Saliency maps for interactive 3D environments in virtual reality

Vivien Gagliano

École nationale supérieure d'informatique pour l'industrie et l'entreprise

---

Internship report

« Saliency maps for interactive 3D environments in virtual reality »

Student : **Vivien GAGLIANO**
ENSIIE academic tutor : **Guillaume Bouyer**

Institute : **Institut National de Recherche en Informatique et en Automatique** (Sophia Antipolis, 06)
Supervisor : **Hui-Yin Wu**

Second year internship, from the 14th of June 2021 to the 20th of August 2021

Preamble

I would like to thank the members of the Biovision research team, who have welcomed me with much kindness within their department and the INRIA research center. In particular Mrs. Wu, who guided my research project and offered me useful insights throughout the whole internship.

This internship was very fruitful as it allowed me to work on a project at a greater scale than what I was used to, it taught me how to organize and manage a project where I work alone. I also was able to deepen my skills in computer graphics by putting into application theoretical knowledge learned in class, and going even further. Graphics being my choice of specialization for the rest of my studies, this has been a great opportunity for me. In addition to that, this project let me have a first taste of the world of research, which I found greatly enriching as I learnt how and where to search for relevant, scientific information on the web.

INRIA research center

INRIA global

Inria is the French national research institute for digital science and technology. With more than 3900 researchers and 200 research groups from all nationalities dispatched in 9 centers mostly in France, this group focuses on major issues connected to the digital world in various different areas.

Biovision team

The team I entered is a small group leading several research projects all related to the human vision. Its purpose is to deepen our understanding of the visual system at different levels, study the impact of vision pathologies and try to design solutions for diagnosis and rehabilitation.

Within this team, I worked on a newly founded project lead by researcher Hui-Yin Wu, my tutor. Being responsible for doing baseline work on this new project, I worked alone during the whole internship. This is actually relevant as it means that, apart from the weekly report I provided my tutor and discussions I had with her, I was effectively organizing the work all by myself, and didn't have to collaborate with other team members.

Despite that, I still was in contact with these other members, notably two young virtual reality engineers with whom I shared an office in lab. This was a fruitful association, as we worked on sensibly similar projects and were able to discuss common issues.

Other groups besides Biovision were also researching on human vision subjects in my work environment, we thus were able to share and discuss our projects in a meeting.

Table of contents

1 Introduction

This report presents the work completed during my internship in the Sophia-Antipolis research center within the Biovision team. I have worked during ten weeks to develop a toolbox application to allow visual attention data collection and analysis in a three-dimension, virtual reality environment. This application is to be used later by researchers to help study visual pathologies and elaborate treatment solutions. I have during these weeks put in place every steps of the workflow: data collection, analysis, and visualization.

1.1 Context

Eye tracking is a recent research field, its study really began in second half of XIXth century. It has several fruitful applications in many different fields, such as marketing, entertainment or in our case medicine. And if the topic has already been well explored in two-dimensional space (on desktop monitor), few research have been conducted in three-dimension, and even fewer in a virtual environment.

1.2 Challenges

By adding another dimension to the problem, several aspects of it change, and the complexity of the study effectively increases at the same time. Searchers now facing these new issues must understand how to address them properly to be able to model precisely subjects' visual attention, and use available data to draw meaningful analysis. Using virtual reality further increases both the difficulties and the opportunities of the research, this technology must thus be mastered down to the little details in order to be used to its fullest potential.

1.3 Goals

To be able to diagnose one subject's visual pathology and offer rehabilitation solutions, one needs to study the pathology on several test subjects, and the approach used in this research project, using eye tracking in virtual reality, comprises several steps, each one having its own technical and conceptual difficulties. Searchers need to collect data among test subjects, process and analyse this data and come up with models to explain the pathology, models which then need to be tested before being accepted.

1.4 Research question

This internship didn't cover every step of this process however, as it wasn't the purpose in the first place, and ten weeks wouldn't be enough time anyways for that. Rather, the issue I addressed was how to gather and prepare subjects' data for further, higher-level analysis. To achieve that, data for both visual attention and head position must first be recorded and synchronized. From this raw data, more relevant information can then be extracted to obtain better insights of the subjects' visual behavior, insights finally displayed in different ways to allow visual interpretation and better understanding.

1.5 Glossary

The following acronyms and abbreviations will be used during this report :

- **HMD**: *Head-Mounted Display* is a display device worn on the head that allows to project images in front of one or two eyes using integrated screens.
- **VR**: *Virtual Reality* will in this report refer to an immersion method into a digital world achieved using a headset, meaning a binocular HMD combined with an inertial measurement device.
- **ET**: *Eye Tracking* the act of recording gaze behavior by using camera pointing at user eyes.
- **POR**: *Point of Regard* refers to a point in space pinpointing gaze focus for one given data sample. This is a not-physically accurate term, as human eye sees in regions rather than exact points, but it remains a very practical tool and terminology.

2 State-of-the-art

2.1 Principles of eye tracking

Despite the technology still being young, numerous research teams have already worked and studied different aspects of the subject. One notices easily that when recording eye movements, among the six degrees of freedom technically possible for a 3D object in space, translation can be forgotten if the user's head is fix or if the camera is embedded in the HMD, these being the vast majority of the cases. Furthermore out of the three axis of rotation, one is actually physically unusable for the human eye (rotation around the axis passing through the middle of the pupil). What is left are thus two coordinates accounting for the rotation of the eye along the remaining axis, in the form of pupil positions in a 2D plane. When eye tracking devices indeed record user gaze, pupil position is actually what is being traced and registered. What most of these devices do is take a reflected static light to reference the pupil position.

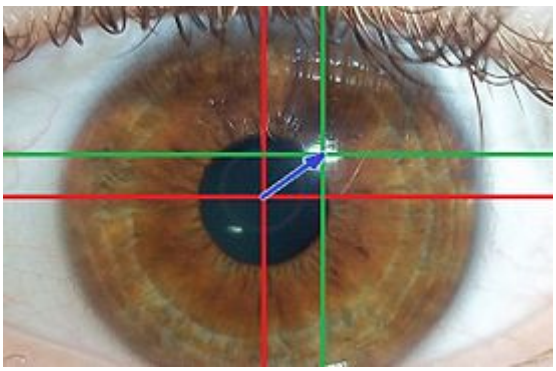

Figure 1: Reflected light used to measure pupil position in the eye

The objective is then to find the stimulus, the point of regard corresponding to each pupil position. That is to say, one needs to find a mapping from the pupil position space to the points of regard space, being it 2D or 3D.

