal and vertical pixels). What is important is that you input the correct information (size and resolution) corresponding to your physical setup.

Finally, the rotation of a Screen is centered around its origin. Afterwards, a translation is added. The operations in order are shown in Figure 29.

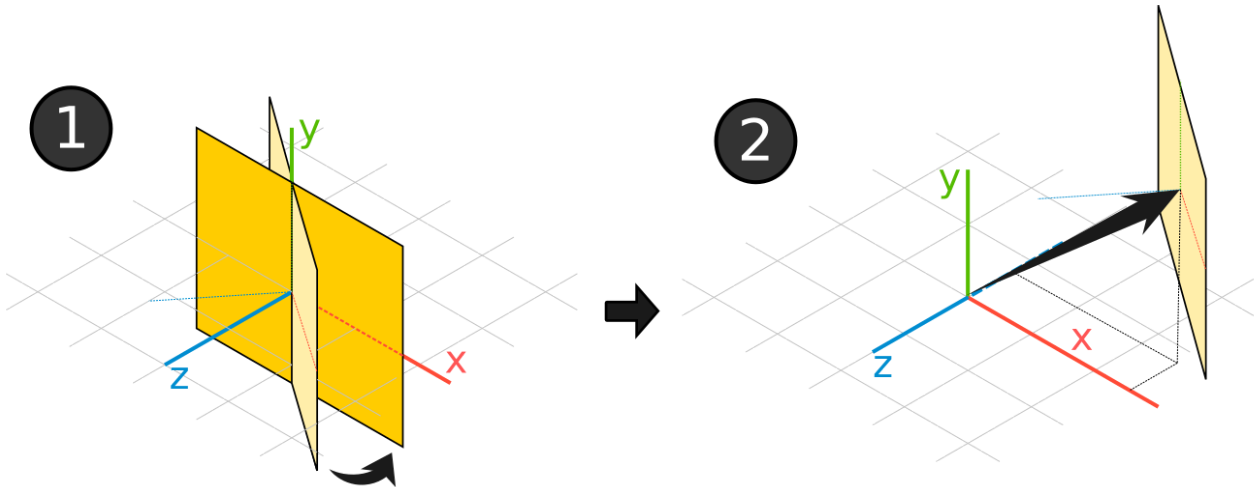


Figure 29 - (**left**) Illustration of the operations to place an object in the 3D environment. First, the Screen is rotated over its origin. (**right**) Afterwards, that origin is translated to its new location.

A Screen can be defined by its rotation, translation, size, and resolution. The methodology for its spatialization is not arbitrary. Switching steps or standards would result in different outputs.

Camera Object Definition

The placement of the camera in the 3D Setup is made in the same way as for a Screen. We start by rotating the camera over its origin, and only afterwards we translate its center to the new position (Figure 30).

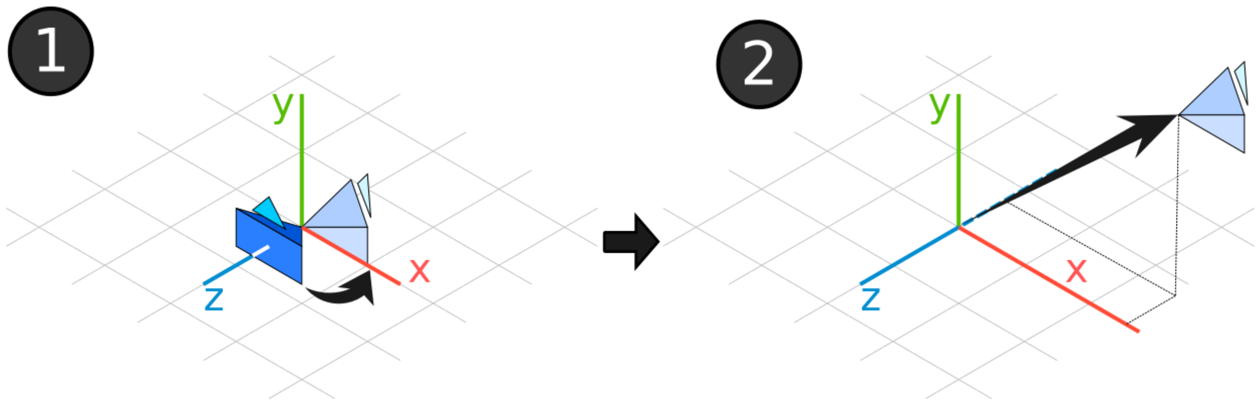


Figure 30 - Illustration of the steps to place a camera in the 3D environment. First the Camera is rotated over its origin (left), and only after that origin is translated to its new location (right).

