https://doi.org/10.2352/ISSN.2470-1173.2018.03.ERVR-451


# Virtual Reality for Sensor Data Visualization and Analysis

***Artur Baltabayev[1,*], Alexej Gluschkow[1,*], Johannes Blank[1], Gero Birkhölzer[1], Jean Büsche[1], Martin Kern[1], Fabian Klopfer[1], Lisa-Maria Mayer[1], Gabriel Scheibler[1], Karsten Klein[1,2], Falk Schreiber[1,2], Björn Sommer[1,2,$]***
***[1] University of Konstanz, Konstanz, Germany***
***[2] Monash University, Melbourne, Australia***
***[$] bjoern@CELLmicrocosmos.org***
***[*] contributed equally***

## Abstract

*The visualization and analysis of sensor data is an important aspect of geovisualization, and Virtual Reality-related technology is increasingly used in this context. Here we present the results of a student project which supports users in recording data (such as environment temperature, humidity and light intensity) with a Texas Instruments SensorTag® and subsequently visualizing the data on a smartphone using mobile Virtual Reality technology. The project resulted in two different applications, which both employ a 2D component for data localization and recording, as well as a 3D component to explore the data on a smartphone. In this paper, we describe our results and summarize our experiences regarding the creation and usage of such software and mobile VR environments.*

## Introduction

### *Sensor Data for Immersive Movement Analysis*

Sensor technology is an essential element in various research fields such as disaster management, crowd movement prediction, crime prevention and the analysis of animal behavior [1]. Especially the analysis of movement is an important aspect of the aforementioned topics which often relies on location sensors [2].

Tracking data based on *GPS (Global Positioning System)* sensors is often combined with additional sensor data, such as temperature [3], or it is used to compute additional features, such as acceleration or wind prediction [4]. Although in the past often the longitude and latitude of GPS data were analyzed exclusively, new approaches also take the altitude into account [5]. When *three-dimensional (3D)* coordinates are available, valuable technologies for their visualization and exploration are *Stereoscopic 3D (S3D)* or *Virtual Reality (VR)* technologies.

Forty years after Sutherland developed the first *Head-mounted display (HMD)*, devices like Oculus Rift® or HTC Vive® are nowadays available to consumers. Especially the cheap HMD versions, the VR cardboards – compatible to most recent smartphones – are used by many people worldwide [6]. These immersive technologies are often also used for analytical purposes, which is referred to as *Immersive Analytics (IA)* [7, 8], and example applications include health-related analysis supporting, e. g., cancer research [9, 10], educational bio visualization purposes and analytics of microscopic structures [11, 12].

Another advantage of the commercial success of HMDs is the availability of development environments. A popular 3D development framework is Unity® which enables a fast and easy access to the HMD technology. However, it should be noted that Unity® is proprietary, the source code is only partially available, and it comes with costs as well as platform limitations [13]. Alternative approaches like JavaScript API WebVR provide easy access to HMD-related as well as web technologies [14, 15]. WebVR has the advantage that it is compatible to a number of recent web browsers and there exist compatible frameworks, such as three.js, which lowers the entry level for new developers [16]. WebVR as well as three.js are based on the WebGL specification, a standard providing access to a subset of OpenGL commands for all recent browsers [20]. For the visualization of position-related data in 3D one of the most popular platforms to date is Google Earth® which also recently added support for HMDs [17]. As an example, we used this platform to create a prototype to communicate the effects of human behavior onto the state of the great barrier reef [18]. A comparison of different S3D and VR platforms and their usage in the context of bird movement analysis can be found in [19].

### *Immersive Sensor Data Visualization - A Student Project*

As part of the 2017 software project course at the University of Konstanz two student groups had the task to visualize recorded data of an external consumer sensor device on a smartphone by using mobile VR technology. The requirements for both project groups were the usage of

1. an external low-prized TI SensorTag®,
2. Android as development platform,
3. a (cheap) VR Cardboard, and
4. WebGL for VR exploration.

For the resulting concepts and software of the two groups, we will discuss in the following how they

- connect a Bluetooth sensor tag,
- record the data,
- cope with potential problems during the recording of 3D coordinates,
- present the data in VR cardboard, and
- transfer the data from a 2D app (Android) to the VR visualization (Website with WebGL)

Finally, we will discuss application areas.

In both projects, a TI SensorTag communicates via Bluetooth with a single smartphone. The smartphone records the data and saves it in a text-based format which can be inspected in the *two-dimensional (2D)* view and GUI. The 3D component can then be used to explore the data on a smartphone. In case a Google

Cardboard-compatible smartphone is connected, the data can be explored in VR with simple head tracking to look around and a joystick-supporting movement.

