A Framework for Pervasive Visual Deficiency Simulation

Christoph Schulz* Nils Rodrigues* Marco Amann Daniel Baumgartner Arman Mielke
Baumann, Christian Michael Sedlmair* Daniel Weiskopf*

University of Stuttgart

ABSTRACT

We present a framework for rapid prototyping of pervasive visual deficiency simulation in the context of graphical interfaces, virtual reality, and augmented reality. Our framework facilitates the emulation of various visual deficiencies for a wide range of applications, which allows users with normal vision to experience combinations of conditions such as myopia, hyperopia, presbyopia, cataract, nyctalopia, protanopia, deuteranopia, tritanopia, and achromatopsia. Our framework provides an infrastructure to encourage researchers to evaluate visualization and other display techniques regarding visual deficiencies, and opens up the field of visual disease simulation to a broader audience. The benefits of our framework are easy integration, configuration, fast prototyping, and portability to new emerging hardware. To demonstrate the applicability of our framework, we showcase a desktop application and an Android application that transform commodity hardware into glasses for visual deficiency simulation. We expect that this work promotes a greater understanding of visual impairments, leads to better product design for the visually impaired, and forms a basis for research to compensate for these impairments as everyday help.

Index Terms: Human-centered computing—Visualization—Visualization systems and tools; Human-centered computing—Accessibility—Accessibility systems and tools

1 INTRODUCTION

There are various approaches to visual disease simulation [3] and compensation [20]. While these provide computational models for various aspects of the visual system, they are not openly accessible or difficult to adapt to new emerging hardware due to technical limitations. Specifically, for deficiency emulation, immersive technologies such as augmented and virtual displays offer a great opportunity for more realistic simulated experiences that are likely to outperform existing approaches for grasping differences in perception [1, 19].

When exploring these new technologies, we encountered several technical limitations of existing approaches: Most research prototypes focus on one deficiency at the time. However, a person might suffer from multiple visual deficiencies at the same time. Even if there is support for multiple aspects of human vision, technical hurdles can impede portability.

To overcome these limitations, we have developed an open source framework for visual system simulation in Rust, a modern systems programming language. Our framework does not require any virtual machine environment and has full GPU support without imposing hard dependencies on platform-dependent APIs such as DirectX or OpenGL. While omitting GPU support would have simplified the software design considerably, virtual reality scenarios would have suffered from low frame rates that quickly lead to dizziness [15]. Depending on the research focus, the application area, and the desired properties of a software system for visual system simulation, requirements may vary widely. Nevertheless, we have identified several non-functional requirements across many application domains: *ease of use*, *performance*, *extensibility*, and *portability*.

Thus, we consider our framework more reusable and exceeding the possibilities of simple research prototypes. Our intention is to simplify the creation of content for researchers and the general public by providing real-time-rendered images to explain differences in perception. Moreover, we expect to facilitate research and development of visual deficiency simulation and compensation. In this spirit, we provide an augmented reality application as design aid along with our framework that allows users to experience visual deficiencies, so that individuals with good vision may be able to recognize these impairments and take them into account, e.g., when designing products. Our implementation of the framework and applications are publicly available on GitHub[1].

*e-mail: firstname.lastname@visus.uni-stuttgart.de

2 RELATED WORK

Here, we only deal with physiological and technical aspects of our framework, since there are already other works that focus on lowering the barrier for replication and evaluation of visualization techniques in general [2, 14].

Visual System Simulation For designing our simulator, we relied on computer graphics and medical literature: The book by Patzelt [17] provides a comprehensive overview of ophthalmology. Fink et al. [9] successfully ray traced high ametropia as early as 1996. The first comprehensive framework for visual eye disease simulation known to us was developed by Banks and McCrindle [3]. Their framework depends on Microsoft-specific technology and thus lacks first-class support for widespread mobile devices and the web platform in general, i.e., HTML5. Hence, their simulator is neither portable nor open source. One year later, Truscott [18] investigated alternatives to blurring to simulate presbyopia. Nießner et al. [16] realized a simulation of human vision through eyeglasses. We have taken these considerations as a basis for ray tracing the entire process, i.e., from light hitting the eye all the way through the retina.