2.2 2D eye tracking

In two dimensions the problem is relatively simple, we need a mapping $f : \mathbb{R}^2 \to \mathbb{R}^2$. Linear mapping provides good results, and higher order polynomials can be used to compensate for some anomalies due to the eye being spherical, and not a perfectly flat plane. Wang et. al have expanded on how to compute the coefficients of the mapping by using a calibration phase [17]. If pure salient features are the desired outcome however, other methods can be used, not involving recording a subject's gaze. Several computational models indeed do exist to extract salient features directly from an image using different computer vision techniques such as Fourier spectrum analysis, decomposition based on low-level visual features or others [16].

2.3 3D eye tracking

Saliency can also be computed for 3D images, however asserting observer fixation in 3D space creates new challenges which need to be addressed. Pfeiffer et al. have explored eye tracking in 3D immersive environment, and have highlighted four major requirements which need to be addressed to have a valid analysis in 3D space[11].

- **Estimating 3D points of regard:** This is the biggest difference between 2D and 3D visual tracking, as depth of objects present in the scene needs to be taken into account to have a physically valid model. Humans use several criteria to evaluate depth perception, both monocular (*occlusion, size in field of view, perspective, color gradient or motion parallax*) and binocular (*disparity, accommodation, and most importantly vergence*) [10]. To evaluate depth accurately, one must thus select and use one or more of these criteria, some of them being easier and more intuitive to use than others.

- **Modelling spread of attention:** Human eyes can see accurately not in a single point but rather in small area. This area corresponds to the fovea centralis, a zone of high acuity found in the retina.

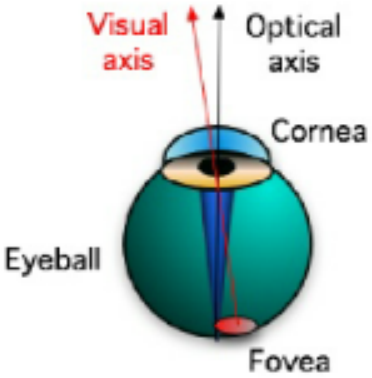

Figure 2: Side view of an eye diagram

  This must be taken into account by applying a 3D spread model on the acquired points of regard. And while applying a 2D gaussian model is commonly seen in the literature, 3D gaussian model has proved to be more realistic[11].

- **Handling occlusion:** As depth has now to be taken into account, handling occlusion becomes more difficult, one needs to know which object was gazed at when several objects are aligned, and partially occluding one another. Occlusion also changes according to the point of view, that can change since subjects can move freely in VR.

- **Allow for multiple perspectives:** Compared to desktop where all viewers see the exact same image, subjects can take several different point of view, as explained above. Computing and displaying visual attention must thus be done in a way that allows exploring the scene in a different way that the subject did.

Wang et al. have also explored the possibility of computing 3D saliency maps without recording user gaze [16], by combining 2D saliency feature extraction with depth information, the two being combined in different ways.

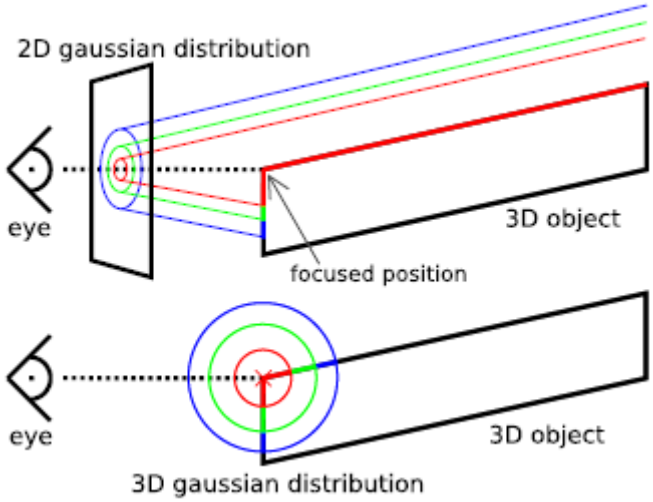

Figure 3: Projection of 2D gaussian model versus actual 3D gaussian model

2.4 Data synchronization

Another delicate issue which needs to be handled in most cases when doing eye tracking research is data coordination and synchronization. Recording data often times provides at least two data streams, gaze and head, which might not come from the same device. This means the timestamps we have for each stream come from different internal clock. The challenge is thus to find or create a synchronization point, meaning a pair of samples, one from each stream, representing the same instant in time. This has proved to be a tedious issue due to the different clocks being unrelated, therefore not synchronized, but also due to differing sample rate of different recording devices.

Methods to create synchronization points in data have been elaborated, by making a predictable coordinated head and gaze movement during a calibration phase in order to pin-point a pair of samples identifying the same instant [2]. Another tool commonly used in the literature is Lab Streaming Layer (LSL), a system for the unified collection of measurement time series in research experiments that handles the networking, time-synchronization, (near-) real-time access as well as optionally the centralized collection, viewing and disk recording of the data. This very-well known tool has been adapted for many devices in many software and programming languages, it can often act as a default, ready-made synchronization tool (at the cost of complexity, as putting it in place demands some work to be done). Some development kits used with data recording devices also provide data synchronization API which can be used directly.

2.5 Computing points of regard

Several methods can be used to compute points of regard, based on different criteria among the ones mentioned previously, the one most commonly found in the literature being the following: ray casting, using vergence, and using a predictive model. Ray casting involves in a way or another to recreate the subject's gaze vector, and check for intersection between this vector and scene objects [11] [9]. This method has limits in that it doesn't always handle occlusion well, and makes the assumptions that gaze is always

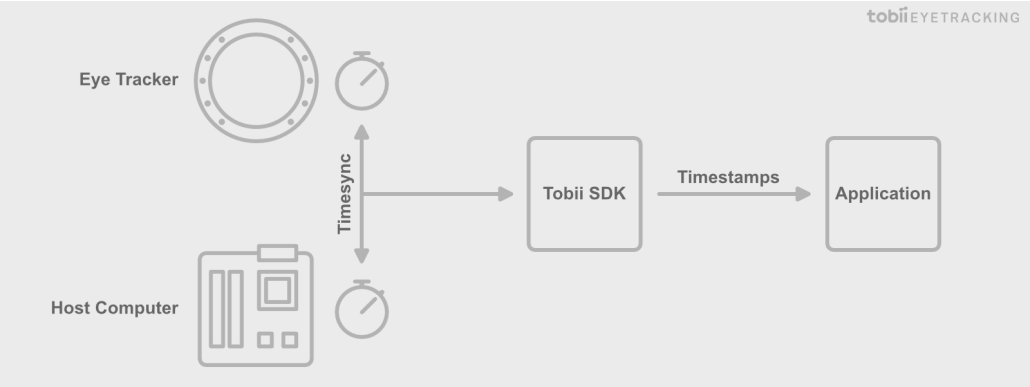

Figure 4: Tobii Eye Tracking SDK time synchronization workflow

focused on an object. It also requires some knowledge on the geometry of the considered scene, to check for intersection.