Two prototypes were developed in the context of the student project: SensorTagVR and Visensor/VR. While both projects started with the same basic target, their development took a quite different direction:

- **SensorTagVR** The idea of the SensorTagVR project was to allow the user to record data of a specific environment and later explore the data in a 3D scene via VR technology. The virtual room should represent a real world place and the data collection was optimized during the project for *indoor usage*, via a WiFi network trilateration. The visualization was done by constructing a 3D height landscape from the data and showing how the temperature differs in a big room, in our case the sports hall of the university. A potential application case is the measurement of problematic temperatures inside buildings and to identify significant spots.
- **Visensor/VR** The idea of the Visensor/VR project was to focus on the measurement and comparison of luminance. Whereas the SensorTagVR uses a predefined area, Visensor/VR focuses on the automatic generation of abstract 3D landscape where the measurements were done using GPS data for *outdoor usage*. A potential application case is in assisting growing plants by determining which area has the most optimal light level for the respective plant.

## Technology

The basic technologies used for both projects are quite similar. Here, we provide an overview of the used hardware and software, as well as a short description of the differences between both projects (apps).

### *Hardware*

Recent smartphones were used in combination with a VR cardboard and a small sensor device. Figure 1 shows the different hardware components used in both projects.


***Figure 1.*** *Hardware used in the projects: VR Box (top), smartphone (Samsung Galaxy S6, left), Texas Instruments SensorTag CC2650 (bottom). SensorTagVR: VR Sensorpark (center), Visensor/VR: XBOX controller (right).*

Currently, the only compatible sensor for both apps is the Texas Instruments SimpleLink™ Bluetooth low energy/Multi-standard SensorTag CC2650STK [21]. Different sensors are part of the SensorTag which can be used to collect data including the ambient as well as object temperature, humidity, pressure, acceleration and light intensity. Any other Bluetooth-equipped sensor that provides similar data could be used by adapting the interface to the sensor.

The VR cardboard headset "Box VR" was used. As the VR scene is shown using a smartphone in side-by-side stereoscopic view, any other VR cardboard device should be compatible, such as the Google Cardboard® or HTC GearVR® in Cardboard mode.

**SensorTagVR** For controlling the SensorTagVR view in 3D space, a Bluetooth Controller called "VR-Sensorpark" (which comes with the VR Box) was used, but a number of alternative Bluetooth controllers are compatible as well. For position indoor tracking, four Raspberry Pis were used; outside of the building, *GPRS (General Packet Radio Service)* could be used (however, in the indoor application case here, GPRS was not applicable). The difference between GPRS and GPS is that GPRS will provide a position relative to the local cellular network, whereas GPS will provide the global position in longitude and latitude.

**Visensor/VR** The Visensor/VR app was used on a Samsung Galaxy S6® smartphone which provides a GPS sensor as well as a barometer. Whereas the GPS is used to store the 2D positions – longitude and latitude – the barometer is relevant to record the altitude with higher precision than GPS. Therefore, a smartphone using Visensor/VR should be equipped with both technologies. The virtual reality mode runs with controller support, in our application an XBOX 360® or XBOX One® gamepad connected to the smartphone via a USB adapter.

### *Software*

For SensorTagVR and Visensor/VR Android versions 5.0 to 7.0 is required to run the app on the smartphone. Google Chrome is required to run the virtual reality components. The connection between smartphone and sensor is done with Bluetooth Low Energy technology. For both projects, Android 5, JavaSript and WebGL were used during the implementation and tests.

**SensorTagVR** For the position tracking this project used four Raspberry Pis to function as WiFi hotspots for triangulation. For this purpose, OpenWRT was used which is an open source Linux-compatible project used in combination with embedded devices to route network traffic [22]. This software was running on the Raspberry Pis, transforming these devices into WiFi hotspots.

## Methods & Implementation

Although both project started within a similar setup and similar basic requirements, each project took a different direction. Here, we shortly describe the different methods and implementations, including the class overview, the data management as well as the position tracking.

