VisioChick: Smart Glasses for assessing poultry housing conditions – Instruction Manual Including setup of a virtual barn tour

v. 2024-3

Dorian Baltzer, M.Sc., University of Bonn, IGG

September 24, 2024

Introduction

The general aim of the VisioChick project is to provide new methods and technologies for knowledge transfer in animal husbandry and management with specific focus on animal welfare. Since visual perception in the barn is particularly important for chicken and humans see in a different way, VisioChick provides an approach to explore a barn from the animal perspective by simulating the animal view for the user. To achieve this, a head-mounted display (HMD) with Augmented Reality (AR) technology was developed in a collaborative project between Prof. Jan-Henrik Haunert, Dorian Baltzer and Shannon Douglas from the University of Bonn, Prof. Inga Tiemann and Dr. Falko Kaufmann from the University of Osnabrück, Prof. Youness Dehbi from the University of Hamburg and the industry partner uReality who took over major parts of the technical realization. The device is referred to as "Smart Glasses" in this manual. While the HMD allows to explore the barn on-site with additional impressions, also methods are needed to provide these insights for the use outside the barn, e.g. in teaching. Therefore, the setup of a virtual tour through a chicken barn is explained in the second part of this manual.

Contents

1 Overview of the Smart Glasses

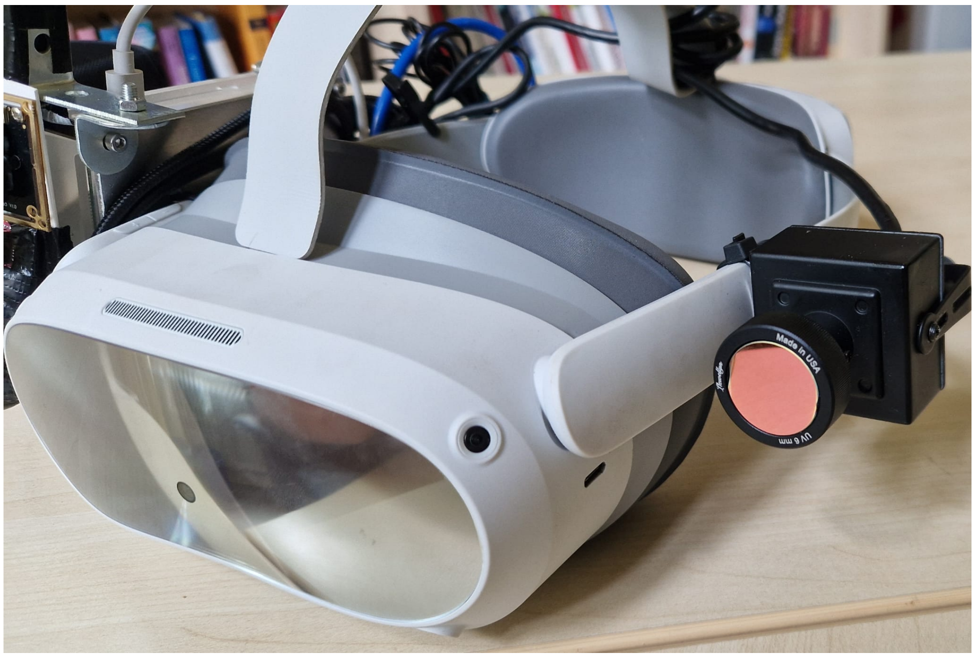


Figure 1: Picture of the Smart Glasses prototype

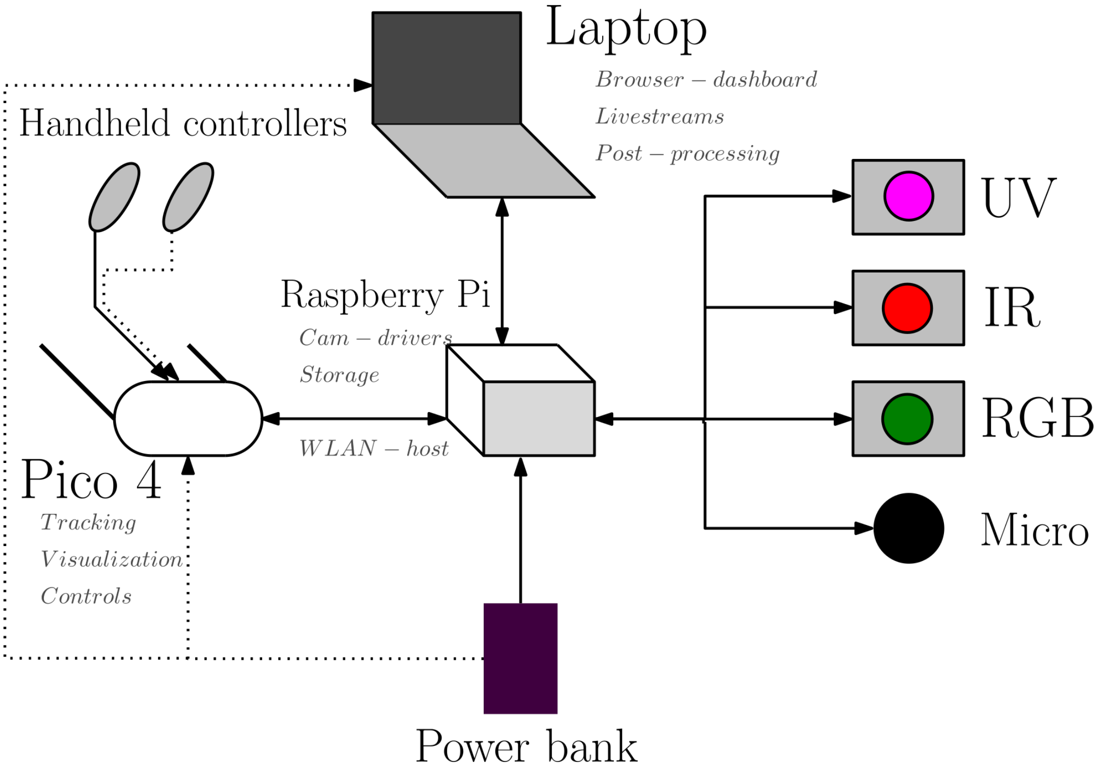


Figure 2: Schematic overview including supplementary components

1.1 Technical specifications

1.1.1 VR Glasses

The HMD device is based on a Pico 4 Enterprise VR device which is commonly available on the consumer market. The HMD without additions has a weight of 591 g. It uses two 2.56" displays with a resolution of 4320x2160 pixels, a frame rate of 90 Hz and a field of view of 105°. The internal inside-out-tracking is based on four mono-fisheye-cameras which use distinctive objects in the environment to locate its position in a local coordinate system. The processing unit is a Qualcomm XR2 octacore with up to 2.84 GHz. The device has 8 GB of DDR5 RAM and 256 GB of internal storage. The internal battery with a capacity of 5300 mAh runs for up to 3 hours, the two handheld controllers run on AA batteries with roughly 80 hours time of use. The HMD is charged via a USB-C connector and can also be used while connected to an external power supplier. The Enterprise version offers additional developer features but is based on the same hardware as the standard Pico 4. The 16 MP front camera can also be used to view and record a see-through video stream of the real environment.

