Visualization Research Center

University of Stuttgart
Allmandring 19
D–70569 Stuttgart

Masterarbeit

Visualization of Differences in Perception Caused by Vision Deficiency

Marco Amann

Course of Study:	Softwaretechnik
Examiner:	Prof. Dr. Michael Sedlmair
Supervisor:	Dipl.-Inf. Christoph Schulz, Katrin Angerbauer M.Sc.
Commenced:	October 12, 2020
Completed:	April 12, 2021

Abstract

Simulation of visual impairments may be used to counter design exclusion and improve the accessibility of products. Whilst first-hand experience of the simulated scene is important, having to manually look for differences caused by vision deficiencies, let alone judge their severity, is time-consuming and error-prone. In this work, we develop an error metric that tracks errors and uncertainty throughout the stages of a simulator for visual impairments. This allows us to visualize errors and uncertainty independently in all tracked dimensions. We add more advanced view modes to the simulator to enable visualization of our error metric in combination with the input and output images. We further extend the simulator to include simulations for strabismus, astigmatism and retinal ganglion cells. By simulating several combinations of vision deficiency, we found that a visualization of the proposed metric can be used to identify problematic areas in a scene. Depending on the use case, one may need to select different combination functions generating scalar values, e.g. the aggregated standard deviation of the RGBXY vector. With a comparison of our metric to SSIM, we found that our metric can cope better with displaced features of an image but may produce blurred visualizations. We conclude that although our metric imposes considerable performance penalties on the simulator, it has advantages compared to approaches based exclusively on input and output images.

Kurzfassung

Simulation von Fehlsichtigkeiten kann dabei helfen, ausgrenzendes Design zu verhindern und die Barrierefreiheit von Produkten zu verbessern. Während die Erfahrung der simulierten Szene wichtig bleibt, ist es zeitaufwendig und fehleranfällig, durch Fehlsichtigkeiten verursache Unterschiede zu erkennen oder deren Schwere zu beurteilen. In dieser Arbeit entwickeln wir eine Metrik, die Fehler und Unsicherheit durch die einzelnen Schritte der einer Fehlsichtigkeitssimulation hindurch verfolgt. Dies erlaubt uns, Fehler und Unsicherheit aus allen aufgezeichneten Dimensionen zu visualisieren. Wir fügen fortgeschrittene Anzeigemodi zum Simulator hinzu, sodass unsere Metrik in Kombination mit dem Eingabe- oder Ausgabebild visualisiert werden kann. Des Weiteren wird der Simulator um Strabismus, Astigmatismus und eine Simulation der retinalen Ganglienzellen erweitert. Durch das Simulieren verschiedener Kombinationen von Fehlsichtigkeiten konnten wir feststellen, dass die Visualisierung unserer Metrik dazu verwendet werden kann, problematische Regionen in einer Szene zu identifizieren. Abhängig vom Anwendungsfall müssen womöglich andere Kombinationsfunktion ausgewählt werden, die einen Skalar erzeugen, beispielsweise die aggregierte Standardabweichung des RGBXY Vektors. Durch einen Vergleich unserer Metrik mit SSIM konnten wir feststellen, dass unsere Metrik besser mit verschobenen Bildelementen klar kommt, aber unter Umständen verschwommene Visualisierungen erzeugt. Wir schlussfolgern aus dieser Arbeit, dass, obwohl die Metrik starke Laufzeitkosten verursacht, sie Vorteile gegenüber Ansätzen hat, die ausschließlich auf dem Vergleich der Eingabe- und Ausgabebilder basieren.

Contents

List of Figures

Acronyms

HVS Human Visual System. 13

IPD Interpupillary Distance. 45

mRGC Midget Retinal Ganglion Cell. 39

MSE Mean Squared Error. 23

RGC Retinal Ganglion Cell. 15

RGCf Retinal Ganglion Cell receptive Fields. 39

SSIM Structural Similarity. 23

VDP Visible Differences Predictor. 24

VSS Visual System Simulator. 13

1 Introduction

Eye diseases like myopia have shown to occur more frequently over the last decades [HFW+16; Vit09], while the exact reasons for the increase are not yet known [AWS18; PRS11]. Elderly people are especially prone to eye diseases that cause severe vision impairments [Har03].

If one has limited visual capabilities, everyday tasks like using a computer [KG95] or finding a store [SZA16] can become challenging.

At the same time, designers that shape the environments surrounding us are often younger than the people with severe eye diseases [CGC14]. This disparity necessitates a deep understanding of the limitations of people with vision impairments to design architecture, user-interfaces and appliances, that can be easily used by people with limited vision.