## SensorTagVR Implementation

The documentation as well as the source code of SensorTagVR is available from: `http://sensortagvr-git.immersive-analytics.org`

### Class Overview

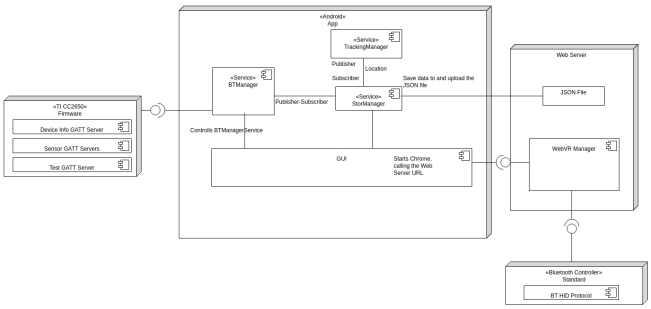


***Figure 2.*** *SensorTagVR: composition diagram*

SensorTagVR is subdivided into two parts (see Fig. 2):

- The mobile app. Its main goal is to connect to an external sensor device via Bluetooth, to process and save the data collected by the sensor, and to pair it with the positional data from the triangulation.
- A web application. In order to explore the saved data in 3D space, the web application generates a virtual reality scene and the stored data is visualized.

These two parts are connected in such a way that the user can open a browser with the according web application from within the app. The upload of the data works by just tapping a button.

In the following we will explain some classes and methods that were used in this app.

**Main Activity** Allows to enter the data recording, have a live data view of the sensor, set the information to connect to a sensor and setup of the WiFi hotspots.

**StorageManager** Gathers the data from the sensor and the position from the TrackingManager via broadcast listeners and combines them into a JSON file, which can be uploaded by the user.

**TrackingManager** Determines the current position by trilateration of the connected WiFi hotspots. Also contains the settings for the hotspots to increase the accuracy of the tracking.

**BTManager** Collects the data from the sensor and broadcasts it inside the smartphone via the Bluetooth connection. The StorageManager then setups a Broadcastlistener to receive the data from the sensor.

**WebVR Manager** The WebVR Manager is located outside of the app on a web server. The spatial model is loaded and the data (which is uploaded from the app) is displayed.

### Data Management

A JSON file containing the recorded data is temporally saved on the smartphone and then uploaded to a web server, after the user finished recording the complete room. Afterwards, the data points are saved in conjunction with a 3D model of the sports hall of the University of Konstanz. This 3D model was manually modeled and textured with the open source 3D modeling tool Blender. The model is stored as an OBJ file on the web server.

### Position Tracking

To track the smartphone position, if possible GPRS and indoors the WiFi hotspots were used. Via trilateration over the signal strength with a few WiFi hotspots the indoor smartphone position was recorded. Fig. 3 illustrates the trilateration process: each point represents WiFi hotspot, the circles depict the range of the hotspot signals up to the red point in the center, representing the smartphone whose position has to be determined. Inside the red circle, the different circular range-representing gray circles intersect. In this way it was possible to track the position indoors with a precision of up to 5 m depending on the WiFi hotspots distance. The trilateration calculation was performed with the following Java library [23]: `https://github.com/lemmingapex/trilateration`

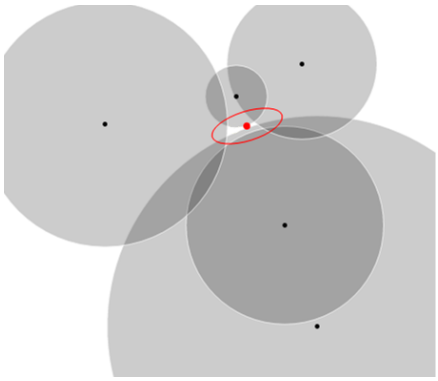

***Figure 3.*** *SensorTagVR: Trilateration*

### Virtual Reality

The VR component of SensorTagVR is based on two frameworks. The WebVR framework was used to acquire the smartphone's orientation. The 3D world was implemented using the three.js library which can be found at: `https://threejs.org`

Three.js was used to load the OBJ model, to construct the plane and to define the camera. The camera orientation, field of view and transformation was provided by WebVR. To calculate the plane from different data points, the inverse distance weighting method was applied, in order to compute the height of all corner points of the plane depending on the recorded data [24].

## Visensor/VR: Implementation

The documentation as well as the source code of Visensor/VR is available from two sources. The Android app (Visensor) for the smartphone is located at `http://visensor-git.immersive-analytics.org` and the WebGL component for the web browser (Visensor VR) can be found at `http://`

visensorvr-git.immersive-analytics.org.

### *Class Overview*

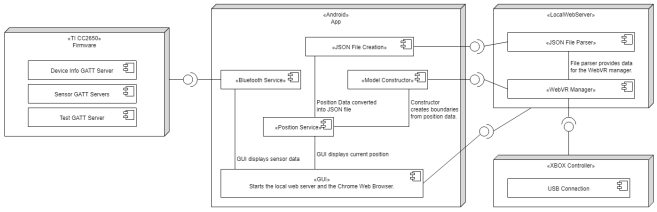


***Figure 4.*** *Visensor/VR: composition diagram*

Figure 4 shows the implemented classes. The classes Service, AppCompatActivity and ArrayAdapter belong to the Android API. The source code of all classes can be found at the app's GitHub repository.