1.1.2 Cameras

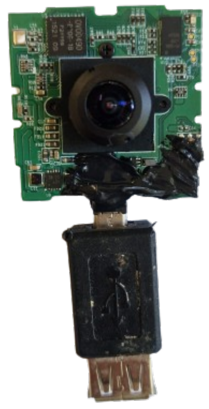


Figure 3: 360 Degree Camera

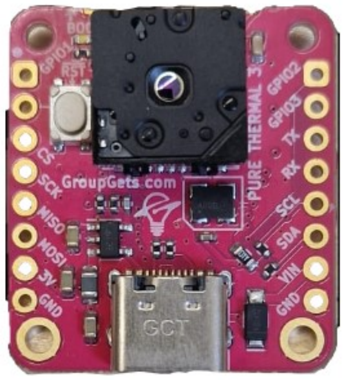


Figure 4: Thermal Infrared Camera

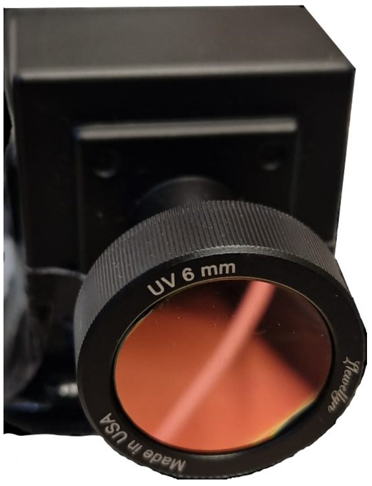


Figure 5: Ultra Violet Camera

UV Camera:
Model: XNiteUSB8M-MNCUVL (5, 6)
Features: resolution of 8 MP (3264x2448 @ 15 fps), Sony IMX179 sensor (limit around 300 nm), Llewellyn Optics Silica lens 6mm F/2.8 (can detect light below 200nm), UV shortpass filter XNite 330C, IR block filter XNiteBP1, total system light range 320-380 nm, UVC interface, compatible with Raspberry Pi, power consumption: 150-240 mA @ 5V DC, price 1950$ + shipping and taxes
Challenges: heaviest camera in the system, resolution needs to be reduced for a 30 fps stream.

Thermal IR Camera:
Model: FLIR Lepton 3 (4, 7
Features: Thermal sensitivity of 50mK, super compact and lightweight design, priced at

300 euros
Challenges: Low resolution of 160x120 pixels, does not come with a built-in UVC interface
Supplementary component: GroupGets PureThermal 3: Compact circuit board with connector for the Lepton 3 camera and USB-C for UVC output. Contains a STM32F412 ARM microprocessor for onboard processing of the Lepton 3 output. Allows running custom drivers to view the thermal image on different devices and operating systems including Linux.

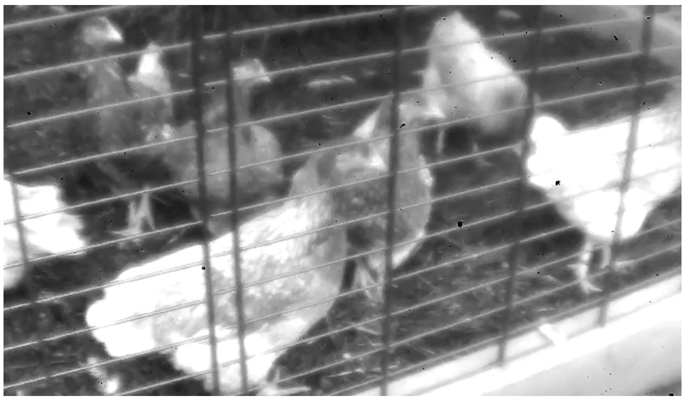


Figure 6: UV Camera Image

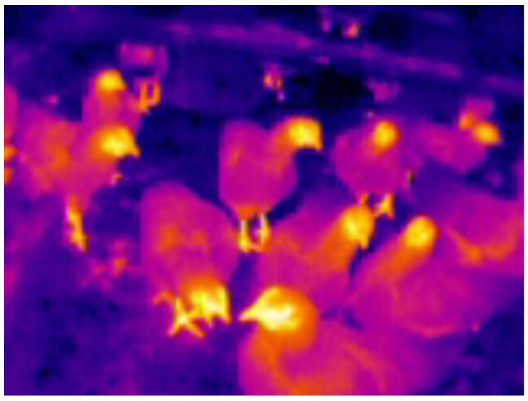
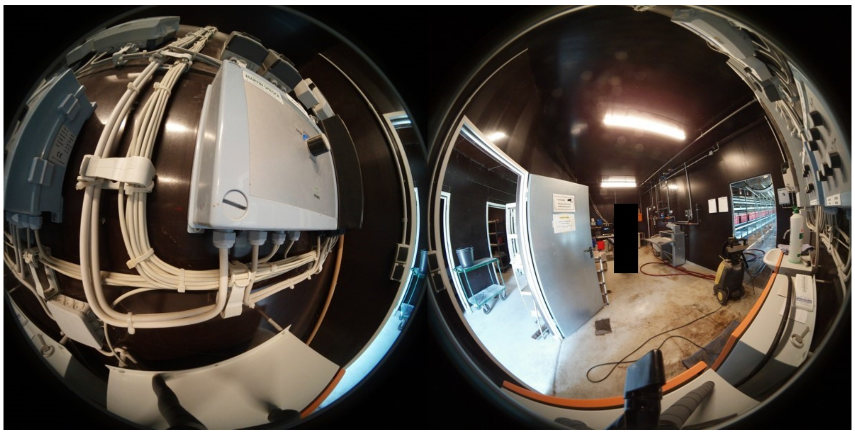


Figure 7: Thermal Infrared Camera Image

RGB Camera:
Bulk 360-degree camera (3, 8) containing two slightly overlapping wide-angle lenses on both sides of a circuit board
Features: UVC interface, compact and light, adjustable focus, priced at 250 euros
Challenges: Protrusions, fisheye distortion, no automatic image stiching