Deficiency Compensation Even though visual deficiency compensation is not the focus of this work, we discuss the topic here since it concerns the application of our framework. Liou [13] defined a model to predict eye performance depending on parameters such as contrast and defocus in 1996. Therefore, a quantitative comparison of healthy and defective vision seems possible.

With the release of new head-mounted displays, several groups have already built systems as every-day help: For example, Lausegger et al. [11] build a smart-glasses-based system to support color-blind people. Essentially, their system shifts color computationally—similar to what EnChroma glasses do physically [7]. Huang [10] developed a HoloLens application that aids people in navigation tasks by detecting small text and re-rendering it at appropriate sizes. One of the most radical, non-intrusive examples we know is Third Eye [20]. This system aids in shopping by translating vision into vibrations that can be felt with a glove.

[1] https://github.com/UniStuttgart-VISUS/visual-system-simulator

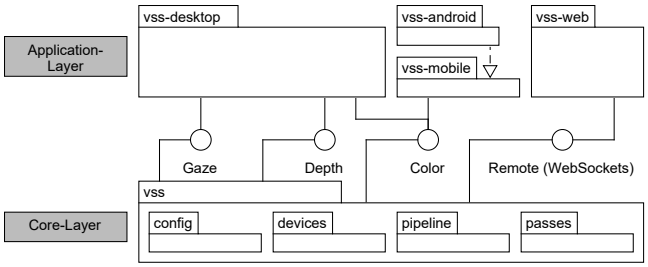


Figure 1: UML component diagram showing the core and application layers with various interfaces in between.

Simulator Sickness Due to the Android application, we also looked into simulator sickness, since we wanted to avoid that users become unnecessarily sick. Barrett [5] and Lawson [12] assess in their reviews that more than two-thirds of participants experience adverse symptoms. Moss and Muth [15] conclude that occluding peripheral vision seems to be an important factor for triggering simulator sickness. Therefore, we decided to implement a hand-held mode as well as a head-mounted mode, although the latter seemed to be more promising for immersion.

Moreover, we looked into actively combating simulator sickness. One idea was to adapt the intensity of the simulation depending on additional measurements. Unfavorably, there seems to be no connection between non-intrusive measurements such as eye movement and simulator sickness [4]. However, strategic and automatic modification of the field of view during a session is a promising approach [8].

3 Simulation System

Based on the observed limitations and desired applications above, we have inferred a set of high-level requirements:

R1 Accessiblity Developers, researchers, and users want a low barrier to entry so that they can easily benefit from our framework.

R2 Flexiblity Developers want easy integration, extensibility, and portability to keep costs low.

R3 Validity Researchers want a simulation that reflects reality to make valid claims.

R4 Responsiveness Users want to experience low-latency simulation to prevent dizziness and foster immersion.

While we can achieve accessibility (R1) and flexibility (R2) simultaneously, there are also competing requirements that need more consideration: On the one hand, the simulation should be valid in the physiological sense (R3), which implies modeling and computation of inherently complex aspects such as light. On the other hand, the simulation should be responsive (R4), which imposes tight latency budgets. The compromise we found here is that we leverage the power of GPUs and only simulate the relevant aspects of light. The technical basis for efficient and reusable GPU-based software lies in good software design, i.e., zero cost abstractions and well-defined interfaces. To this end, we have divided our software system into multiple layers (Figure 1): At the core layer, there is a reusable, platform-independent package that contains components for configuration, remote control, input-output devices, the simulation pipeline, and its rendering passes. At the application layer, which builds on the core layer, there are platform-dependent packages for integration into the respective platforms.

3.1 Core Layer

The core layer is bundled into a library that defines interfaces for the simulation while providing common functionality such as configuration. This ensures easy setup for user studies, better reproducibility, and thus is useful for testing and research in general.

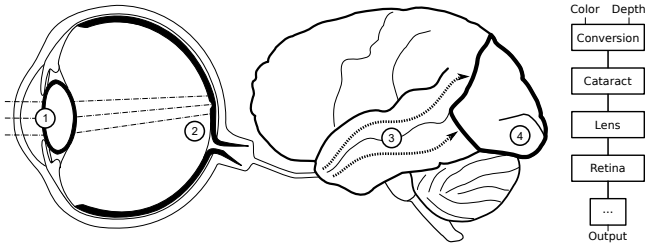


Figure 2: Physiological pipeline of the human visual system and the corresponding simulator pipeline. In human eyes (left), light passes the lens system (1), and then hits cells on the retina (2). These cells translate light into nerve signals that are propagated through the visual pathway (3), all the way to the visual cortex (4). Our simulation pipeline (right) approximates this process.