Vergence refers to the fact that human eyes will rotate so that pupils can move towards or away from one another when focusing. Another way to compute points of regard is to use this physical phenomenon by intersecting gaze vector for both eyes, the intersection indeed being the area of visual focus [11]. This is often more practical, as it requires no geometrical information on the environment.

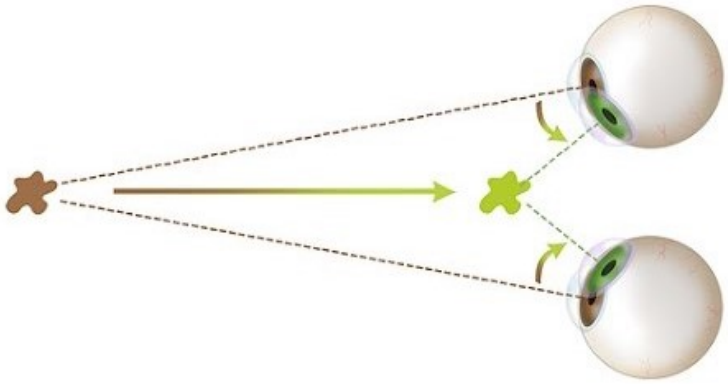

Figure 5: Vergence phenomenon

Finally POR can be computed with machine learning models learning the correct mapping from the 2D coordinates of the fixations of both eyes on a display to the fixated point in depth, as studied by Pfeiffer et al. [10] and Pirri et al. [12].

2.6 Classification algorithms

An essential part of eye-movement data analysis is fixation identification. Indeed not all eye movement events contain relevant information, a selection thus has to be made to filter out useless samples. These events are most commonly split between fixations, intuitively groups of points where a subject has focused his/her gaze, and saccades, points between fixations. This classification is primordial in an ET study, as many higher-level

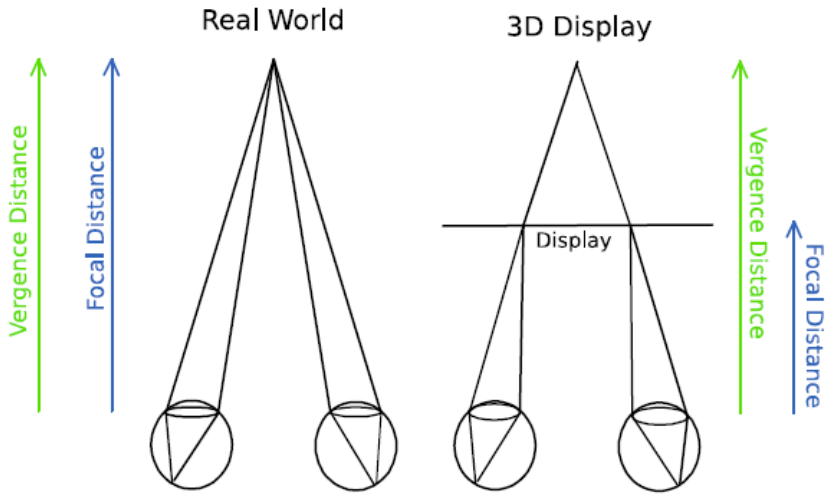

Figure 6: Unlike in real worl, vergence and focal distance are not equal in VR

analysis rely on it to draw further conclusions, and it also simplifies data by filtering and clustering. Many classifiers have been studied in the literature, each model having its advantages and inconveniences. The main idea revolves around the fact that PORs for one fixation will be spatially and temporarily close from one another. The points must thus be compared using specifically chosen metrics, and the algorithms are categorized according to these discrimination criteria [13]. An import contribution to this field is the velocity-based classifier presented by Nyström and Holmqvist [8], which served as stepping stone for many following research groups [5] [6].

2.7 Visualization methods

Depending on the purpose of the study and how the data is processed, it is useful to have several ways to display subject attention so that study results are rapidly and easily understandable by both scientists and less-informed persons. People to which the studies are presented are indeed not always experts in the field, having clear and straight-forward representation is important for them. Here follows a (non-exhaustive) list of the methods most commonly encountered.

- **POR as sphere:** The most basic display is to simply display the computed PORs or fixations as sphere, using size or color of the sphere to encode additional information such as distance from user or duration. Yet this is a physically inaccurate model, as the human eye sees in region and not point as discussed previously, it allows for a fast and intuitively easy representation. Clay et al. for example have use this method to display gaze points from their subjects [4].

Figure 7: Spheres represent hit points from gaze vectors, color representing distance to user (red = far, blue = close)

- **Scanpath:** A slightly more detailed method that consists in displaying fixations as sphere, but linked together to be able to retrace the path of the user gaze during the experiment. Order of passage can be encoded in different ways, and sphere size usually reflects fixation duration.

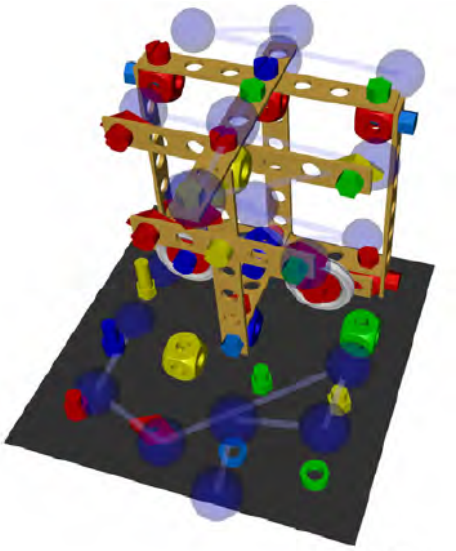

Figure 8: Visualizing sequence of fixation using 3D scanpaths by Pfeiffer et al. [9]

- **Gaze ray:**Instead of representing PORs or fixations, which are gaze ray intersections with the scene, one can choose to directly display gaze rays. This provides additional information in that we also have the origin of the ray, that is the user position, for every sample, and it also allows to display visual attention even when user is not gazing at an object (in the sky for example). This method was used by J. Simpson to display the results of his study on outside laboratory eye-tracking, where the dynamics of pedestrian gaze distribution upon the environment is visualized [14].