[redacted]
Figure 8: 360 Degree Camera Image

1.1.3 Additional hardware

In addition to the three cameras, also a microphone is connected to record audio files. All sensors are controlled by a Raspberry Pi 4 mini computer which runs the hardware drivers, records and saves the streams, prepares the datasets for export and hosts a local network from

where the dashboard can be accessed by the HMD. The Pi has a Broadcom BCM2711 Quad core Cortex-A72 (ARM v8) 64-bit SoC @ 1.8GHz, 4 GB of DDR4 RAM, 4 USB ports, MicroSD slot and is powered by 5V DC (min. 3A) via USB-C connector.
For the system to work, all electric components need to be supplied with power. While the HMD itself has a built-in battery which can be charged, the Raspberry Pi needs external power. To solve this, a portable power bank is being used. To handle the weight efficiently, it can be stored in a small bag attached to the belt. The Raspberry Pi is then connected by cable which needs to be handled with caution since the stream recording will be lost if the cable gets disconnected by accident. All cameras are provided with power via USB cable from the Raspberry Pi (9). If the HMD battery runs empty in the barn, also the HMD can be connected to the power bank to continue recording. The saved datasets are stored on an SD memory card in the Raspberry Pi.

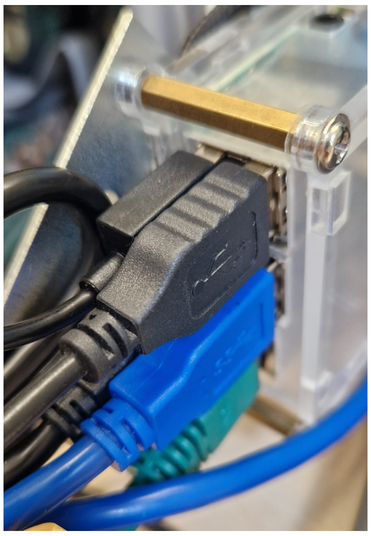


Figure 9: USB Connectors at the back of the Pi

1.2 Software Stack on the Smart Glasses

1.2.1 VR Glasses Application

The application running on the Pico 4 VR device was developed in Unreal Engine and Android Studio. For developing apps running on the Pico 4, the Pico XR plugin is required. This plugin works best with Unreal version 4.27. For Unreal version 4.27, the Android NDK version 21.4.7075529, API level 29 and the corresponding SDK version are recommended.
The app is mainly responsible for binding buttons of the Pico controllers to functionalities and for accessing the internal camera-based tracking and see-through video stream. The cameras are not directly managed by the app, so generally no changes are needed to make here. Instead the app opens a simple browser on startup viewing the html dashboard of a local server backend running on the Raspberry Pi which is wirelessly connected. The only output data generated on the Pico is a JSON file containing the tracking information. The file is saved on the internal storage of the Pico as a backup, but the data is also transferred to the backend server on the Raspberry Pi via HTTP GET request.

1.2.2 Pico 4 Local Server

The local server running on the Raspberry Pi is based on Socket.io for NodeJS and WebRTC. The Raspberry Pi is configured to automatically run a local server on startup and provide an HTML dashboard with all functionalities. The codebase is structured as follows:

1. Server.js: Defines sockets and their functions (see below).
2. Broadcast.js: Defines the WebRTC connection handling, control of the different cameras and recording functionality. Constantly running on Pi as "broadcaster" with GUI (Broadcast.html) showing current video streams.
3. Watch.js: Running on the client device. Responsible for the WebRTC connection to the client, controls device switching commands and the choice of icons. Communicates to the broadcaster via sockets.

The GUI consists of the following three modules:

1. Broadcast.html: Local stream view on the Raspberry Pi. Can be used as a viewer for bugfixing and making changes on the Pi. The stream that is running here is also running on the client device. The transmission is done by sending single frames.
2. Dashboard.html: Desktop version of the Dashboard GUI. Views live camera stream, offers the option to switch between cameras and record (without tracking). Contrary to the VR version also offers the option to download and delete content.
3. Dashboard_vr.html: VR version of the Dashboard which is accessed by the Pico 4, optimized for See-Through stream and VR controllers. Does not offer the option to download or delete content but includes icon selection. Can also be accessed by a desktop browser for testing or development purposes.

The following sockets are implemented in Server.js:

1. Launch Dashboard: http://localhost:4000/dashboard.html
2. Start broadcaster: http://localhost:4000/broadcast.html (Broadcaster gets started in the browser automatically)
3. Start watcher: http://localhost:4000
4. Switch to RGB: http://localhost:4000/rgb_view
5. Switch to IR: http://localhost:4000/ir_view
6. Switch to UV: http://localhost:4000/uv_view
7. Next available video device (cycles through all of them): http://localhost:4000/next
8. Capture frame (invoked from client, saved on server): POST requests to '/save-frame' http://localhost:4000/save-frame
9. Download (all saved content of the last session as a ZIP file): http://localhost:4000/download
10. Clear content: http://localhost:4000/clear-content

1.2.3 Specific handling of the Thermal Camera

Since the Thermal Infrared camera FLIR Lepton 3 using the PureThermal 3 board is not directly compatible with the system allthough it has a UVC interface, a workaround needs to be implemented. The camera supports 5 pixel formats with 4 of them showing a thermal image in a way the user would expect it (e.g. a colorscale with red representing high temperature and blue representing low temperature). The fifth format is called Y16 and contains a grayscale image organized as 16 bit integer values for each pixel stored from left to right and top to bottom. This format can't be processed by the PureThermal board so that the resulting stream is constantly black. In the end it depends on the hosting device which output format is selected, so that some applications show a black image while others show the correct thermal image. Testing it with different devices and applications showed that some web browsers use the Y16 format, among others Chromium which is the standard browser for Rasbian (The Raspberry Pi operating system) and used in the broadcaster service transmitting the stream to the VR glasses. However, the camera works with Mozilla Firefox which is also available on the Pi. Therefore, an additional script called `broadcaster.sh` is introduced running `broadcast.html` in Firefox instead of Chromium. The script needs to be set as executable and put into the autostart configuration of the Pi. When everything is configured correctly, the browser window showing `broadcast.html` should appear shortly after booting the Pi without additional user inputs so that the Pico can be connected and view the stream as shown in 10.