In the following section relevant classes and methods are discussed that were used in Visensor/VR.

**Main Activity** Contains a list that displays the sensor data, a record button to save the sensor data, a disconnect button and a button to enter the virtual reality mode.

**BluetoothConnectionList** Displays the available Bluetooth Low Energy devices that were found by the smartphone.

**PositionService** Determines the current position with the GPS and barometer from the smartphone. This service is started whenever a bluetooth device is selected from the list.

**BluetoothService** Collects the data from the sensor. This service gets started whenever a bluetooth devices is selected from the list.

**Data saving** Whenever the recording is started the app will create a plain text file in a separate folder on the smartphone. The sensor data is then saved during runtime along with the position data in a JSON file format. The file will be closed once the record button is pressed again to stop the recording.

**Data visualization** The data points are visualized at their respective positions as colored spheres in a virtual room. The user can choose between a logarithmic or linear scaling for the spheres.

**Model constructor** The model constructor takes the sensor data along with the positional data and creates a *virtual landscape patch*, representing a rough approximation of the terrain the data was recorded on. Additionally the outermost points are used to create a visual fence. Since the virtual landscape patch is bigger than the actual terrain used for recording, the fence helps to determine the exact recording area in the visualization. The model construction is based on Paul Bourke's triangulation Java code [25] at `http://paulbourke.net/papers/triangulate/triangulate.java`.

### *Data Management*

The JSON file is saved with the date of recording, i. e., JJJJ-MM-dd-hh-mm-ss.json. The virtual landscape patch is created from an OBJ file that the Model Constructor creates. The object file is saved along with the JSON file in a separate folder on the smartphone.

### *Position Tracking*

To determine the current position, we use conventional GPS technology, which is included in most modern smartphones. However, the raw GPS data is very inaccurate and thus we employed further methods to determine the position as accurately as possible. We will discuss those methods in the following section.

**Altitude measurement with barometer and Kalman filter** To obtain a more accurate model of the environment where sensor data was recorded a special measurement of the altitude is required. This is also a major difference between both projects, as SensorTagVR does not require the height information, but Visensor/VR does need it. While GPS can already offer a resolution of 1 m for the altitude this precision is too low for Visensor/VR's purposes. Therefore, the barometer was utilized that is nowadays built into many smartphones.

However, the data provided by the barometer is very noisy. The chosen solution for Visensor/VR is a Kalman filter, a simple but powerful method of reducing statistical noise in a series of measurements over time. The basic idea is to combine the measured values with a prediction of the next measured value based on a linear model. This leads to a final estimation with a much lower error than the original measurement.

Since the Kalman filter is only applied to one measurement, and there are no other inputs that have an influence on the measured value, Visensor/VR uses the Kalman filter in its simplest form. This means that the matrices that the algorithm usually uses are all $1 \times 1$ matrices, and parameters that take external influences into account can be set to 0. For the prediction of the next measurement value we assume that the measured value stays constant. This means that the linear model is the identity function. Equations 1 through 5 describe the algorithm of the resulting Kalman filter. The equations are calculated in the following order for each measurement at step $k$. The output of the Kalman filter is $\hat{x}_k$.

$$\hat{x}_k = \hat{x}_{k-1} \tag{1}$$
$$P_k = P_{k-1} + Q \tag{2}$$
$$K_k = \frac{P_k}{P_k + R} \tag{3}$$
$$\hat{x}_{k-1} = \hat{x}_{k-1} + K_k \cdot (y_k - \hat{x}_{k-1}) \tag{4}$$
$$P_{k-1} = (1 - K_k) \cdot P_k \tag{5}$$

where
$y$: input coming from the sensor
$\hat{x}$: guess
$P$: error covariance
$R$: measurement noise
$Q$: process noise
$K$: Kalman gain

Upon initialization of the Kalman filter measurement noise and process noise need to be set. The measurement noise $R$ is simply the standard deviation of the sensor. The standard deviation for barometers used in smartphones is around 0.013 hPa. The process noise $Q$ represents the idea that the state of the system, in our case the air pressure, changes over time. In several tests we empirically determined a practically reasonable value of $5 \times 10^{-7}$. The process noise basically controls how fast the Kalman filter reacts to changes in the measured values. The smaller the process noise the slower the reaction of the Kalman filter. With a process noise of 0 the Kalman filter will barely show any reaction on a change of measurement values. If this parameter is too big then the Kalman filter will no longer reduce noise.