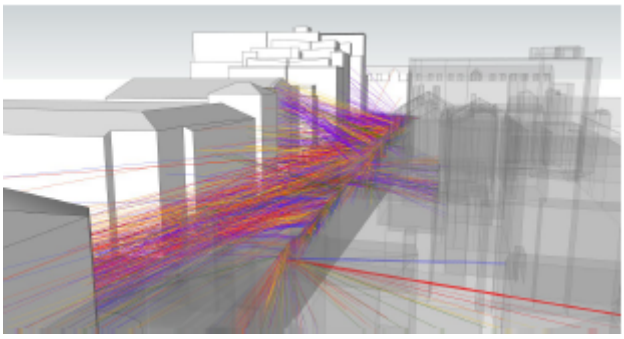

Figure 9: 3D mapping of gaze rays in a street model

- **Attention volume:** Another possible way to display fixations is to use a continuous weighing function as a fixation membership function. The idea is to model attention spread to have physically more accurate results by using 3D gaussian distribution. The resulting visual attention values are then displayed using color-coding, and fixations where PORs are clustered are thus highlighted [9].

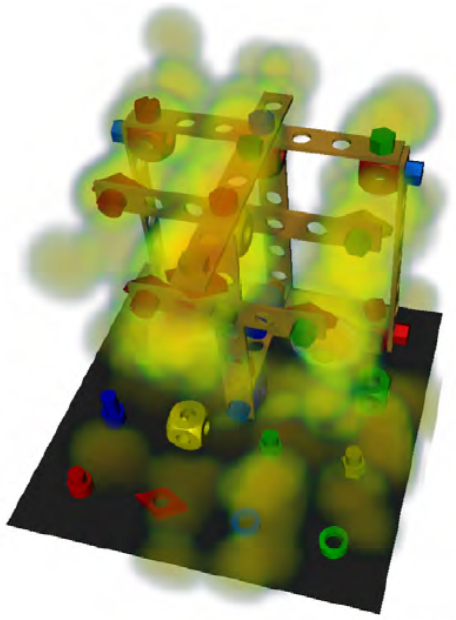

Figure 10: 3D attention volume use volume-based rendering

- **Heat map:** or saliency maps are interesting as they highlight where the center of attention is among all gazed areas. They can either be 2D [15] or 3D [11].

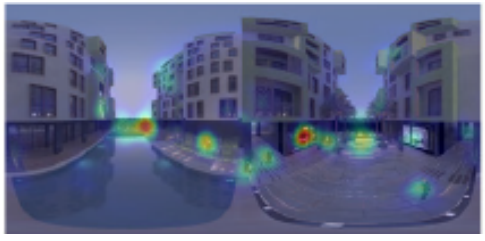

Figure 11: 2D heatmap in a VR panorama

Figure 12: 3D heatmap as a texture

2.8 Design choices

The objective of my internship was not to add to the state-of-the-art by discovering new technique or model, but rather use the already existing material to create a tool which could then be used to research and make contributions. Using the existing bricks, I thus developed a modular application, in the sense that each part was well separated, allowing for a precise control over methods and features used later down the line.

3 Implementation

At its simplest, the workflow that was put in place can be split in three different steps, which I implemented one by one in the following order: collect, process, and visualize data. The steps are in essence independent, and great deal of caution was taken to keep this independence even within the code, so that if one needs to upgrade a piece, the whole code doesn't need to be rewritten.

3.1 Tools and concepts explored

ET research usually relies on external eye tracking devices, typically camera pointing at subjects' eyes or eye tracking glasses. However in VR, external cameras cannot be used as eyes are covered, integrated camera must thus be used. Several HMDs now include eye tracker, yet most of them are not meant for general, commercial use, but rather are research-oriented (more expensive, less practical use, but higher specifications). The headset I worked with is the HTC Vive Pro Eye, actually one of the most accessible eye tracking headset available on the market, offering 0.5° - 1.0° accuracy at 120 Hz gaze data output frequency and 90 Hz refresh rate.

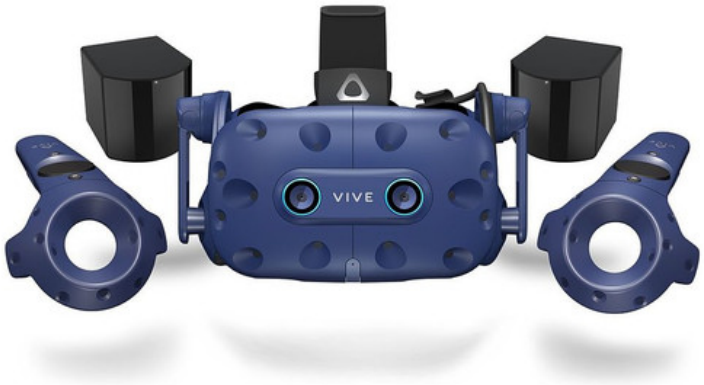

Figure 13: HTC Vive Pro Eye ET headset

The headset is a Vive product, however the built-in eye tracking device is from Tobii Pro, a major eye-tracking solutions provider. To access tracked gaze data, two development kits are available for this headset: Vive SRanipal SDK or Tobii XR SDK. The

latter has two usable licences, a basic one with most commonly used features and a more advanced one, offering a wider range of feedback, tool and configuration. The licence for the advanced API being outside of the project's current budget, I decided to work with the Vive SDK, which had the advantages of being free and still more complete than Tobii's basic SDK.

To develop the application, two options were available: create a C++ software using the chosen SDK, with a custom graphic interface made with Qt. However this option was not ideal in this case, as it would take much more time to develop, and would be mush harder to build upon further down the line. The second option, which was chosen, was to develop using Unity3D game engine. This allowed to focus on setting-up the eye tracking context, and will allow easier integration and incrementation of the project. The programming language used for the main workflow is mostly C#, mixing component-oriented and object-oriented notions. Furthermore some extent of graphics languages (CGPROGRAM, ShaderLab and HLSL) were also used in the later part of the internship, when attempting to create a shader to visualize 3D saliency maps.

The tools I used to organize my work and communicate with my tutor were Microsoft Teams, where I took notes on my work in the form of weekly reports, which I presented during regular meetings, as well as a Trello board, mainly used to list and visualize the different tasks of the project.

3.2 Data collection

The first step in the process for eye tracking research is to collect the data which will later be used for analysis. Two data streams have to be collected: head data, with head position and rotation, and gaze data. The data package sent by the eye-tracking device comprises information on user presence, eye expressions, as well as more detailed knowledge for each eye singularly and for a virtual combined eye. This includes gaze origin and direction, eye openness and pupil diameter, as well as a 64 bit validity value which can be used to extract validity for all previously mentioned fields. Both head and gaze samples also have a timestamp, recorded each on their own internal device clock (the timestamps from different streams can therefore not be compared directly, some comparison method must be established).