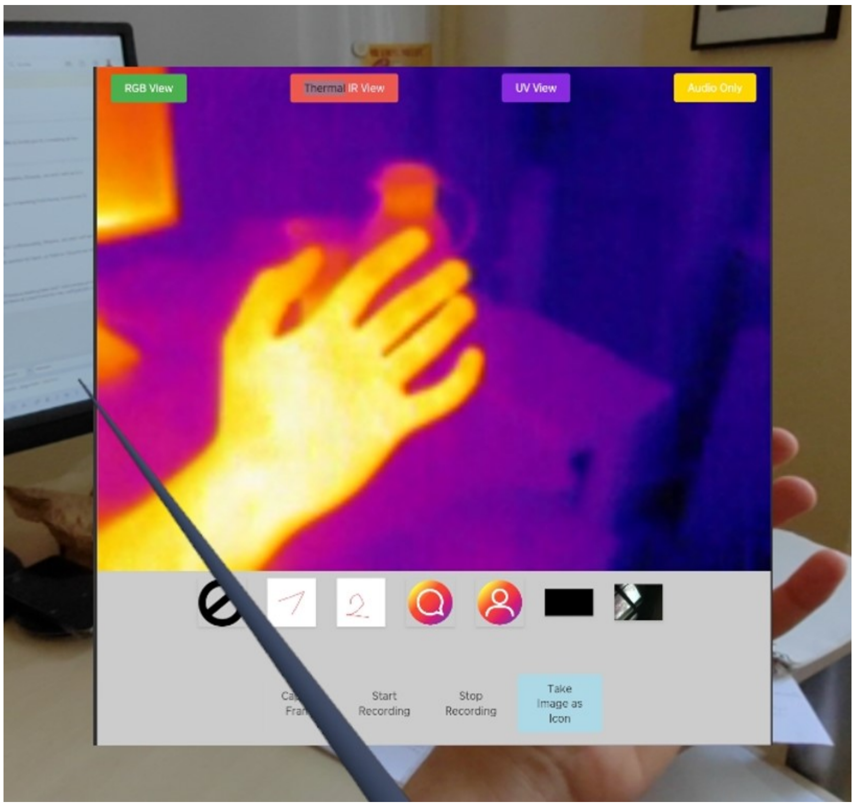


Figure 10: Thermal image using colorscale from blue to yellow, live view in VR Application.

Another aspect of the Thermal Camera to consider is that it consits of a camera module in form of a black box sized $1 \times 1cm$ connected to a pink board (cf. 4). From time to time or

in case of errors, it should be checked whether the camera is still correctly connected to the board. If the camera is loose, there will be no image stream transmitted. Additionally, the board has two physical buttons: Reset (front side) and Boot (back side). These buttons are not needed in usual operation, they are mainly important for editing the camera firmware. However, in case of errors the camera can be rebooted by using these buttons. In case of errors it should also be precluded that the buttons are pressed by accident, e.g. by attached cables or mountings.
The board also contains a small light indicating its current operation mode. In regular operation, the light blinks once a second indicating that it is ready for recording or it blinks very fast indicating that it is currently recording. If the light is off, the board is either not connected to a source of electric current (the Pi) or it is not booted. A continuos dimmed light indicates an additional mode which is used for accessing the firmware by the connected device. This mode should not be used with the Smart Glasses.

2 Using the Smart Glasses

2.1 App Start-up

To start the system, the following steps need to be taken in advance:

1. Check all cable connections.

2. Connect Raspberry Pi to power bank or other source of electric current. The Pi will boot automatically.

3. Boot the Pico by pressing the power button on the side of the glasses.

4. Take care that the Pico will connect to the WLAN network "chicken2". If it does not connect on its own, manually connect to the network by entering the password "chicken23".

5. As soon as the Pico is connected to the Pi, the app can be started. The Pico 4 is based on Android and the app can be found in a menu simmilar to many other Android devices such as smartphones. Upon running the app, a video stream of the actual environment should be visible along with an overlay showing the VR Dashboard (see below). If the Dashboard is shown correctly, the connection was established successfully and the recording session can be started.

> If there is an error shown on App-startup and the Dashboard overlay does not appear, check the WLAN connection and if the Pi is running.

2.2 VR Dashboard

To view and record streams from the different sensors within the smart glasses, a communication between glasses and Raspberry Pi is necessary. To avoid the use of too many cables, the connection was set up via a WLAN network hosted by the Pi. Commonly the smart glasses are already connected automatically as soon as the Pi is turned on and the network is running. If the network settings were changed, select "chicken2" in the network list to reconnect. The main dashboard is displayed in 11. At the first row, the camera or microphone input can be selected. The buttons are colored differently according to their function to make the interface as intuitive as possible. Below the live view there is a scroll bar containing icons which can be selected by clicking on them. Once an item is selected, it is highlighted by a frame. The next recorded media file will get a reference to the respective icon afterwards. Under the icons, there are additional buttons for capturing images from the live view and starting or stopping the recording. The button Take image as icon captures an image of the current live view just like "Capture Frame" but additionally it adds the new image to the icon list. By this procedure it is also possible to add real previews to the saved media as icons.

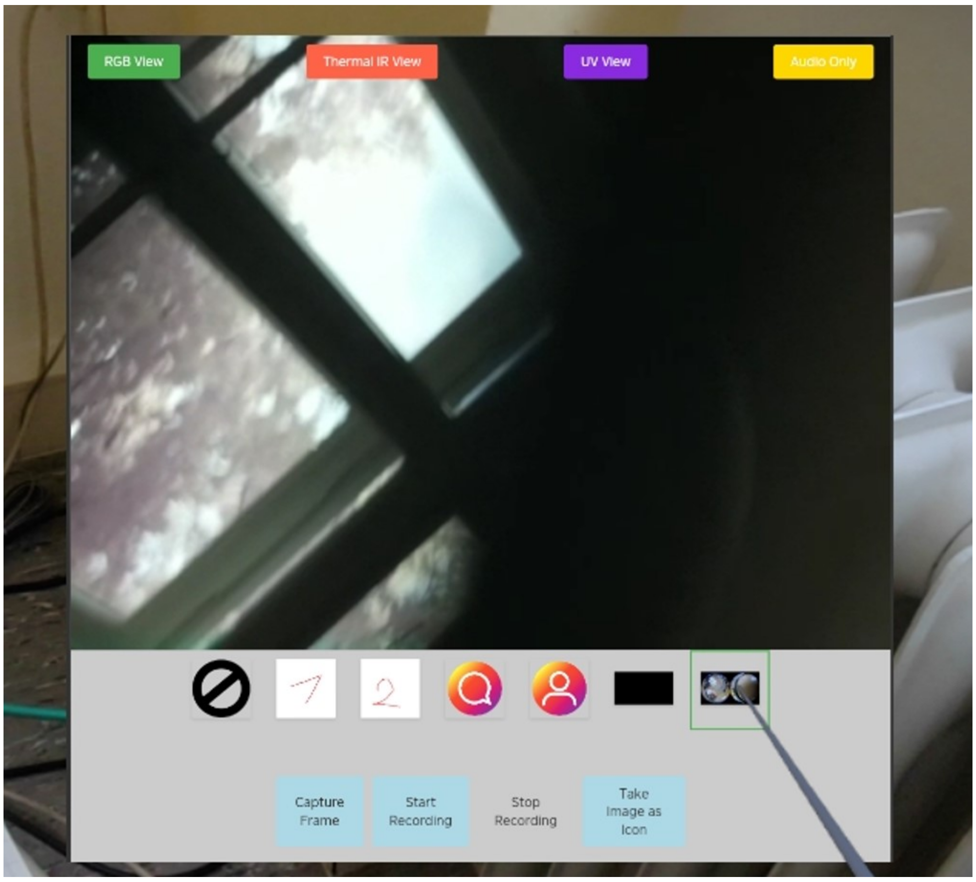


Figure 11: VR Dashboard

The dashboard view can also be changed to view of only the stream without controls by pressing B once or faded out by pressing again.