Additionally, our remote-control protocol allows for inspection and reconfiguration of the simulation during operation. This is useful for cross-device setups where an operator monitors and controls the experiment. Another important factor is documentation—not just for developers, but also for users. That is why we decided to keep configuration and description of visual deficiencies closely coupled to form a kind of knowledge base.

Implementation-wise, `vss` is a Rust library that depends on `gfx`[2] for platform-independent graphics and other libraries. For configuration, we use a JSON-based format that describes which device (image, video, camera) should be used and which simulation aspects (passes) should be applied and how (parameters). The remote-control protocol is realized using WebSockets. The simulation pipeline is composed of multiple rendering passes that are arranged to match the processing stages of the visual system (Figure 2). We opted to design the communication between these passes as swapchain, i.e., we only require two intermediate render targets.

Color-space conversion At the beginning of the pipeline, there is a pass for color-space conversion: While we usually have 8-bit sRGB color and depth textures for image devices, the situation is completely different for video and camera devices where we have to combine three independent 8-bit YUV textures, each representing one channel. Then the actual simulation passes follow; currently cataract, lens, and retina.

Lens simulation In general, we simulate aspects of the lens system using ray tracing. The cataract pass plays a special role and thus is split apart as a pre-processing step. The lens pass has various 1D and 2D parameters such as overall lens geometry and a cornea map. Note that the cornea is not part of the lens but part of the lens system and thus implemented in this pass. The corneal map is used to simulate deformations of the cornea, e.g., to give an impression of corneal diseases regarding deflection of light. Next, the light rays hit the retina.

Retina simulation We have kept this pass relatively simple for now—conceptually more similar to a CCD sensor—with parameters such as tint and a retina map. The retina map describes the distribution and sensitivity of rod and cone cells. We provide procedural functions to pre-compute this map for simulating glaucoma, macular degeneration, and achromatopsia. However, it is possible to specify a hand-drawn retina map.

At the end of the pipeline, the result is then displayed or written to a file by the device. If new aspects should be added to the simulation, a developer may either modify existing passes or add a new one.

[2] https://github.com/gfx-rs/gfx

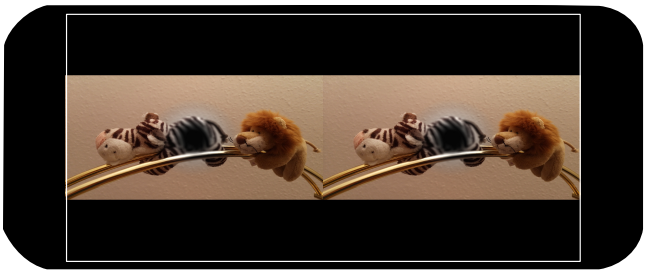


Figure 3: Simulation of macular degeneration in Google Cardboard mode using the back camera to achieve video-see-through.

3.2 Application Layer

To guide the development and demonstrate the usefulness of our framework, we envisioned multiple application scenarios: (1) A desktop application that is useful for research and development of the framework. (2) A mobile application as portable looking-glass demonstrator. (3) A generic post-processing-hook that can be injected into other applications for usability testing. (4) An interactive web application for the general public. All of these options are potentially compelling for users. However, we wanted something less technically complicated, easy to test, and with potential support for augmented or virtual reality hardware. Thus, we selected the first two scenarios, as the third one is technically difficult and the fourth one implies cross-browser testing.

Desktop App Our primary goal was to demonstrate how easy it is to set up prototypes, e.g., for studies. Thus, the desktop application only features a command line interface and relies on the generic web interface `vss-web` for interactive reconfiguration. Note that having the configuration out-of-process turned out to be an advantage during development since the configuration is not lost if the application restarts. All in all, the code boils down to a few lines of code for configuration, device creation, event processing, and some platform-specifics. We hope this conveys a low entry barrier.