Transformations, Conversions, and relation with GazeSense SDK

The camera model followed on GazeSense can be seen in Figure 31. We use the pinhole camera model[2].

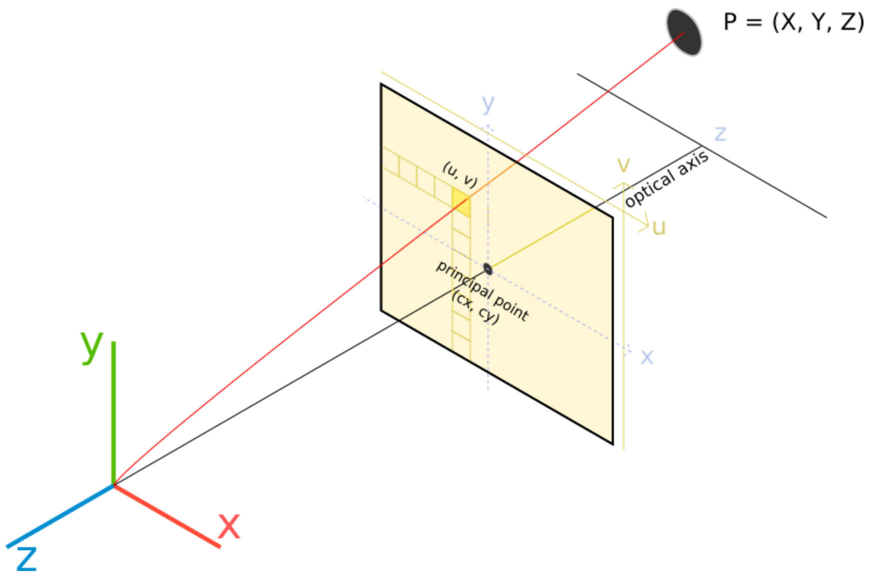


Figure 31 - Illustration of the camera model used by GazeSense.

Please consider that if you need to integrate Eyeware technology into your application without the GazeSense user interface, please contact us about obtaining GazeSense SDK, which is our product optimized for advanced integration with embedded systems, performance-critical applications, operating systems, etc. Having said that, Figure 32 illustrates the pinhole camera model used by GazeSense SDK.

[2] More information on the pinhole camera model can be found in https://en.wikipedia.org/wiki/Pinhole_camera_model and http://web.stanford.edu/class/cs231a/course_notes/01-camera-models.pdf

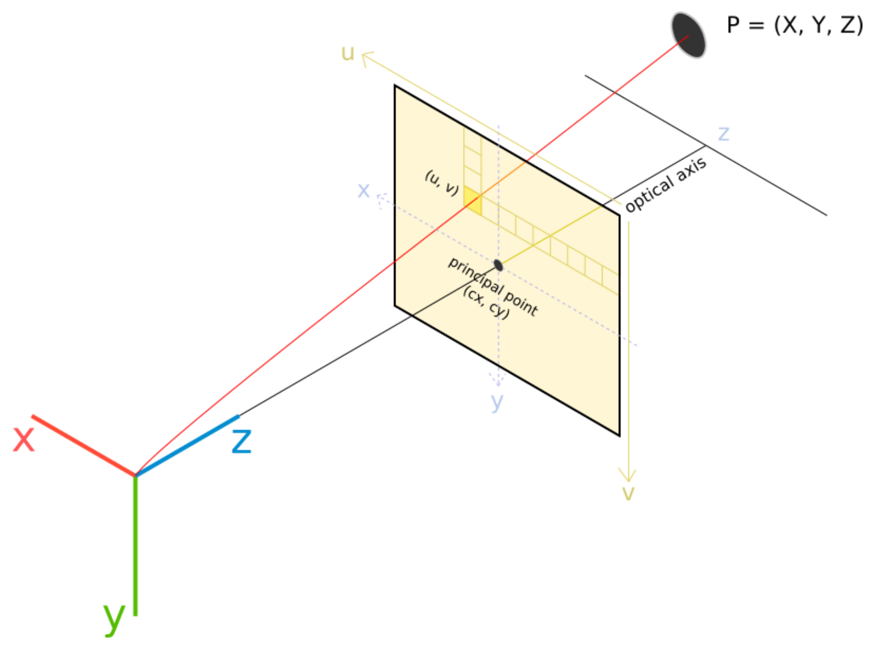


Figure 32 - Illustration of the pinhole camera model used by GazeSense SDK.