Always select an icon before starting to record.

2.3 Controls

The controls for interacting with the Dashboard within the VR app were made as simple as possible. Allthough the Pico 4 has two controllers, only the right one is needed. As 12 shows, there are only three important buttons:

1. Confirm: This button can be most easily triggered by the forefinger since it is on the back side of the controller. It is the most needed button since it is used to click virtual buttons on the Dashboard, e.g. starting to record, switching cameras or selecting icons.

2. Switch view: This button changes the view mode between three variants: a) Dashboard with buttons and camera view (is set on startup), b) Live camera view in fullscreen without controls and c) No dashboard, only see-through video of the real environment.
3. Exit: Double clicking this button will exit the application.

The other buttons are generally not important. The button on the right side allows to take screenshots. Since an option to take images with the different cameras is included in the Dashboard, this is commonly not needed. However, if a screenshot of the Dashboard is wanted e.g. for presentation purposes, this might be helpful as well. It is also possible to take images of the real environment with the built in front RGB camera by fading out the Dashboard first by pressing A twice and then taking the screenshot.

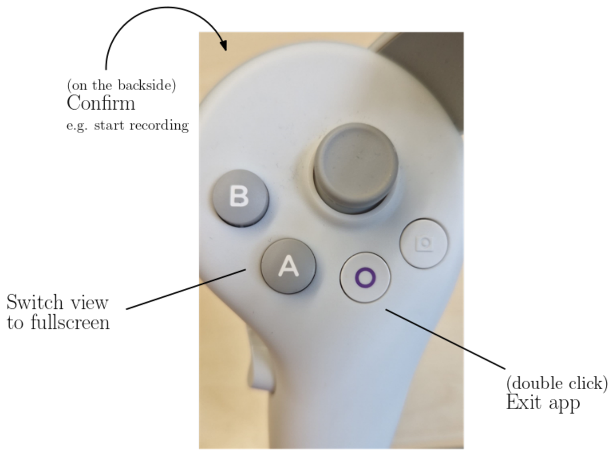


Figure 12: Controls binded to the right handed Pico 4 controller

2.4 Desktop Dashboard

The graphical user interface running on the Raspberry Pi can not only be accessed by the Pico VR glasses but also by a different device like a desktop computer, laptop or tablet. The device just needs to be connected to the WLAN network "chicken2" to allow for wireless communication between the Raspberry Pi and the accessing device. The Dashboard generally offers similar options like the VR Dashboard including a live view of all cameras and the possibility to start and stop recordings or capture images. Contrary to the VR version, there are the additional buttons Clear Saved Folder and Download Session Content (13). The first one deletes all recordings and tracking data from the file system. This is recommended to do after the data was exported (described in the following section) and before starting a new recording session. The other button creates a zip archive with all recorded data and starts a download to the client file system.

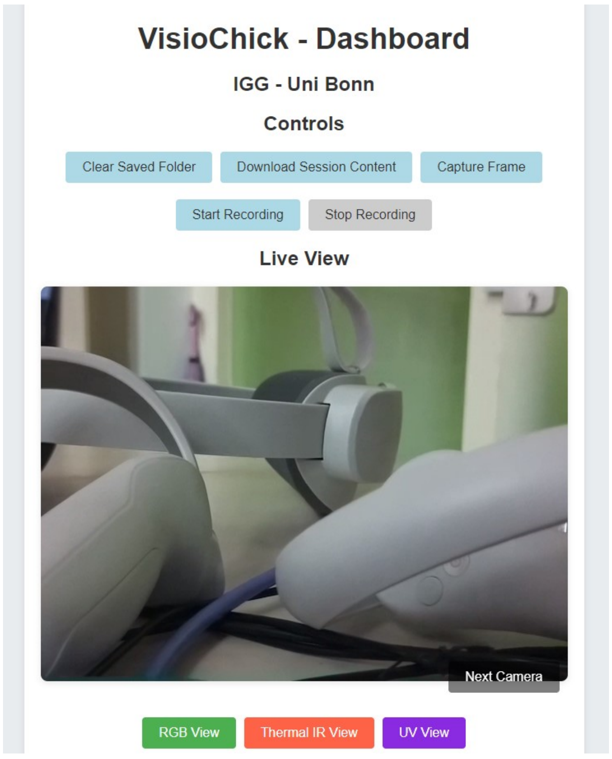


Figure 13: Desktop Dashboard

2.5 Exporting Data

The Raspberry Pi always stores the latest acquired dataset in a way that it is accessible from the desktop dashboard for download. Older datasets are still stored but cannot directly be accessed via the dashboard. Therefore it makes sense to export the data directly after every recording session. To do that, a device with a web browser such as a laptop needs to be connected to the Raspberry Pi WLAN network "chicken2". Then the dashboard needs to be accessed as explained in the last section.

Data should be exported to an external device (e.g. laptop, tablet) after each recording session.