Despite the data coming from the same headset, the sensors used for head and gaze tracking are not the same. The way to collect these are in consequence not the same either. The head samples are collected on a sampling thread, separate from the main one where the engine runs, by registering to an event invoked by the API when ET device does the sampling. For the head data however, there is no event called by the recording device, we thus use a sampling routine, also running on another thread, that regularly records head information in the Unity scene. Both are then stored on disk using JSON format for easier accessibility and adaptability. For data synchronization purposes, a third file is also written where system timestamps of each stream first sample are recorded. Caution needed to be taken however when handling data collection, as not all methods and variables were available when sampling data from other thread, and

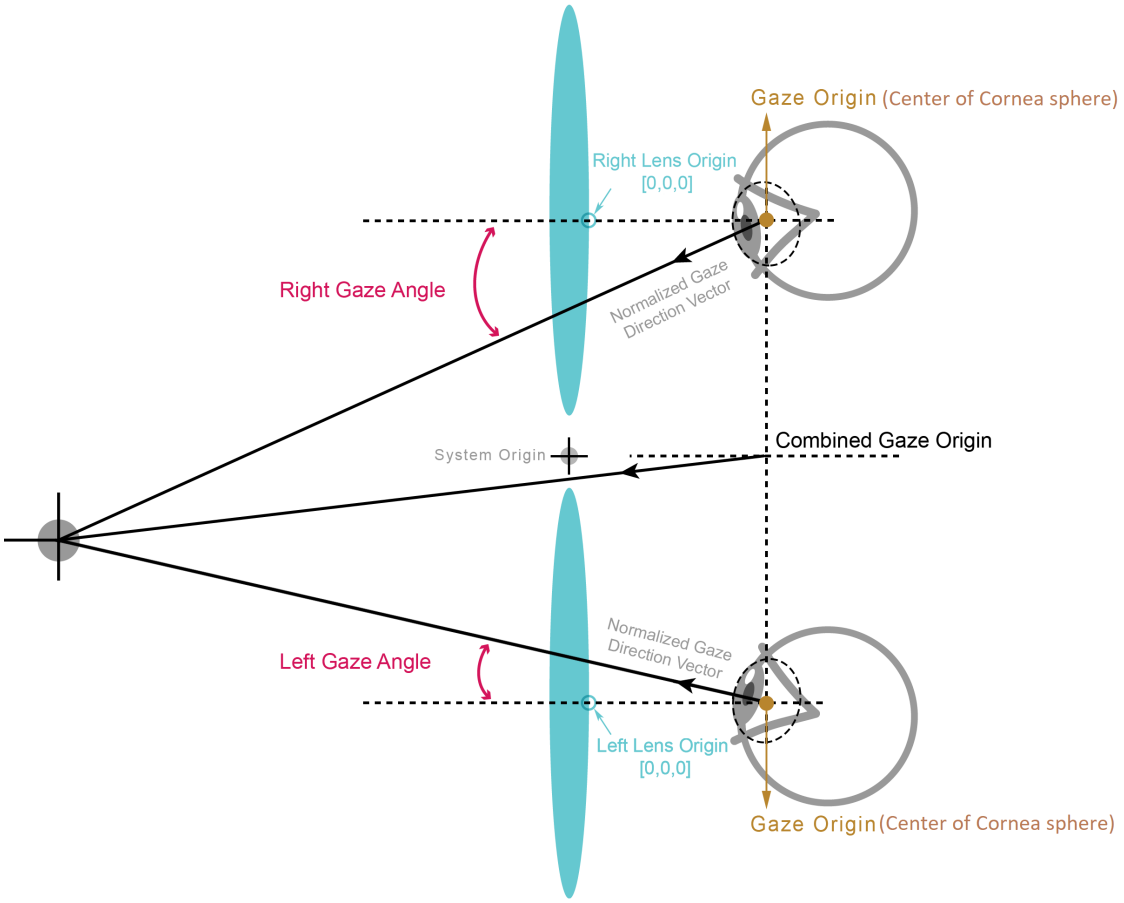

Figure 14: Information provided by ET data package

writing to the synchronization file from two different threads could lead to simultaneous file access, causing fatal error.

3.3 Data processing

This part of the process is the most important one, it is where we make use of the data we have collected. However it cannot be used raw, some filtering and processing need to be done in order to achieve exploitable results. As head data is collected by recording desired field in Unity scene, it is never invalid. However the data provided by the eye-tracker can be, we must ensure all samples we use are trustworthy. For that we check with the aforementioned validity bit mask that the fields we use are valid, that the user is present in the headset, and that he is not blinking.

Once the invalid samples have been removed, the issue of data synchronization can be handled. This remarkably delicate problem stems from the fact that timestamps for both data stream come from different, unrelated clocks. The different techniques used to

record data also imply that the sampling doesn't start at the same time for both streams. With no way to directly compare the two different timelines, a custom synchronization solution has to be devised to create a synchronization point, meaning a pair of samples, one from each stream, where both represent the same *i*nstant in time.

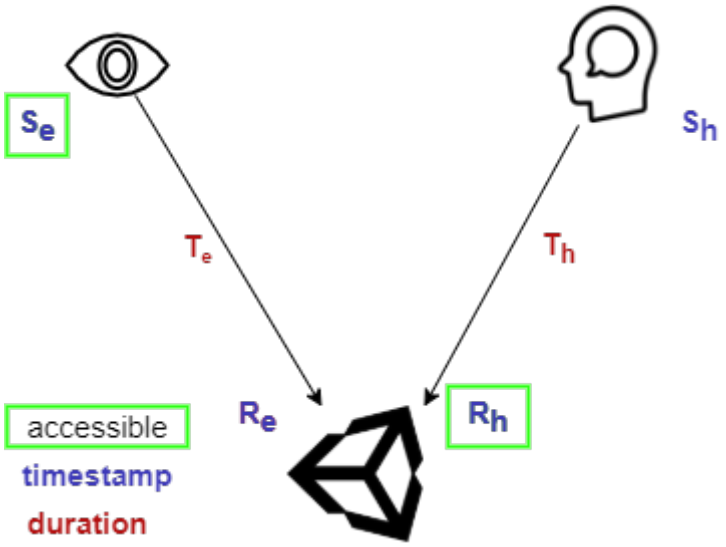

Figure 15: Diagram displaying when data are sampled and received by the system, the time between the two being the travel time

One must pay attention here: S and R are timestamps, recorded in their own device clock, whilst T describes a duration. Duration are absolute, while timestamps are relative (in the sense that one duration has meaning by itself, unlike a timestamp which needs another timestamp or a global view over the clock timeline to be relevant). Perfect synchronization however wasn't achievable as we did not have access to travel times T, nor to eye receive time R_e or head sample time S_h because the chosen API did not provide the necessary elements (it must be noted that while other API may provide more detailed data, not all mentioned values are obtainable, especially travel times, as it would require communication protocol between different devices that are not meant to communicate in such way). I therefore implemented a temporary and approximate solution that allowed to get decent results.