Android App Our goals for mobile simulation were more ambitious: We wanted a fully-fledged application to publish on Play Store for inexperienced users with video-see-through, a native-looking user interface, and a knowledge base for educational purposes. Furthermore, we have implemented a Google Cardboard mode (Figure 3) so that users can experience visual deficiencies with a low-cost head mounted display—this usually requires drilling a small hole into the smartphone mount so that the light from outside reaches the camera.

The Android application is considerably more complex because of performance requirements on mobile devices and required non-negligible effort to achieve a native user interface. A particular engineering hurdle was the foreign function interface between Rust and Java to achieve zero-copy access to the camera data. In contrast, we were able to generate the integrated knowledge base directly from the documentation without duplicated work.

4 Examples

In this section, we demonstrate multiple visual deficiencies and combinations thereof that we can simulate using our framework. We list various off-the-shelf examples in this section and show the simulation results in Figure 4 and Figure 5. The physiologically motivated passes of our framework (cf. Section 3) allow for simulation of more deficiencies by simply adjusting the configuration. Thus, the following examples are just a coarse sampling of all possible configurations. For tracing implementation details and reproducibility, all configurations can be found on GitHub.

4.1 Cataract

Denatured proteins in the eye's lens become less transparent, leading to clouding. In pronounced cases, this can be seen externally, making the lens appear milky. The clouding not only hinders light on its way through the lens but also scatters it. The results are faded colors, blurred vision, increased sensitivity to strong light sources (glare), and difficulties in dark surroundings [6, p. 338].

We start with an original unchanged image (Fig. 4a). It shows the scene as it would look like without any defects. The cataract pass of our simulation framework applies blurring as well as color fading to this input image. Both effects are adjustable using parameters that subjectively represent ranges from weak to severe defects. Weak settings lead to a mostly blurred image (Fig. 4b), whereas more severe cases also exhibit faded colors Fig. 4c.

4.2 Myopia and Hyperopia

Refractive errors (ametropia) are the most common deficiencies with 1–2 billion people being affected world wide [6, p. 926]. Widely known among them are near- (myopia) and farsightedness (hyperopia). Both result from a mismatch of the eye's length and too strong or weak refraction along the light path, for instance in the cornea or lens. Thus, the focal point moves in front of or behind the retina. This offset varies strongly with the distance of viewed objects.

Our lens pass requires three inputs to simulate both deficiencies: (1) The output image of the cataract pass. We use the artificially rendered Fig. 5a as an example and assume that there were no cataracts. (2) A depth map to obtain the distance corresponding to each color pixel. We can produce such a map from the artificial scene for demonstration purposes (Fig. 5b). In practice, users could also import depth maps from external systems such as stereo cameras. (3) The degree of near- and farsightedness in units of diopters. This information is readily available to any wearer of glasses, making it an easily adjustable parameter. Users of our framework can simulate their sight intuitively and compare with one another due to the well-known unit system.

Fig. 5c shows the result for nearsightedness with −2 dpt. Farsightedness makes objects appear less sharp, the closer they are. Simulating it with +2 dpt leads to the result in Fig. 5d. Myopia and hyperopia are opposing effects of the same cause. They are mutually exclusive and the framework does not allow for a combination of both. Therefore, the parameter sign indicates what defect to simulate.