3 Overview of the Virtual Tour Platform

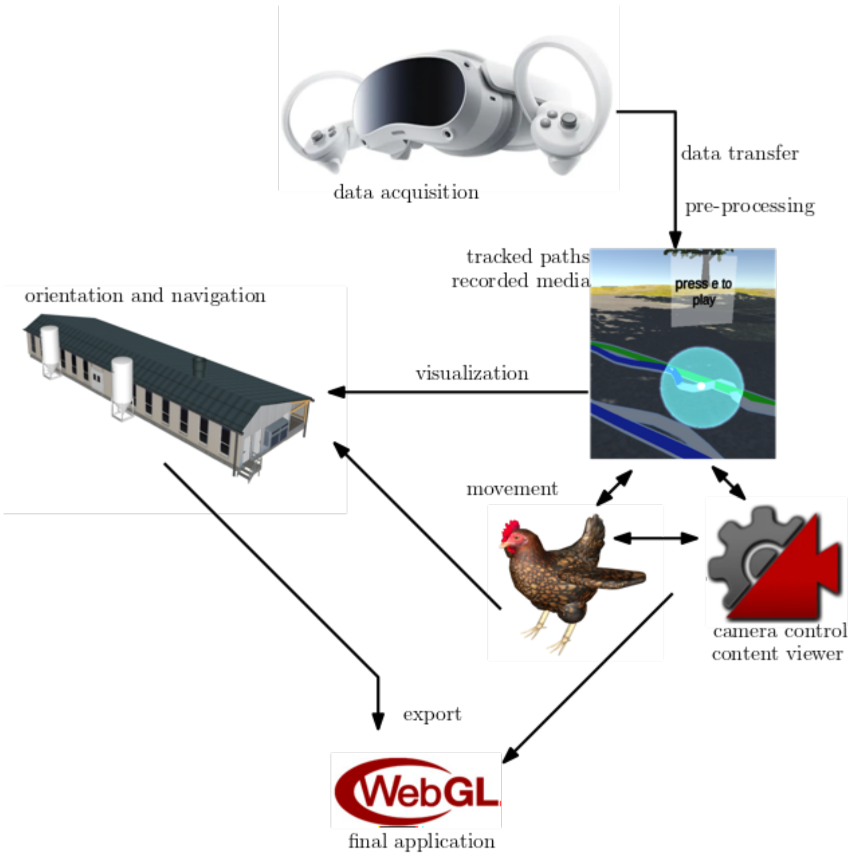


Figure 14: Overview of the virtual tour platform

The Virtual Tour Platform consists of six main components (cf. 14). To create a virtual environment/map a 3D model is needed. Typically this should be a replica of a real barnyard or the environment where the glasses are supposed to be applied. More detailed information about the 3D model are given in the following section.
Another data source of high importance next to the 3D model is the recorded dataset from the smart glasses. Exporting data after a recording session is explained in the previous chapter. The whole folder of recordings can be integrated in Unity by a few simple steps as shown in 4.3. Subsequently, the 3D model and recordings are merged by coordinate transformations so that the tracked paths show up in the place they where originally recorded. The accuracy depends on the 3D model and the real environment but can be expected in the range of a few dm. The paths are then automatically visualized in different colors according to their type of content.
For moving within the virtual environment in the virtual tour, a virtual animated chicken is integrated in the package. The controls are similar to common third person role playing games. During movement and for switching between free movement and viewing recordings, different virtual cameras are responsible. There is one camera following the chicken during third person movement, one camera following it in bird's eye perspective and a third camera viewing media content.
Finally, for exporting the virtual tour application from Unity to a standalone format, WebGL is employed. WebGL is a JavaScript API for 3D graphics in web browsers. It enables

the opportunity of distributing the application online on a webserver.

4 Setup of a Virtual Barn Tour

In this section, all components and steps needed to compile a virtual barn tour are outlined. The general idea is to use a 3D model of the barn as a basis for integrating data from the smart glasses. Most of the used concepts should be seen as suggestions while alternative software products might be suitable as well.

4.1 3D Model of the Barn

To visualize buildings, objects and landscape in virtual space, it is an appropriate way to create artificial 3D models of the real objects. There are different ways of producing 3D models and the most eligible one depends on the application and the desired level of detail. If the highest possible level of detail and geometric accuracy is needed, the use of terrestric laser scans is recommended. Therefore, the scanner needs to be placed at several viewpoints to cover every view that is needed and creates a point cloud for every viewpoint. In the next step, all point clouds are connected to one by a registration procedure with redundant data being eliminated. Finally, the point cloud can be transformed to a mesh by different methods and software products.
This technology is expensive and not available for everyone, so the focus is set on consumer-grade hardware and methods. As a rule of thumb one should keep in mind, the less effort is being put in taking accurate measurements and data acquisition, the less realistic becomes the result. Simple measurements at the inside and outside of buildings e.g. for the length and height of walls can be taken by laser distance meters. Smaller objects can be measured by sliding calipers and tape measures. All measurements on-site should be documented in a way that they can correctly be assigned afterwards. Also everything supposed to be modelled in virtual space should be photographed at good lighting conditions and in high resolution since the images can be used to create textures.
Once all measurements are taken, the work can be continued on a computer. A good start to create simple 3D models is Sketchup which offers a free version with limited features. If a ground plan of the respective building is available, it can be imported and easily converted to 3D space. If not, the groundplan should be drawn first based on the measurements and then the height is added in the second step. Photos can be imported and used as textures for every object, but also standard textures are available for download as well as basic 3D objects like furniture, windows or plants. Another free software for 3D modelling is Blender which also allows to create animations. The result in both software products heavily depends on the effort. While a simple model of one building can be created within a few hours without previous experience, one could also spend weeks or months to create a detailed virtual environment. An additional option to create 3D models especially of small objects is camera-based modelling software which is also available in form of mobile apps nowadays. The resulting quality can not be compared with industrial grade laser scanners but it is still an option worth of trying for fast and low-cost modelling.

4.2 Integration in a Game Engine

Once the 3D models of the barn and related objects are prepared, they somehow need to be made interactable for the user. One way which is possible without a lot of programming experience is the use of game engines since these offer lots of pre-scripted modules for interaction. The most common engines nowadays are Unity and Unreal Engine. While both would be convenient in a similar way, Unity is being used in this guideline.
To get started in Unity, an empty 3D project using the Universal Render Pipeline (URP) can be created. Unity is based on so-called Game Objects which are listed in the Hierarchy menu. Every Game Object is a virtual object with different attributes and appearances. A 3D model of a building can be a Game Object but also a virtual camera which views the building would be a Game Object. The attributes of Game Objects can be view in the Inspector window. Typical attributes could be gravity, collision or visible materials (including color).

4.3 Importing Data from the Smart Glasses

Downloading data from the smart glasses delivers a zip archive including one full recording session. If you do not know how to download the data, please read sections 2.3 and 2.4. For processing the data in Unity, the archive needs to be unzipped to a common folder. A typical example of the folder structure is shown in 15.

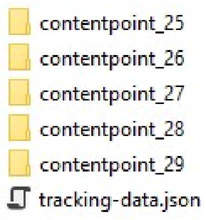


Figure 15: Folder structure of a dataset

Since the video recordings are done in the WebM format by the Pi and Unity does not support this codec, they need to be converted to a common codec such as mp4 first. Use a video converting software and put the videos in the same place with the same name and ending afterwards. To include the data in Unity, open the project window and put it into Assets » Resources per drag & drop similar to 16.