The result of using a Kalman filter on the barometer is a noticeably more stable altitude measurement with values varying previously (using GPS) by around $\pm 1m$ to now (using the barometer) around $\pm 0.3m$. This could be improved even further by using sensor fusion as we also have a measurement of altitude by GPS.

### *Virtual Reality*

The implementation of the virtual reality mode was realized by using the A-Frame framework [26]. A-Frame serves as an entity component system for three.js and thus enables the creation of 3D environments and in particular WebVR scenes using HTML. Therefore, 3D objects can easily be created using tags. Properties of such objects are then manipulated by adding or changing certain attribute values. In the same manner the OBJ model of the virtual landscape is loaded into the scene. Data points are added by extracting their position and values from the JSON file and then creating corresponding objects.

To enable other devices in the local network to use the VR mode simultaneously, a simple webserver has been created that launches the web application. The source code can be found at the GitHub repository.

## Results & Discussion

As both apps were developed in the context of student projects focusing on the prototype development, not on evaluation, this publication focuses on the discussion of technical challenges. We hope that this may help other students or researchers to develop similar projects as well as to extend our approaches. Table 1 shows the comparison of the two different approaches.

In the following two sections, the different data representations and visualizations are discussed.

### *SensorTagVR*

SensorTagVR provides two different representations of the data:

1. Figure 5 shows a semi-transparent 3D landscape which is used to depict the temperature,
2. Figure 6 shows the data points represented as spheres hovering inside the 3D world.

The former is the default visualization of SensorTagVR which provides a better access to the height information in the VR world of the different data points than a simple plane 2D visualization. This should enable users to find anomalies or other significant points. Since the position data provided by the smartphone was not always accurate, the triangulation was improved

***Table 1.*** *Comparison between SensorTagVR and Visensor/VR*

| | **SensorTagVR** | **Visensor/VR** |
|---|---|---|
| App and Sensor Technology | | |
| Bluetooh: BLE | x | x |
| Android app | x | x |
| TI sensor tag | x | x |
| VR Frameworks | | |
| WebGL technology | x | x |
| WebVR | x | x |
| Aframe | | x |
| three.js | x | x |
| Tracking Technology | | |
| Smartphone's GPS | | x |
| Raspberry Pi's WLAN | x | |
| Surface Reconstruction | | |
| Triangulation | | x |
| Navigation | | |
| X-Box Controller | | x |
| VR-Box Controller | x | |

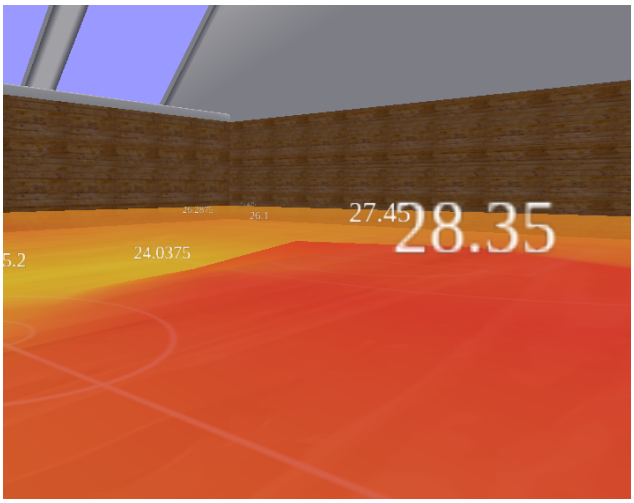


***Figure 5.*** *SensorTagVR: temperature landscape representation*

in order to create an appropriately-shaped 3D landscape representing the intensity of the measured data. The Inverse Distance Weighting (IWD) interpolation was used from the corners of a flat plane, that was subdivided into many small parts, to calculate the heights depending on the data (in case of Figure 5 the temperature). The plane is colored depending on the temperature intensity: red represents a high temperature, yellow a lower one.

Figure 8 provides a walkthrough through the different activities (screens) of the apps along with the functionalities of the respective buttons.

Using the WiFi signal strength and trilateration between four Raspberry Pis, we could achieve a indoor accuracy of up to 5 m without the usage of GPRS. This is not good enough for an accurate representation of the conditions inside a room. We tried to improve the accuracy by decreasing the distance of the hotspots, but this approach did not yield a significant increase of accuracy, as too many factors determine the strength of a WiFi signal.