4.3 Presbyopia

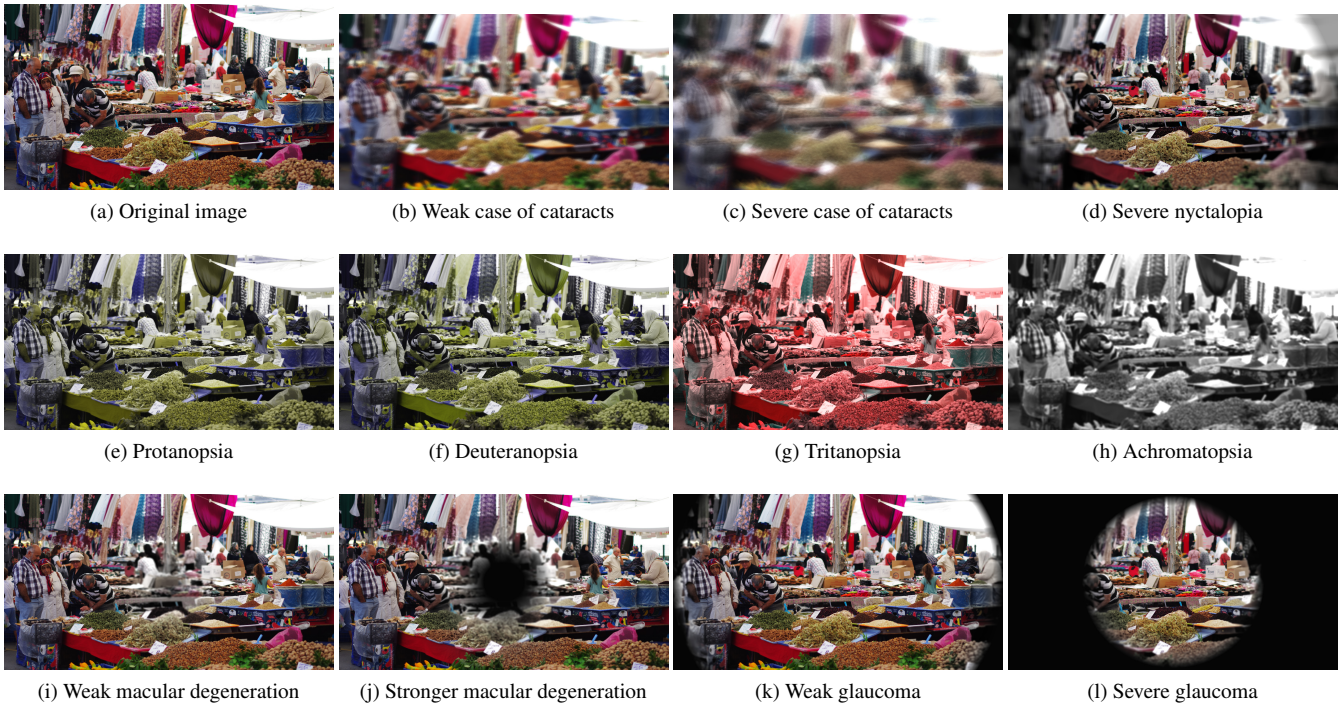
With increasing age, the human eye gradually loses the ability to accommodate for different focal distances. This is due to protein changes in the lens, making it less flexible [18]. The closest focusable distance (near point) increases, making closer objects appear blurred. Everything beyond the near point is still projected normally onto the retina.

Our framework simulates presbyopia similarly to near- and farsightedness. It requires a scene image and a depth map. The adjustable near point serves as an external parameter and decides what portions of the image become affected. Parts that are closer suffer from blur, while farther segments remain unchanged. Using the near point as a parameter is very intuitive as users can easily measure it themselves by simply positioning objects—such as a finger—as close as possible while still seeing it sharply.

As presbyopia is caused by stiffness of the lens, it can appear in combination with near- and farsightedness. Our framework allows combining both refractive errors, as well as other additional defects.

4.4 Nyctalopia

Night blindness can have various causes that lead to slightly different effects. They all have in common that sensitivity to low light is reduced, leading to darkness in insufficiently lit areas in the visual



(a) Original image (b) Weak case of cataracts (c) Severe case of cataracts (d) Severe nyctalopia

(e) Protanopsia (f) Deuteranopsia (g) Tritanopsia (h) Achromatopsia

(i) Weak macular degeneration (j) Stronger macular degeneration (k) Weak glaucoma (l) Severe glaucoma

Figure 4: Simulation of non-refractive defects on an original unchanged image (a). Cataracts (b, c) showcase the first pass of our simulation. Various retinal maps in the last pass can simulate other vision disorders (d–l).

field. Depending on the cause, for instance, loss of peripheral vision, the affected area of the visual field might vary from person to person.

Our framework accounts for this by using a map of the retina in its retina pass. This input is either supplied by external sources or generated by setting a strength parameter in percent. To provide an example, we use the original picture in Figure 4 and assume there are no other deficiencies. The input image does not contain dark areas but the output in Fig. 4d shows that people with strong nyctalopia are still affected.