This approximation is to consider that the first sample of each data stream are synchronized, meaning they represent the same instant in time. Since samples are never going to be synchronized again after that because of the different sample rate, and the fact that two unrelated events happening at the exact same time is a null probability event, we need to synchronize manually. To do that data has to be interpolated: gaze data is used as baseline as this data stream has a higher base sample rate, and for each gaze sample, the head sample that is synchronized to it is computed. To do this, we use the fact that both the timestamps of the first samples (each one in its own device clock) and the duration between initial sample and current sample are known, because we have access to gaze timestamps. Subtracting a timestamp from a timestamp resulting in a duration, this duration can be added to the head initial timestamp, thus obtaining the timestamp of the head sample that would exactly be synchronized with considered gaze sample. However this head sample won't be found among the recorded samples, hence we get the two nearest head samples and interpolate our head sample between these two. For now, interpolation is linear, however different method could be explored to have a physically more accurate model.

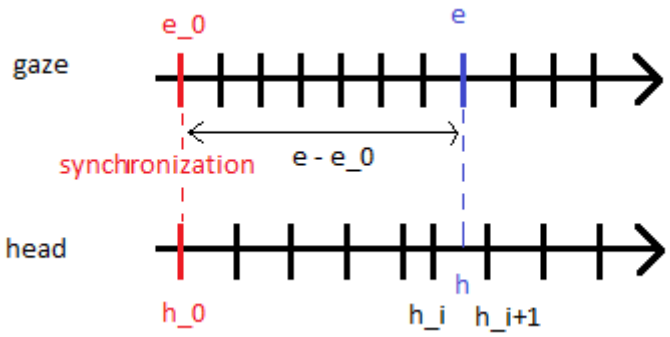

Figure 16: First approximation for data synchronization

Considering first samples as synchronized if of course is not true, because the sample times for both are unknown. Even the reception timestamps are probably not the same, because of how data is recorded differently for each stream. To account for that, the previous model can be upgraded by adding a delta value, representing the duration between the first sample of each stream. To do this, the reception timestamp of the first sample of each stream are recorded in a file, the mentioned synchronization file, and the delta is set to the difference of the two.

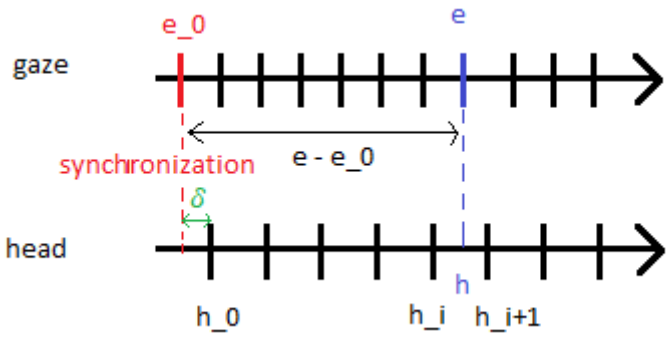

Figure 17: More accurate approximation for data synchronization

However another approximation is made, further lessening the accuracy. Because head data collection is done by updating variables in Unity main thread, and then actually sampling and writing data in alternate thread, the first few samples are incorrect, and usually are duplicated. The reason for that is that the first few frames of a scene always take more time to execute, as all initialization are done here. Therefore the sampling thread will sample more than the main thread will update the variable. The solution used is instead of taking the first sample for synchronization, the first sample after sampling becomes regular is used, effectively eliminating the first few milliseconds of the experiment. However the timestamp used to compute delta is still the same.

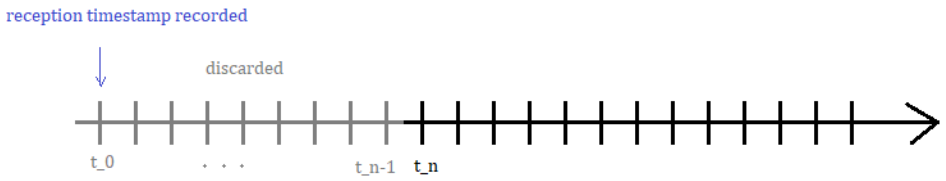

Figure 18: Cleaning head data

This diagram highlights better the issue: post-cleaning we consider head sample number n to be the first sample. Later in code, when using first head sample timestamp we thus refer to t_n, but when using first head reception timestamp, we still refer to the one of sample number 0.

When working with eye tracking, raw collected data goes through several steps of refinement to achieve higher level of analysis and precision, each step relying on the previous one to be computed.

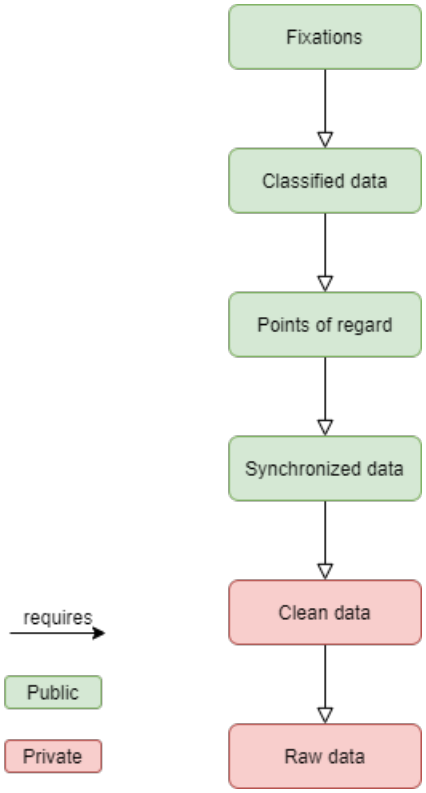

Figure 19: Different steps of ET data processing

After cleaning and synchronizing, points of regard can then be computed. Two methods have been implemented, both already present in the literature: ray casting and using eye vergence. With these, eye movements classification was achieved with a very basic velocity-based algorithm, from which fixations were then extracted. The implementation that were put in place for these steps were both custom-made and overly simplified, as the objective was to set up a workflow with all the basic elements. Upgrading all these algorithms to state-of-the-art models to obtain scientifically exploitable results was not

the goal of this internship, and has been left up to the searchers that will use this tool later on.

3.4 Data visualization

To best understand and exploit results, it is important to be able to visualize them. Two display methods were implemented during this internship: points of regard and fixations as sphere. For points of regard, instantiating a dummy sphere for each POR is sufficient, and allows to visualize well the user's gaze path. As for fixation, the same thing could be done, adjusting sphere size accordingly to the fixation's duration.