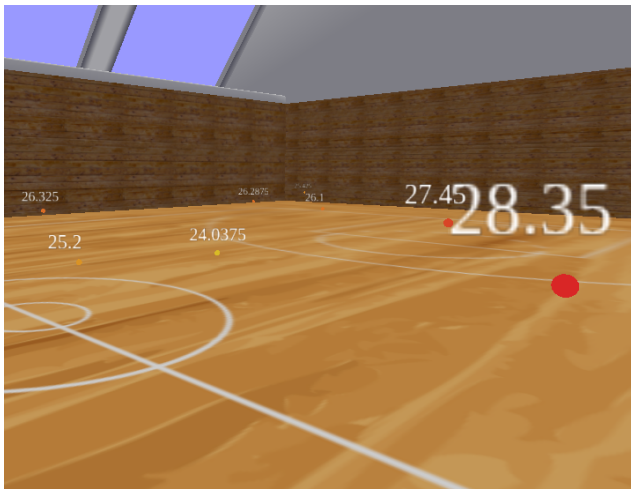


***Figure 6.*** *SensorTagVR: temperature sphere representation*

***Figure 7.*** *SensorTagVR: real world: University of Konstanz' sports hall where the data was recorded*

However, given the data from the sensor and the position data from the smartphone we were able to construct a plane. Since the sports hall's size was sufficiently huge, the impact of the inaccurate positions was fairly low. So a person can explore the room and look at the data from many angles and get a feeling for how the temperature is changing inside a room.

### *Visensor/VR*

Visensor/VR is targeted towards outdoor use. Since GPS does not reliably work inside buildings, we focused on visualizing outdoor terrains. The visualization of the terrain is only a rough approximation due to limited processing power of the smartphone and partly unreliable measurements from the barometer.

Figure 9 provides a walkthrough through the different activities (screens) of the apps along with the functionalities of the respective buttons. Figure 10 shows the virtual 3D world representation. Obviously, the Visensor VR component can visualize different data recorded with the sensor. Figure 10 top shows the humidity in blue-white scale (52.88% white to 57.8% blue), the figure's center illustrates the illuminance in black-white scale

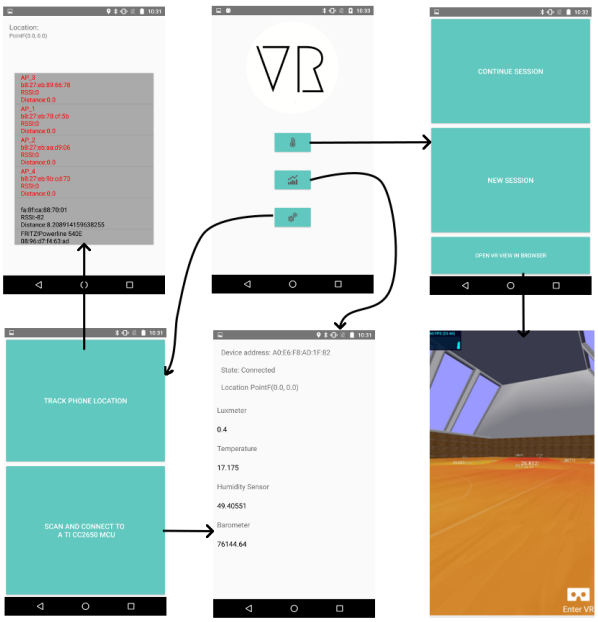


***Figure 8.*** *SensorTagVR VR: Screen Walkthrough*

(252.64 —lx black to 1516.8 lx white), and at the bottom the temperature in rainbow scale (26.44 °C to 28.25 °C).

Figure 11 shows the comparison between (left) the real world landscape and (right) the virtual 3D world. The virtual fence shows the boundaries of the measurement. Obviously, the VR world is a very rough abstraction of the real world. Figure 12 shows the application in side-by-side mode ready to be used in conjunction with the VR Box. Via head movement, the user has a 360°view, and via the connected joystick, shown in Figure 1, the user can travel the virtual world.

## Outlook

Both groups presented very promising prototypes, with a different focus regarding the use of the sensor technology. We provide here the links to the corresponding projects so that interested students or researchers can extend the underlying source code. A next step would be to extend the introduced projects and improve them towards application cases. Also, these projects could be the base for a number of application case-oriented evaluations, or could be used as a starting point to improve the stereoscopic vision for mobile devices by profiting from previous studies [27].

Finally, a few lessons learned and future directions for the two projects should be discussed.