If products are developed for special groups of the population, the prevalence of certain types of eye diseases might be more severe: When a user interface for a device is designed, that is used exclusively by patients with diabetes, eye diseases like diabetic retinopathy are more common than in the rest of the population. Therefore, the device needs to fulfill very specific needs, that a designer might otherwise dismiss as only relevant for a tiny fraction of the population.

To raise awareness and foster the understanding of the challenges faced, simulators can be used [CGC14]. Further, having the ability to experience the effects of vision deficiencies first-hand, not only allows one to evaluate the own work, but also might be used to convince clients that certain decisions are exclusive.

To be able to quickly identify problematic parts of the observed design or object, it might be advantageous if the used visual deficiency simulator would not only output the simulated scene but also has means of visualizing and highlighting problematic regions. This removes the need to manually compare input and output images to detect problematic areas or judge their severity. When a designer evaluates two proposals in regards of usability despite simulated deficiencies, they might want to compare the detected problematic areas rather than manually comparing the simulated output images.

In this work, we develop a measure to describe the deflection, scattering and absorption of light in a simulator for vision deficiencies. This measure is meant to be able to capture perceived modification in every step of the simulator, rather than simple pixel-wise differences between the input and output images. Using this measure we can then visualize the differences caused by the deficiencies. Based on these metrics, highlighting is used to quickly attract the user's attention to problematic areas.

Further, we extend the existing simulator software to be able to visualize activation of retinal ganglion cells based on the approach outlined by Aleman et al. [AWS18]. This is a first step towards including the advanced processing and filtering capabilities of visual information in the

human visual system into the simulator. The simulator is further extended to be able to simulate astigmatism. To simulate strabismus, we first extend the simulator to have two independent eyes, enable these eyes to have a configurable distance to each other, and then implement the deficiency.

2 Background

This chapter covers topics that lay the foundations for the rest of this work. We present the parts of the Human Visual System (HVS) that are relevant for this work, as well as some selected visual impairments, either because they are already simulated in the Visual System Simulator (VSS) or need to be simulated in the course of this work. We then lay out the architecture of the VSS and describe some important implementation details.

2.1 Human Visual System

This section describes some of the components of the HVS that are simulated in the VSS. The basic function of the components described here is later used to explain the proposed distance metric and the changes to the VSS.

2.1.1 Anatomy of the Human Eye

For the explanation of the anatomy in this section, we refer to Bhattacharyya [Bha09, ch. 1]. The following paragraphs explain the relevant components of the human eye in the order incoming light interacts with them based on their work. The location of the components in the eye is shown in 2.1.

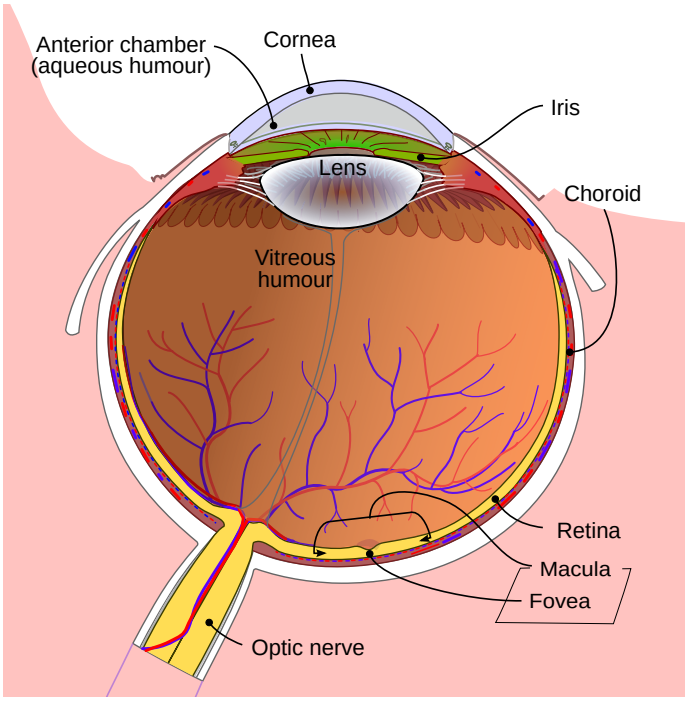


Figure 2.1: Schematic diagram of the human eye. Image from Wikimedia Commons [1].