4.5 Protanopia, Deuteranopia, Tritanopia, and Achromatopsia

A healthy retina has cone cells as detectors for short (blue), medium (green), and long (red) wavelength light. Anomalies in their functionality, numbers, or distribution cause deficiencies in color vision. For instance, partial or full loss of function in red detector cells leads to protanomaly and protanopia (Fig. 4e) respectively. Loss of signals from green cells causes deuteranomaly and deuteropia (Fig. 4f). Both deficiencies hinder the distinction of red and green color in the human visual system, prompting the common name of *red-green blindness*. The defects result in similar color perception but are still distinguishable through subtle differences. Anomalies concerning receptors for blue light cause tritanomaly or tritanopia (Fig. 4g), also known as *blue-yellow blindness*. Missing or non-functioning detectors for all three wavelengths removes all color information, leaving only a monochromatic sensory impression (achromatopsia or color blindness). As shown in Fig. 4h, having no functioning color receptors in the eye's macula affects not only color perception but also visual acuity.

The retina pass handles all these deficiencies simultaneously. Color information from the input picture is transformed according to a retinal map. This map is either supplied externally or generated from three parameters that correlate to each color channel. The parameters allow for constant deficiencies between 0 and 100 percent of impairment. External maps are better suited for simulation of anomalous cone cell distributions that might result from measurements with real humans.

Our framework allows for simulating a combination of all types of night and color blindness with a single retinal map: We encode the three cone cell types into RGB-color channels and rod cells into the transparency channel. Thus, regions of the retina with too little cells can be subject to loss of color (cones cells) and blurring (rod cells).

4.6 Macular degeneration

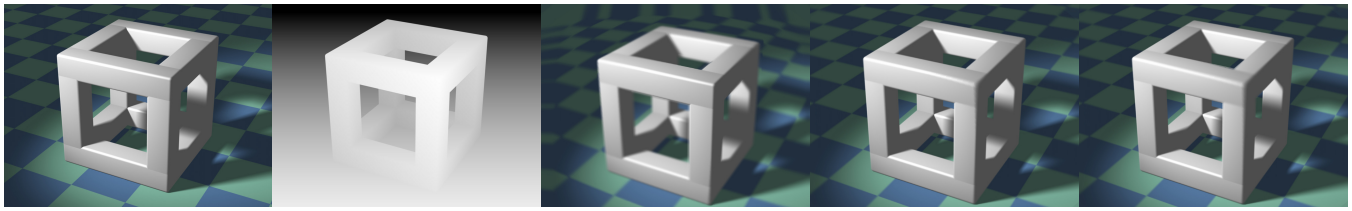
Progressive degeneration of the eye's macula leads to blurring and color-fading in the center of the visual field. Peripheral vision remains unaffected. We can simulate this defect with our framework by using the same retinal map as for color deficiencies (cf. Subsection 4.5). Blending multiple maps onto a single texture combines deficiencies and thus results in an efficient simulation.

The retinal map either comes from an external source or the framework generates it according to one of two schemes: The advanced method requires the adjustment of two parameters: A radius that defines the affected area and a strength factor that indicates how many cells will be removed. The simple method derives both parameters from a single severity setting in percent.

Fig. 4i is an example of weak macular degeneration. The center part only shows a loss of acuity and color. A stronger case, as in Fig. 4j, has a blind spot with a complete lack of visual information.

4.7 Glaucoma

Glaucoma is a loss of vision from damage to the optic nerve that itself can have various root causes. It presents as blindness in the outer visual field. Although the causes of glaucoma do not have to lie within the eye, the framework can simulate its effects using the retinal map. Whether the signals got lost during transmission (in the nerve) or were never generated (in the receptor cells), the result is always the same: the image does not arrive in the brain.



(a) Original color map (b) Original depth map (c) Nearsightedness at -2 dpt (d) Farsightedness at $+2$ dpt (e) Presbyopia (n. p. at 60 cm)

Figure 5: Simulation of refractive errors. Our lens pass requires an original image (a) and a depth map (b) as input. It is then able to simulate adjustable degrees of near- (c) and farsightedness (d), as well as presbyopia (e).

Our simulation framework provides automatic methods for generating a retinal map similar to the one for macular degeneration (cf. Subsection 4.6), although this time it is inverted. An example for a weak setting is in Fig. 4k, while Fig. 4l is more severe. As previously noted, blending this deficiency with existing retinal maps also allows for efficient simulation.