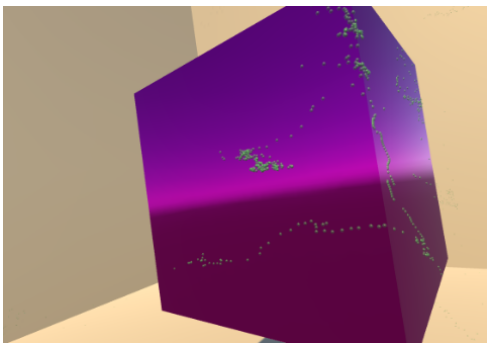

Figure 20: POR computed by ray casting

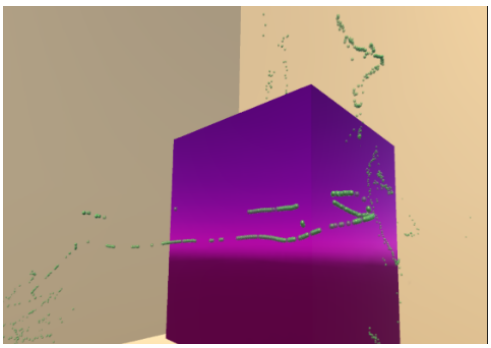

Figure 21: POR computed by vergence

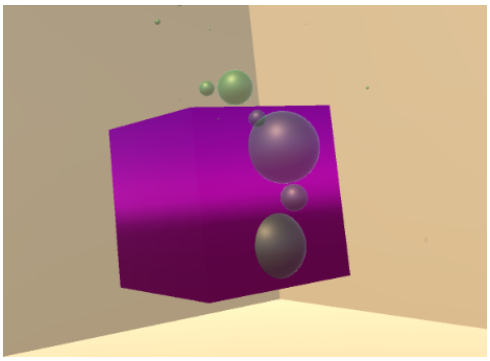

Figure 22: Fixations

Alongside visualization by sphere, I produced a first prototype of saliency map shader, that once applied to scene objects would display the visual attention received by each point of each objects from the fixations in the scene. The concept of this display method is to apply a 3D gaussian model to the computed fixations, and display how it intersects with the scene objects. An attention value is thus attributed to each point in space, value obtained by adding the contribution of each fixation towards the considered point. In order to achieve real-time rendering speed, all attention values are compute from with the shader, in the fragment program. Computing from vertex program would mean less operation, but it is actually an error as the resulting attention value would then be interpolated for all fragments within mesh triangles during the GPU rasterization stage.

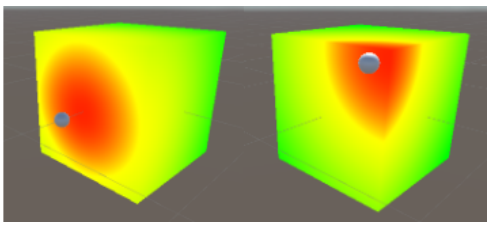

Figure 23: Computing attention from fragment shader

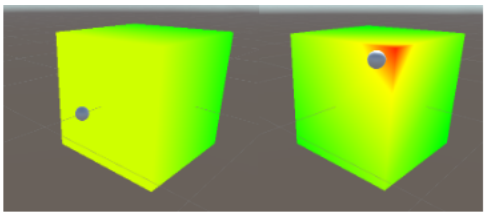

Figure 24: Computing attention from vertex shader

However this feature was left to an experimental stage, as many problems were encountered while trying to implement it, and I was ultimately unable to overcome all these problems in given time. Major factors to the lack of successful outcome were the lack of time, the high specificity and complexity of the required task, as well as the fact that a consequent part of the resources found online on subject was aimed at an implementation that differed vastly from ours (tutorials found usually related to OpenGL graphics library, whilst we were using CGPROGRAM in Unity). The context and ways to handle different objects varied in consequence.

Some of the options I explored while trying to find a solution include implementing a system similar to shadow mapping to handle occlusion, both from CPU and GPU, applying attention heatmap as a post-processing effect on the rendered image, and saving saliency maps as textures for each scene object, computed from GPU during a render call.

3.5 Application flow

The different steps of the process are handled by different classes: data collector, processor and visualizer. The three of them are purposely independent in code, yet they sometimes communicate between each other. The link between these three classes and the user (the one using the application, not the one whose gaze s being recorded) is made via a menu scene, where a graphical interface has been put in place with basic Unity UI elements, combined with a script to handle the menu logic.

Figure 25: Menu scene

When running the application, the workflow is the following. Starting from the menu scene, the supervisor is able to select and adjust different experiment settings, including numerical parameters, file path, display mode and others. Clicking on corresponding button will then start a recording or visualization session in selected scene. If recording, the data collectors will be enabled and will start sampling, writing data in files. If visualizing however, data visualizer will be enabled and will query the data processor for the elements required to display what has been selected. At this point the processor will compute just what is needed, not more, and send the data back to the visualizer.

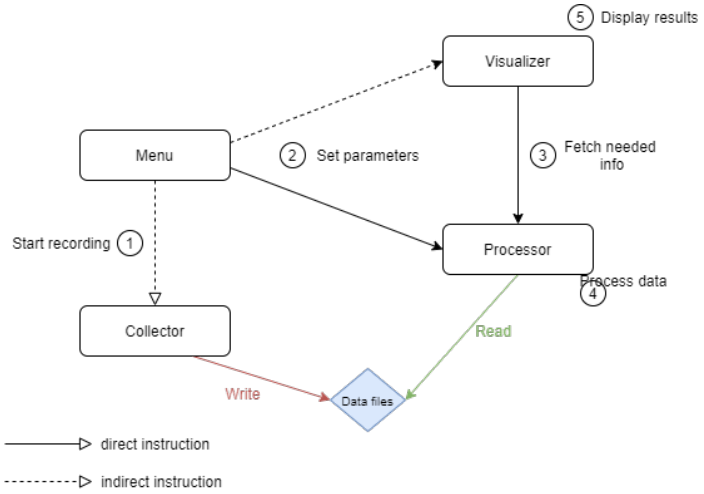

Figure 26: Relation between application entities

To use custom scene for a session, the steps were simplified as much as possible. First import the scene into the project, and enable VR features by upgrading camera to a VR camera. Then a package containing elements necessary to the inclusion of ET in the scene (made into a prefab in the project files) can simply be dragged into the scene hierarchy. The only thing left is to add the scene in build settings, so that it can be selected from menu.

4 Conclusion

During this internship, I was able to put in place an application in the form of an Unity project which lays the baseline elements needed for an eye-tracking research study. The application covers the three main steps required study, that is data recording, processing and visualizing. The projects includes a functional graphical interface, making it non-developer friendly as it was produced for a target that is not necessarily familiar with computer science. The output raw data files are encoded using universal JSON format, allowing for great readability and portability. Furthermore the project was carried with a modular set of mind, meaning the basic elements put in place to complete the workflow can easily be upgraded without affecting the code for the other parts, and additional features will easily be set up further down the line.