Cornea The cornea is the first part of the eye in direction of incoming light, that is simulated by the VSS. The cornea is the main refractive medium with about 42 dpt. It consists of many layers, separating the anterior chamber from the outside world. The surface curvature and thickness of the cornea are decisive for correct projection onto the retina. If the curvature does not meet the requirements of the individual eye in the respective meridians, astigmatism can be observed. For a detailed description of astigmatism, refer to 2.2.1.

Anterior Chamber The anterior chamber is the space between the back side of the cornea and the lens. It is filled with liquid that allows for the lens to change its shape to enable accommodation. The anterior chamber also adjoins the iris. Whilst its connection to the lens allows light to pass through, the iris blocks incoming light.

Lens The Lens is a crystalline body, connected to the ciliar body. The muscles situated there allow the lens to change its shape. This changes the refractive strength of the lens and allows for accommodataion. The lens is transparent in a healthy eye. If protein accumulates in the retina, staining its purity, a cataract develops.

[1] CC-BY-SA https://commons.wikimedia.org/wiki/File:Schematic_diagram_of_the_human_eye_en.svg

Vitreous Humour The vitreous humour is a transparent substance that fills the largest part of the eyeball. Its transparent, soft structure allows for the lens to adjust itself, whilst remaining in contact.

Retina The Retina houses the photoreceptors in the eye and covers roughly the backside of the eyeball, having contact to the vitreous humour. It also contains the retinal ganglion cells. The retina has some special areas that are worth mentioning here: The macula is an area that is responsible for a majority of the human vision, since it contains the highest density of receptors. The fovea, located within the macula, is a small area with the highest density of receptive fields, allowing for the best resolution.

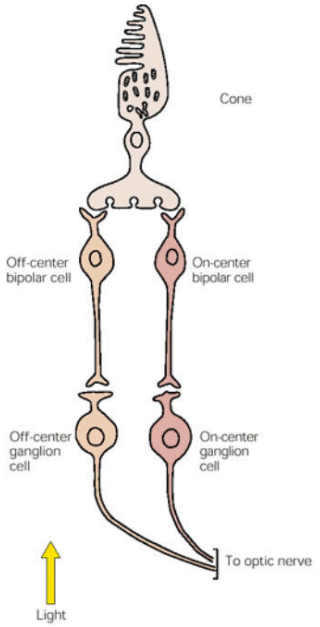
2.1.2 Retinal Ganglion Cells

To describe the structure and tasks of retinal ganglion cells in this section, we resort to the model taught by Kandel [Kan13]. Retinal ganglion cells are located in the retina, in direction of incoming light just before the photoreceptors. Most of the retinal ganglions are not photosensitive themself but receive information from the receptors they are attached to. To not block light from reaching the photoreceptors, the ganglion cells are mostly transparent. Retinal ganglion cells connect to bipolar cells, which are in turn connected to the photoreceptors. This simplification will be enough for the implementation of the ganglion cell simulation in later parts of this work. In the real retina, there are more connections and cell types, as depicted in 2.2b. Note that 2.2b also depicts how Retinal Ganglion Cell (RGC) are connected to their receptive fields: One ganglion cell can be connected to multiple photoreceptors and a single photoreceptor can be connected to multiple ganglion cells.

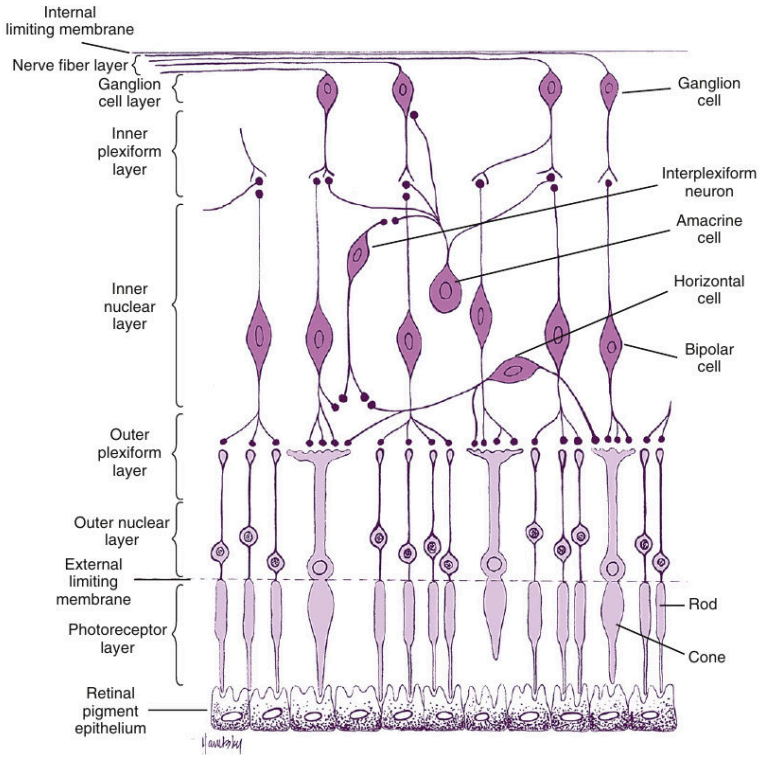
The output of the ganglion cells is forwarded to the brain via the optic nerve to filter and process visual information. There are two major types of retinal ganglion cells, P cells and M cells, which specialize in detection of small and large features. While M cells are used to detect contrasts in luminance, P cells are involved in the perception of color contrast. Whilst the above explanation will suffice as a basis for this work, it is worth noting that there are more than 30 different types of RGC that can be distinguished [SM15].