5 Discussion

In comparison to previous approaches, our framework is open source under a very liberal license (Apache 2.0), robust, and fast because of Rust's safety and zero-cost guarantees, easily extensible and portable because of our overall software design. However, the Rust ecosystem was still massively evolving during development, i.e., we had to adapt to changing APIs. The promise of Rust "if it compiles, it should not crash" compensated for such issues quite well.

Validation and Evaluation We left it open to future work to fully validate our simulation with real users for multiple reasons: (1) We wanted to split implementation, evaluation, and validation as proposed by Lücke-Tieke [14]. (2) Our framework is meant to be used by other researchers. (3) Validating our simulations and (4) evaluating the usefulness of our framework within applications in everyday life are two different things that would go beyond the scope of this work. Nevertheless, we partially succeeded in validating our simulation results through reference literature. During that process, it quickly became clear that real users might get nauseous from experiencing such a simulation in immersive environments. Thus, conducting a larger user study could present an ethical problem because of the risk-benefit ratio. Moreover, neither people with eye diseases nor those with normal vision would be able to tell if the simulation is correct since they lack a common reference. What complicates validation more, are possible interactions between visual deficiencies. For instance, strabismus and bad target stabilization both impact depth perception. Ideally, we could collect real-world usage data through the Android application.

There are also some physiological aspects that we omit in our simulation, such as the refraction of light at the cornea since we could not find any information about that parameter space in literature. Similarly, we had to learn some parameters using a particle swarm optimization algorithm—the resulting image looks correct but individual parameters are not always physiologically plausible.

Technology Another difficulty lies in physical quantities and measurement: What is measured on a camera sensor is rarely a perfect representation of light in sRGB color space. Usually, the measurement is represented using some proprietary RAW color space, already neglecting uncertainty of the measurement (accuracy and precision). This representation is then subject to lossy conversion into sRGB, introducing even more uncertainty with respect to the light that hit the camera sensor.

In particular, we hope that HDR images might help to better represent sources of light in image space, which is important for glare effects. Furthermore, we expect that in the near future stereo cameras and eye trackers will become part of commodity augmented and virtual reality hardware. This would allow us to simulate even more aspects correctly due to the measured depth and gaze information. Furthermore, we envision validation and illustration of foveated rendering techniques as an application for our framework.

Compensation Now that we have a computational model for various visual deficiencies, the next step would be to develop and improve techniques for compensation. Such techniques allow for research questions that can be meaningfully evaluated by checking for benefits in everyday life. Additionally, this type of research is less ethically problematic due to its better risk-benefit ratio.

6 Conclusion

We have presented a framework that allows rapid prototyping of visual deficiency simulations in pervasive computing environments. The primary motivation behind our framework is to foster the development and research of such simulations. To showcase this aptitude, we have developed two applications: A simple desktop application and a fully-fledged Android application. To demonstrate the applicability, we showcased several visual deficiencies that can be simulated using our framework.

In future work, we want to incorporate support for stereo-cameras, eye trackers, and more aspects of the human visual system. Regarding the simulation, we would like to add support for processes that tend to happen in the brain. For example, we would like to implement a target stabilization pass at the end of the pipeline for simulating an impaired vestibuloocular reflex (VOR) to better understand the influence of vision on body balance. Simulating brain-related processes would allow users to get a glimpse of the visual-ants syndrome, alcohol consumption and the vision-related issues of the Parkinson syndrome. We expect that this framework will help researchers to conduct studies: While it is obvious that not everyone has perfect sight, we see also applications for deficiency compensation, vision training, and validation of visualization techniques.

Acknowledgments

This work was funded by the German Research Foundation (DFG, Deutsche Forschungsgemeinschaft) within projects A01 and B01 of SFB-TRR 161 (ID: 251654672). We also thank Jingxi Zhang, Kanan Allahyarli, Maximilian Korn, Noel Schäfer, and Simon Braitsch for their contributions to the academic project that led to this framework.