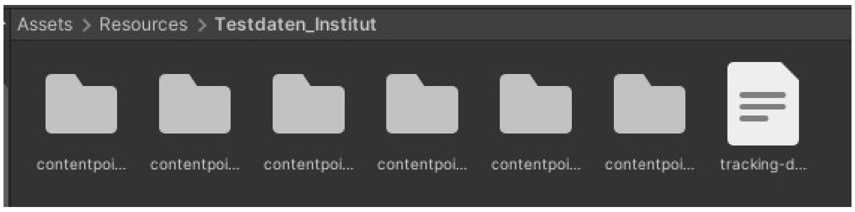


Figure 16: Data in Unity

Next, head to the Hierarchy window and open the Inspector of the GameObject DataPathFactory. This contains an attribute called drag & drop json file. Put the JSON file from the Project window here and click the button Create Data Path below to transform the data into Game Objects. The dataset will be included as a child element of DataPathFactory and the sub paths (e.g. related to one video file) are child elements of the recording session. Double clicking on the recording session Game Object in the Hierarchy moves the Scene view to the object. By using the Move, Rotate and Scale tools in the Scene, the paths can be adjusted so that they fit in the 3D model. In the Scene view, the trajectories should look similar to the example in 17

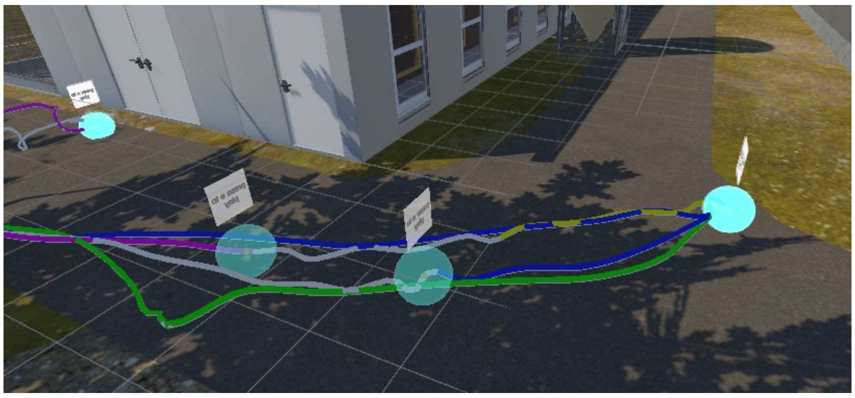


Figure 17: Visual tracking paths in Unity

All paths will be colored automatically depending on the type of content. For changing the colors, open Assets ≫ Materials in the Project window and edit the attributes of the Materials in the Inspector window.

The data folder needs to be unzipped before the integration in Unity.

4.4 Adding Interaction

The basic set of interactions between the user-controlled chicken and the trajectories and media is already implemented and consists of the following modules:

- View media content: When approaching a point of interest such as the starting point of a video, a preview label is displayed saying "press e to play". By pressing this button, the game mode switches from free movement to the media viewer in fullscreen.
- Close media content: When in media view, the mode can be switched back to free navigation by pressing q.
- Switch between content at one POI: When in free movement mode and close to a POI with multiple media content, the focussed content can be switched by pressing q.
- Open doors: When close to doors, they can be opened or closed by pressing e. They also open automatically so that the chicken can move through doors on forced movement along a path.
- Media viewer: Viewing videos with related tracking paths will automatically move the chicken along the respective trajectory. The rotation and movement speed result from smoothing the tracking point data by Beziér curves.

If additional interaction is desired, e.g. by objects in the environment, please refer to common Unity guidelines.

> All required interaction scripts are already included in the toolbox.

4.5 Exporting the Tour Application

As soon as the Unity project is finished and the virtual tour works as desired, it can be exported to a standalone format so that users don't need Unity to view the tour. Unity supports different platforms including Android and iOS for smartphones. The most simple way for viewing the tour on a Desktop computer is to create an executable file (.exe) which runs locally. This can be done for Windows, Mac or Linux. Generally this supports all standard settings and results in the highest possible quality. To create a local executable file, go to File > Build settings and select Windows, Mac, Linux as a platform. Then choose the Target Platform on the right hand side and go ahead with building the application by the button on the lower end.
However, it might be too much effort for the provider as well as the users to distribute the educational tour as an executable file. There is also the option to view the tour online in a browser (cf. 18) based on WebGL which is a JavaScript interface for 3D graphics. In the Build Settings window, choose WebGL as a platform to proceed. Since WebGL does not support every feature of a local executable file, a few changes need to be made:

1. Depending on the server hosting the application, the compression format might need to be changed. If the right setting is unknown, deactivate the compression by heading to Player Settings > Publishing Settings > Compression Format.

2. Also the Color Space needs to be adjusted for WebGL applications. Go to Player Settings > Other Settings > Color Space and choose Linear.

3. There are a lot more quality settings which are not mandatory but might help in improving the performance especially on older systems. For instance it is possible to lower the visual complexity of lighting and shadows to decrease computational effort without disturbing the essential content of the tour.

4. It is also possible to adjust the resolution/size of the application window within the browser by going to Player Settings > Resolution and Presentation > Default Canvas Width/Height. An option to switch into fullscreen mode will be automatically included without changing any settings.

One crucial difference between an executable file and a browser application is that the executable file can easily contain all resources on the local file system while the browser application is supposed to be uploaded to a server. When exporting a WebGL application, Unity does not include large data resources such as the recorded media from the smart glasses. Therefore, it is required to upload the data to the server along with the exported files. On the server, go to the folder hierarchy level where the URL should be accessed by the user (e.g. www.example-university.com/virtualtour). In this folder, put the data exported from Unity. In the common case this should be an index.html, a folder called TemplateData (if the default Unity template was used) and a folder called Build containing the actual application. On the same level, put the folder containing the smart glasses recordings the same way it was included in the Unity project before, e.g. Assets > Resources > testdata. By this procedure, it is guaranteed that the automatically created URLs for each media file will still work on the server.

▌ If the application is supposed to run on a server, upload the smart glasses data as well.