### *SensorTagVR*

During the development of the SensorTagVR project it was noted that the trilateration of the position is not very reliable. Often, the returned position strongly depended on the smartphone's orientation or on the WiFi hotspot's location. Sometimes it was a problem if the hotspot was located behind the person doing the data collection. To increase tracking reliability, an increased number of WiFi hotspots should be a promising approach. As we used


***Figure 9.*** *Visensor/VR: screen walkthrough*

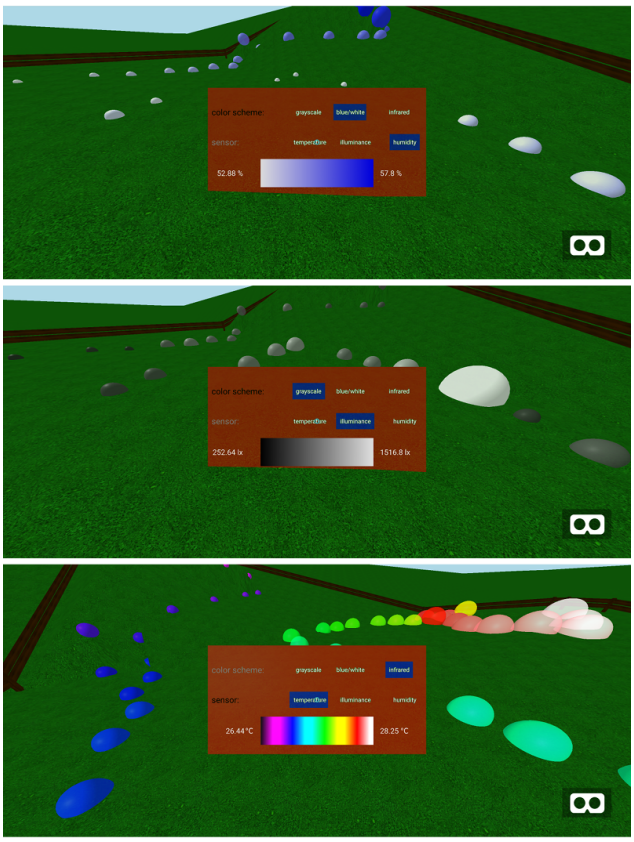


***Figure 10.*** *Visensor/VR: 3D world in the web interface, with the color scheme and data selection dialog shown: top: humidity (blue-white scale), center: illuminance (black-white scale), bottom: temperature (rainbow)*

inside a large gym only the minimum number of four hotspots, a higher coverage of hotspots should increase the accuracy. Also, the placement of the different WiFi hotspots could be optimized. Outside the building, GPRS could be used and might lead to better results.

For the VR part, a time-laps would have been the next step since the timestamps are recorded. It would have been interesting to see how the data is changing over the course of a week or longer. This could be done by constructing two planes and then interpolating between them over a few seconds to see at which position the room temperature is changing.

### *Visensor/VR*

The resulting prototype Visensor/VR can be used to create abstract 3D luminance maps. Potential future applications could be the design of solar arrays or green spaces. The beta version of the app can be downloaded from the Google Play Store.

`visensor-play.immersive-analytics.org`

As these are open source projects, we would like to invite everyone who is interested in this topic to adapt our source code to take these projects a step further – the corresponding links to the GIT repositories are located in the *Methods & Implementation* section.

## References


[1] R. Kays, M. C. Crofoot, W. Jetz, M. Wikelski, Terrestrial animal tracking as an eye on life and planet, Science, pg. aaa2478. (2015).

[2] G. L. Andrienko, N. V. Andrienko, P. Bak, D. A. Keim, S. Wrobe, Visual analytics of movement, Springer Science & Business Media, 2013.

[3] R. E. van Wijk, A. Kölzsch, H. Kruckenberg, B. S. Ebbinge, G. J. D. M. Müskens, B. A. Nolet, Individually tracked geese follow peaks of temperature acceleration during spring migration, Oikos, pg. 655. (2012).

[4] R. Weinzierl, G. Bohrer, B. Kranstauber, W. Fiedler, M. Wikelski, A. Flack, Wind estimation based on thermal soaring of birds, Ecology and Evolution, pg. 8706–8718. (2016).

[5] I. Watts, M. Nagy, R. Holbrook, I. Robert, D. Biro, T. B. de Perera, Validating two-dimensional leadership models on three-dimensionally structured fish schools, Royal Society Open Science, pg. 160804. (2017).

[6] I. E. Sutherland, A head-mounted three dimensional display, Proc. ACM Fall Joint Computer Conference 1968 part I, pg. 757–764. (1968).

[7] B. Sommer, D. Barnes, S. Boyd, T. Chandler, M. Cordeil, K. Klein, T. D. Nguyen, H. Nim, K. Stephens, D. Vohl, S. Wang, E. Wilson, J. McCormack, K. Mariott, F. Schreiber, 3D-Stereoscopic Immersive Analytics Projects at Monash University and University of Konstanz, IS&T Electronic Imaging: Stereoscopic Displays and Applications XXVIII Proceedings, (IS&T, Springfield, VA, 2017).