References

[1] R. Addison. Detour: Brain deconstruction ahead. *IEEE Computer Graphics and Applications*, 15(2):14–17, 1995. doi: 10.1109/38.364998

[2] W. Aigner, S. Hoffmann, and A. Rind. EvalBench: A software library for visualization evaluation. *Computer Graphics Forum*, 32(3pt1):41–50, 2013. doi: 10.1111/cgf.12091

[3] D. Banks and R. McCrindle. Visual eye disease simulator. In *7th International Conference on Disability, Virtual Reality and Associated Technologies (ICDVRAT 2008)*, pp. 167–174, 2008.

[4] K. Barajas. *Do Individual Differences in Eye Movement Scanning Predict Simulator Sickness?* PhD thesis, Florida State University, 2014.

[5] J. Barrett. Side effects of virtual environments: A review of the literature. *Information Sciences Laboratory*, p. 54, 2004.

[6] A. K. O. Denniston and P. I. Murray, eds. *Oxford Handbook of Ophthalmology*. Oxford University Press, mar 2018. doi: 10.1093/med/9780198804550.001.0001

[7] EnChroma Inc. EnChroma color blind glasses. `https://enchroma.com/`, 2019. Accessed 2019-02-01.

[8] A. S. Fernandes and S. K. Feiner. Combating vr sickness through subtle dynamic field-of-view modification. In *IEEE Symposium on 3D User Interfaces*, pp. 201–210, 2016. doi: 10.1109/3DUI.2016.7460053

[9] W. Fink, A. Frohn, U. Schiefer, E. Schmid, N. Wendelstein, and E. Zrenner. Visuelle abbildung bei hohen ametropien - computergestützte simulation mittels strahlenoptischer rechnungen. *Klinische Monatsblätter für Augenheilkunde*, 208(06):472–476, 1996. doi: 10.1055/s-2008-1035266

[10] J. Huang. A hololens application to aid people who are visually impaired in navigation tasks. Technical report, Dartmouth College, 2017.

[11] G. Lausegger, M. Spitzer, and M. Ebner. OmniColor a smart glasses app to support colorblind people. *International Journal of Interactive Mobile Technologies (iJIM)*, 11(5):161, 2017. doi: 10.3991/ijim.v11i5.6922

[12] B. D. Lawson. Motion sickness symptomatology and origins. In *Handbook of Virtual Environment: Design, implementation, and applications*, pp. 532–587. 2014. doi: 10.1201/b17360-29

[13] H.-L. Liou. *Optical modelling of visual performance*. PhD thesis, University of Melbourne, 1996.

[14] H. Lücke-Tieke, M. Beuth, P. Schader, T. May, J. Bernard, and J. Kohlhammer. Lowering the barrier for successful replication and evaluation. In *Proceedings of the Beyond Time and Errors on Novel Evaluation Methods for Visualization - BELIV '18*, 2018.

[15] J. D. Moss and E. R. Muth. Characteristics of head-mounted displays and their effects on simulator sickness. *Human Factors: The Journal of the Human Factors and Ergonomics Society*, 53(3):308–319, 2011. doi: 10.1177/0018720811405196

[16] M. Nießner, R. Sturm, and G. Greiner. Real-time simulation and visualization of human vision through eyeglasses on the gpu. *Proceedings of the 11th ACM SIGGRAPH International Conference on Virtual-Reality Continuum and its Applications in Industry - VRCAI '12*, p. 195, 2012. doi: 10.1145/2407516.2407565

[17] J. Patzelt. *Basics Augenheilkunde*. Elsevier, Urban & Fischer, 2005.

[18] R. J. Truscott. Presbyopia. Emerging from a blur towards an understanding of the molecular basis for this most common eye condition. *Experimental Eye Research*, 88(2):241–247, 2009. doi: 10.1016/j.exer.2008.07.003

[19] J. Väyrynen, A. Colley, and J. Häkkilä. Head mounted display design tool for simulating visual disabilities. In *Proceedings of the 15th International Conference on Mobile and Ubiquitous Multimedia*, pp. 69–73, 2016. doi: 10.1145/3012709.3012714

[20] P. A. Zientara, S. Lee, G. H. Smith, R. Brenner, L. Itti, M. B. Rosson, J. M. Carroll, K. M. Irick, and V. Narayanan. Third Eye: A shopping assistant for the visually impaired. *Computer*, 50(2):16–24, 2017. doi: 10.1109/MC.2017.36