Regardless, the same logic applies. We can obtain the coordinates of any projected point in pixels by applying the following transformation:

$$s\ m' = A[R|t]M'$$

$$s \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} r_{11} & r_{12} & r_{13} & t_1 \\ r_{21} & r_{22} & r_{23} & t_2 \\ r_{31} & r_{32} & r_{33} & t_3 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}$$

The symbols have the following meanings:

- **s** is the scale factor to be used if a camera is intrinsically using a scaled image.
- **(u, v)** is the projected pixel coordinate of a point in the setup.
- **A** is the camera matrix of intrinsic parameters.
- **(cx, cy)** is the central, or principal, point coordinate.
- **(fx, fy)** is the focal length.
- **R** is the rotation matrix for a point in 3D space.
- **T** is the translation vector for a point in 3D space.
- **M'** is the vector representation of any arbitrary point in 3D space in normal coordinates.

18. Known Issues

1. In Windows, switching on the Face Mesh Visibility in the viewer causes performance drops for lower end computers.
2. Resolution for 4K monitors can sometimes cause UI issues for Ubuntu users. This is due to Ubuntu not being optimized for 4K resolutions.
3. Some users may experience connectivity issues with the Orbbec Astra device on Ubuntu. If that happens, after each session the user must unplug and plug the device back in, then connect.
4. Some returning users may experience license activation issues. How to troubleshoot & solve:
    a. Open a command prompt in the installation folder
    b. Run `license-manager --deactivate`
    c. Run `license-manager --activate`
    d. Enter your license key

19. Frequently Asked Questions

2D vs 3D Eye Tracking

What is the difference between 2D and 3D eye tracking?
The key problem with 2D based approaches is that they poorly address head pose variations. If the participant moves nearer or further from a 2D sensor, the software has to guess where in the 3d space the participant is, thereby delivering unreliable calibrations and lower accuracy.

General Questions

What are the key features of GazeSense?
We use consumer grade 3D cameras to track a user's gaze & head pose. By using a single sensor, GazeSense can provide a high degree of accuracy with a wider degree of head rotation for many use cases.

What components are required?

1. GazeSense App or SDK from Eyeware
2. Computer
    a. Windows 10 or Linux Ubuntu 16.04 LTS or higher
    b. 6th Generation Intel® Core™ Processor or higher
    c. USB 3.0
    d. 2GB RAM & 1 GB of free hard drive space

3. Multiple camera options (purchased separately)
    a. Intel RealSense D415 Recommended for most applications
    b. Orbbec Astra Embedded S
    c. Other 2D & 3D cameras are possible. Contact us for more information.

What is output I can expect to receive?

1. Our output is data only, via a CSV Export or API.
2. General data available: Head position (6DoF), attention towards surfaces, point of regard. For more information please refer to page 30 of our User Guide.
3. We do not currently offer data visualization or analytics features but can suggest the following third party solutions:
    a. Tableau, Ogama, Direct communication via our API

What is the difference between GazeSense & SDK

GazeSense is simply a user interface that runs on our core technology, available in our SDK.

What is included in GazeSense SDK?

1. The C++ SDK consists primarily of C headers that can be used on C or C++ programs.
2. The software package also shows Code Samples. C++ samples are provided for showing practical examples of how to use the GazeSense SDK. We provide pre-compiled binaries, as well as compilation instructions to get the code samples up and running.

Getting Started

Once I have the required components, how do I get started?

Please see our onboarding video or email contact@eyeware.tech

Do you offer free trials?

While we do not offer free trials, we have tried to make evaluations easier by offering 1 month evaluation licenses.

How do I download GazeSense App or SDK

Please send a request to contact@eyeware.tech and the software/ license key will be emailed to you.

Technical Specifications

What is the accuracy?

1. 1°- 3° error for screen-based applications with calibration

2. 7°- 10° error for uncalibrated human-machine interaction

What is the range
Generally, 0.2 - 1.5m, but longer distances are possible.

What frame can GazeSense achieve?
5 - 60fps, with restrictions that apply depending on use case.

Do you support multiple cameras?
To use multiple cameras, please contact sales for the SDK. The SDK can provide processed data from an arbitrary number of devices as long as you have the processing power available. We recommend the Intel RealSense D415 models, as they are designed to avoid interference in a multi camera setup.

Can GazeSense track multiple participants at once?
GazeSense App currently does not support multiple cameras. However, this can be developed with our SDK.

Do you provide pupillometry?
We do not provide pupil tracking due to lower resolution of the sensors.

How do I use the API?
GazeSense can be used together with third-party software by using a Network API. This is done with a Python script that listens to "tracking events" coming from GazeSense. For each event, the script does something with the tracking data, such as printing the geometric values in a terminal, or playing a sound, etc. This Network API is thus a valuable and simple way of building powerful, interactive demos with a few lines of Python code. Full guidance is available in the User Guide (page 30).

Copyright Notice