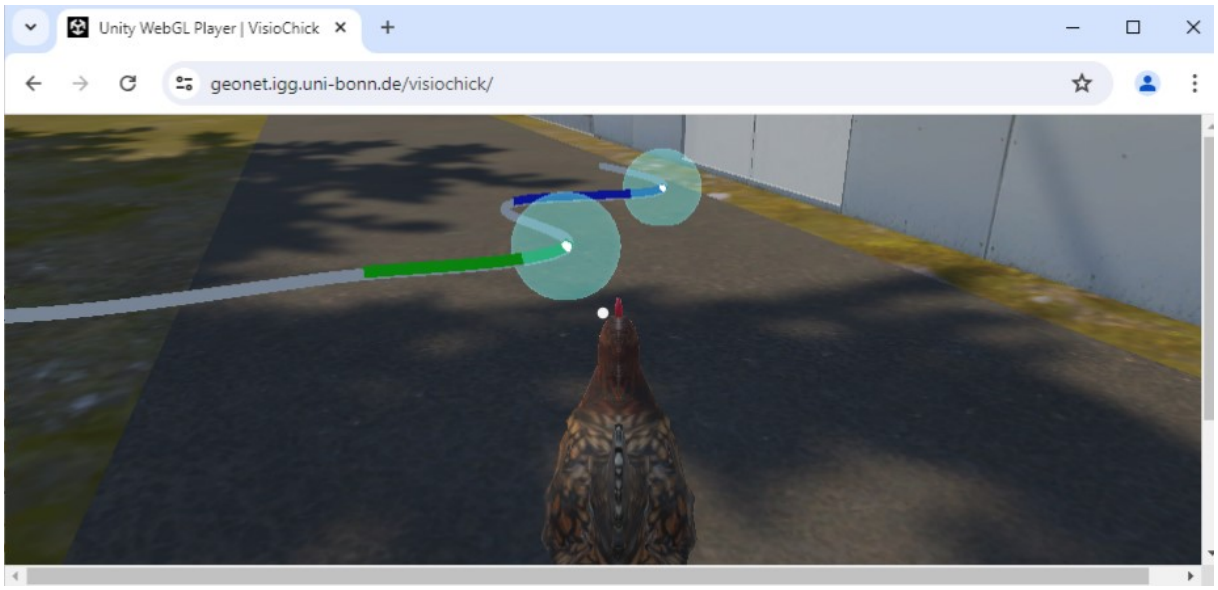

Unity WebGL Player | VisioChick
geonet.igg.uni-bonn.de/visiochick/

Figure 18: Final application in a web browser

5 Appendix

5.1 Prerequisites to run the Dashboard App on the Smart Glasses

For the device to successfully work, you must ensure that:

1. The glasses (Pico 4) are connected to the Raspberry Pi via WLAN (network name: chicken2).
2. The Raspberry Pi is provided with power and turned on.
3. The "virtual walls" are turned off on the Pico 4 glasses (developer settings).
4. You have selected an icon before starting to record a stream.

5.2 Prerequisites to create the virtual tour application

1. Unity version: Using different version might lead to incompatibilities. Consider using version 2021.3.25f1 if any errors occur.
2. Webserver: You should have a Webserver prepared in order to upload the final application in case you want to distribute it online. Refer to standard guidelines for running a Webserver since this is not part of this document.
3. Color space: When switching between local executable files and WebGL as export format, the color space might need to be changed from Linear to Gamma. Try this change in case errors occur when running your WebGL application.

5.3 Frequently asked questions

How to install Unity

Visit the Unity Hub Download Page: https://unity.com/de/download (19.07.2024)
Click on the "Download Unity Hub" button to download the installer. Run the downloaded installer and follow the on-screen instructions to install Unity Hub on your PC. Launch Unity Hub after the installation is complete. Sign in with your existing Unity account or create a new one if you don't have an account.
Add a New Unity Version: In Unity Hub, go to the "Installs" tab. Click the "Add" button to add a new Unity version. Choose the Unity version you want to install from the list. Click "Next" to proceed.
Select Modules: Choose additional modules you might need, such as support for Android build support, iOS build support, or other platforms/tools you require. Click "Next" to continue.
Start Installation: Review the components to be installed and click "Done" to start the installation process. Wait for the installation to complete.
Create a New Project: If not already open, launch Unity Hub. Go to the "Projects" tab. Click the "New" button to create a new project. Choose a project template (e.g., 3D, 2D, etc.) based on your project requirements. Enter a name for your project and choose the save location. Click the "Create" button to create your new Unity project. Unity will open, and you can start working on your project.

Additional notes:
Ensure your PC meets the minimum system requirements for running Unity.
Unity Hub allows you to manage multiple versions of Unity and switch between them as needed.
You can install additional components and modules anytime via Unity Hub by selecting an installed version and clicking "Add Modules".

Which Unity version should I use?

This project was developed in Unity 2021.3.25f1. To secure compatibility of the toolbox without any changes, the use of the same version is recommended.

Is the used software in this guideline free?

Yes, everything included in this guideline can be done by free software. However, in some cases the free versions are limited in time or functionality and can be upgraded to paid retail versions.

Can I create a livestream from the barn?

Yes, this is possible with additional soft- and hardware. One possibility is to connect a Laptop to the Raspberry Pi via WLAN and access the browser dashboard (see section 2.3). The Laptop then also needs an internet connection, either via a second WLAN connection, cable or mobile internet such as a surfstick. A simple approach would then be to stream to the browser window e.g. via screensharing in video conference software. There might also be apps for the Pico 4 allowing to stream from the glasses, however the Pico does not allow for 2 WLAN connections simultaneously to our knowledge.

Is it possible to record several streams at once?

So far, the option to record multiple streams at a time is not implemented. To our knowledge, the hardware capacities of the Raspberry Pi would reach their limits here. If this is wanted, probably a hardware upgrade is needed in the first stage.

Can I integrate different cameras in the device?

Yes, the cameras can be replaced. Main requirement for the cameras is a UVC interface to run the camera via USB without additional drivers. After replacing a camera, the assignment needs to be changed in the code. The Pi has 4 USB slots meaning that the total number of simultaneously connected cameras is limited. It might be possible to increase the number of cameras by an additional USB hub or by connecting one camera to the CSI connector directly on the mainboard. However it should be considered that the total weight of the HMD is already quite high with three cameras and the mandatory components.

6 Authorship, Copyright and Funding

The content in this guideline emerged from a collaborative research project involving Prof. Jan-Henrik Haunert, Dorian Baltzer and Shannon Douglas from the University of Bonn, Prof. Inga Tiemann and Dr. Falko Kaufmann from the University of Osnabrück, Prof. Youness Dehbi from the University of Hamburg and the industry partner uReality who took over major parts of the technical realization. We are also grateful to Felix Köcher who greatly helped in 3D modelling and technical development in Unity. The guideline itself was written by Dorian Baltzer and reviewed by the project partners. All images were taken or created in the University of Bonn, either at the Institute of Geodesy and Geoinformation, Meckenheimer Allee 172, Bonn or at Versuchsgut Frankenforst 4, 53639 Königswinter. This work (Model- and Demonstration Project for animal welfare) is financially supported by the Federal Ministry of Food and Agriculture (BMEL) based on a decision of the Parliament of the Federal Republic of Germany, granted by the Federal Office for Agriculture and Food (BLE); grant number 2820MDT160.