[8] T. Chandler, M. Cordeil, T. Czauderna, T. Dwyer, J. Glowacki, C. Goncu, M. Klapperstück, K. Klein, K. Marriott, F. Schreiber, E. Wilson, Immersive Analytics, Big Data Visual Analytics, BDVA 2015, pg. 73–80. (2015).

[9] J. Mueller, S. Butscher, H. Reiterer, Immersive analysis of health-related data with mixed reality interfaces: potentials and open questions, Workshop Immersive Analytics 2016 (in conjunction with

***Figure 11.*** *Visensor/VR: (left) real world vs. (right) virtual world*

***Figure 12.*** *Visensor/VR: Running on a smartphone inside the VR Box in side-by-side view*


ACM Companion on Interactive Surfaces and Spaces 2016), pg. 71-76. (2016).

[10] A. Chirico, F. Lucidi, M. De Laurentiis, C. Milanese, A. Napoli, A. Giordano, Virtual reality in health system: Beyond entertainment. A mini-review on the efficacy of VR during cancer treatment, Journal of Cellular Physiology, pg. 275–287. (2016).

[11] S. Johnston, L. Renambot, D. Sauter, Employing WebGL to develop interactive stereoscopic 3D content for use in biomedical visualization, Proceedings of the SPIE 8649: The Engineering Reality of Virtual Reality 2013, pg. 864905. (2013).

[12] B. Sommer, C. Bender, T. Hoppe, C. Gamroth, L. Jelonek, Stereoscopic cell visualization: From mesoscopic to molecular scale, Journal of Electronic Imaging, 1, 011007. (2014).

[13] Unity3D, Unity Game Engine, `https://unity3d.com`. (last accessed 2017-04-20).

[14] World Wide Web Consortium, WebVR: Repository for the WebVR Community Group and the WebVR Specification, `https://github.com/w3c/webvr`. (last accessed 2017-06-13).

[15] World Wide Web Consortium, WebVR - Bringing Virtual Reality to the Web, `https://webvr.info`. (last accessed 2018-01-07).

[16] Authors, three.js - Javascript 3D library, `http://threejs.org`. (last accessed 2018-01-07).

[17] Google Earth Blog, Google Releases Google Earth VR for HTC Vive, `https://www.gearthblog.com/blog/archives/2016/11/google-releases-google-earth-vr-htc-vive.html`. (last accessed 2017-06-14).

[18] H. T. Nim, M. Wang, Y. Zhu, B. Sommer, F. Schreiber, S. E. Boyd, S. J. Wang, Communicating the effect of human behaviour on the great barrier reef via mixed reality visualisation, IEEE Big Data Visual Analytics (BDVA) 2016, pg. 1–6. (2016).

[19] H. T. Nim, B. Sommer, K. Klein, A. Flack, K. Safi, M. Nagy, W. Fiedler, M. Wikelski, F. Schreiber, Design Considerations for Immersive Analytics of Bird Movements Obtained by Miniaturised GPS Sensors, Proceedings of Eurographics Workshop on Visual Computing for Biology and Medicine (VCBM) 2017, pg. 89–96. (2017).

[20] C. Marrin, WebGL Specification, Khronos WebGL Working Group (2011).

[21] Texas Instruments, SimpleLink™ Bluetooth low energy/Multi-standard SensorTag, `http://www.webcitation.org/6wISQqsPM`. (last accessed 2018-01-07).

[22] The OpenWrt developer team, OpenWrt, `https://openwrt.org`. (last accessed 2018-01-07).

[23] S. Wiedemann, Solves a formulation of n-D space trilateration problem using a nonlinear least squares optimizer, `https://github.com/lemmingapex/trilateration`. (last accessed 2018-01-07).

[24] D. Shepard, A two-dimensional interpolation function for irregularly-spaced data, Proceedings of the 1968 ACM National Conference, pg. 517-524. (1968).

[25] P. Bourke, Triangulate: Pan Pacific Computer Conference, Beijing, China, 1989, `http://URL`. (last accessed 2018-01-07).

[26] A-Frame Authors, A-Frame Make WebVR, `https://aframe.io`. (last accessed 2018-01-09).

[27] H. Morikawa, Y. Banchi, S. Tsukada, Y. Hasegawa, S. Takahashi, K. Ohta, T. Kawai, Effect of inter-lens distance on fusional limit in stereoscopic vision using a simple smartphone head-mounted display, IS&T Electronic Imaging: Stereoscopic Displays and Applications XXVII Proceedings, (IS&T, Springfield, VA, 2016).