The internship was nevertheless not carried without missed objectives. I indeed stumbled on several unforeseen issues, which I couldn't always successfully overcome. As discussed, more advanced tools and/or more time would have been necessary to tackle these problems.

The current project will be able to serve as starting point for further eye-tracking studies, however additional work must be done before conducting scientific research. The above-mentioned unmet goals must obviously be attended to before anything, in order to have a fully functional application. Further, upgrades to the basic modules will also be necessary to achieve the rigor required for a proper scientific study.

A Annex : Sustainable development and corporate social responsibility

Inria research center has established a Local Sustainable Development Committee (CLDD), which role is to implement the actions outlined in the national sustainable development plan within the centre, while adhering to local constraints and taking account of the expectations of local stakeholders.

It is also responsible for raising the awareness of the centre's staff, as well as of service providers and partners of all kinds, and for developing the notion of sustainable development within the work and processes undertaken by Inria.

Created in September 2020 for an indefinite period of time, this commission takes over from the GT DD (Sustainable Development Working Group). This commission has no decision-making power, but more than a simple consultative body , it is intended to be both a force for proposals and concrete achievements.

In order to carry out the numerous projects, the commission has planned to meet once a month, including two meetings a year in extended mode, i.e. open to all (on the model of the center committee) and with the invitation of a representative of the management and the various key services.

In addition, the committee has planned a sub-project operation allowing the center to progress independently of the rhythm of the monthly meetings, while offering the possibility to any agent of the center who would be interested in participating in a specific theme or project, to take actions at its influence level.

This commission also works in networks to share good ideas and have more impact on the actions carried out, whether locally (university campus...), in other Inria centers or at the national level.

The commission has identified the following different themes, as a continuation of the actions started by the members of the working group. Two transversal themes:

- Awareness raising and communication: organization of events, communication material (guides, newsletters, etc);
- Diagnosis and monitoring of the impacts of the actions carried out: CO2 and energy balance of the building, quantification of inputs and waste.

Five themes on the functioning of the center:

- Computer and electronic equipment: internal and external second-hand cycle;
- Mobility (daily and mission): proposals for a better layout of the bicycle parking lots, bicycle workshops ;
- Energy consumption of the building and servers;

- Zero waste objective;
- Power supply.

References

[1] Evangelos Alexiou, Peisen Xu, and Touradj Ebrahimi. Towards modelling of visual saliency in point clouds for immersive applications. In *2019 IEEE International Conference on Image Processing (ICIP)*, pages 4325–4329, 2019.

[2] Birgitta Burger, Anna Puupponen, and Tommi Jantunen. Synchronizing eye tracking and optical motion capture: How to bring them together. *Journal of Eye Movement Research*, 11(2), 2018.

[3] Juan J. Cerrolaza, Arantxa Villanueva, and Rafael Cabeza. Study of polynomial mapping functions in video-oculography eye trackers. *ACM Trans. Comput.-Hum. Interact.*, 19(2), July 2012.

[4] Viviane Clay, Peter König, and Sabine Koenig. Eye tracking in virtual reality. *Journal of Eye Movement Research*, 12(1), 2019.

[5] Asim H Dar, Adina S Wagner, and Michael Hanke. Remodnav: robust eye-movement classification for dynamic stimulation. *Behavior research methods*, 53(1):399–414, 2021.

[6] Lee Friedman, Ioannis Rigas, Evgeny Abdulin, and Oleg V Komogortsev. A novel evaluation of two related and two independent algorithms for eye movement classification during reading. *Behavior research methods*, 50(4):1374–1397, 2018.

[7] Jose Llanes-Jurado, Javier Marín-Morales, Jaime Guixeres, and Mariano Alcañiz. Development and calibration of an eye-tracking fixation identification algorithm for immersive virtual reality. *Sensors*, 20(17), 2020.

[8] Marcus Nyström and Kenneth Holmqvist. An adaptive algorithm for fixation, saccade, and glissade detection in eyetracking data. *Behavior Research Methods*, 42(1):188–204, 2010.

[9] Thies Pfeiffer. Measuring and visualizing attention in space with 3d attention volumes. In *Proceedings of the Symposium on Eye Tracking Research and Applications*, ETRA '12, page 29–36, New York, NY, USA, 2012. Association for Computing Machinery.

[10] Thies Pfeiffer, Marc E. Latoschik, and Ipke Wachsmuth. Evaluation of binocular eye trackers and algorithms for 3d gaze interaction in virtual reality environments. *JVRB - Journal of Virtual Reality and Broadcasting*, 5(2008)(16), 2008.

[11] Thies Pfeiffer and Cem Memili. Model-based real-time visualization of realistic three-dimensional heat maps for mobile eye tracking and eye tracking in virtual reality. In *Proceedings of the Ninth Biennial ACM Symposium on Eye Tracking Research & Applications*, ETRA '16, page 95–102, New York, NY, USA, 2016. Association for Computing Machinery.

[12] Fiora Pirri, Matia Pizzoli, Daniele Rigato, and Redjan Shabani. 3d saliency maps. In *CVPR 2011 WORKSHOPS*, pages 9–14, 2011.

[13] Dario D. Salvucci and Joseph H. Goldberg. Identifying fixations and saccades in eye-tracking protocols. In *Proceedings of the 2000 Symposium on Eye Tracking Research & Applications*, ETRA '00, page 71–78, New York, NY, USA, 2000. Association for Computing Machinery.

[14] James Simpson. Three-dimensional gaze projection heat-mapping of outdoor mobile eye-tracking data. *Interdisciplinary Journal of Signage and Wayfinding*, 5(1):62–82, 2021.

[15] Vincent Sitzmann, Ana Serrano, Amy Pavel, Maneesh Agrawala, Diego Gutierrez, Belen Masia, and Gordon Wetzstein. Saliency in vr: How do people explore virtual environments? *IEEE Transactions on Visualization and Computer Graphics*, 24(4):1633–1642, 2018.

[16] Junle Wang, Matthieu Perreira Da Silva, Patrick Le Callet, and Vincent Ricordel. Computational model of stereoscopic 3d visual saliency. *IEEE Transactions on Image Processing*, 22(6):2151–2165, 2013.

[17] Xi Wang, Sebastian Koch, Kenneth Holmqvist, and Marc Alexa. Tracking the gaze on objects in 3d: How do people really look at the bunny? *ACM Transactions on Graphics*, 37(6), December 2018.

Figures list