The density of retinal ganglion cells varies with eccentricity, as does the distribution of P Cells and M Cells [Bar99]. The density is highest in the fovea and decreases with greater eccentricity, although is quite hard to describe the exact decline [Wat14]. With increasing distance from the fovea, the size of the receptive fields increases: while in the fovea the receptive field of a ganglion spans a few minutes of an arc, it spans 3°-5°in the outer regions of the retina [Kan13]. This roughly follows the distribution of photoreceptors, which also have their highest density in the fovea and decrease in density towards the peripheral regions. A more detailed description of the shapes and sizes of RGC can be found in 4.3.

The way RGC work, can be summarized as taught by Kandel [Kan13]. The individual ganglion cells respond to differences in stimuli of the photoreceptors in their receptive fields. Two types of ganglion cells can be distinguished: Those that react when the center of their receptive field is stimulated with light while the perimeter is not (On-Center) and those that react in the opposite way (Off-center). By only reacting when there is a difference between the center and perimeter, retinal ganglion cells are able to detect contrast. This means that when presented with an image of



(a) Two ganglion cells of different types, connected to one cone via their respective bipolar cells. The separate pathways of On-Center and Off-Center cells are shown. Image modified from [Kan13, p 444].



(b) A detailed view of the connections in the retina. Image modified from [Rem12, p 71]

Figure 2.2: Schematic drawings of retinal ganglion cells and their connections.

constant luminance without any contrast detectable by the cells, the retinal ganglions remain mostly silent. Since the ganglions can detect contrast, they form a first processing and filtering step in the visual system.

One single photoreceptor can be connected to multiple ganglion cells, hence enabling parallel processing and forwarding of information collected by one rod or cone [Wäs04]. This enables selective processing at high speeds without losing information by assigning photoreceptors to individual ganglion cells. These conenctions are shown in figure 2.2a. When developing our RGC simulation, we will make use of this overlap.

On/Off pathways, forwarding the outputs of the ganglions, are organized mostly in separate layers in the retina, except a few outliers [HLMM09].

By blocking one of the pathways, detection of patterns can be inhibited: Schiller et al. [SSM86] observed that with blocked On-pathways, monkeys showed reduced capabilities to detect patterns that appeared lighter than their surroundings.

Studies have shown that by blocking one of the On-/Off pathways by injecting substances into the vitreous, eye growth can be influenced in animals [CC03]. Aleman et al. [AWS18] have further shown that stimulating dominantly On- or Off-ganglions by changing contrast polarity in images influences the thickness of the choroid in the human eye. They conclude that contrast polarity might influence eye growth in humans and hence be a factor for developing myopia.

2.2 Visual Impairments

Since we extend the functionality of the VSS to include more visual impairments, we summarize the underlying theory behind the diseases in this section. We restrict ourselves to only describe impairments that are newly implemented in this work. Therefore, this does of course not pose a comprehensive list.

2.2.1 Astigmatism

Astigmatism is a refractive error caused by unequal refraction along different meridians [DM14]. The difference in refractive power of the eye might be caused by several factors, some of which are: the curvature of the cornea, unequal refractive index or curvature of the lens as well as pressure on the lens or globe [Bha09]. In most cases, the principal meridians are 90° apart, although other constellations are possible and create an irregular astigmatism [Bha09]. The axis of the principal meridians most commonly are located in such a way, that the vertical meridian has the steepest curvature and hence the strongest refractive power [Rem12]. To cover all possible axis, the astigmatism is described as the spherical equivalent refractive error and the axis of the principal meridian with the steepest curvature [Goe93]. In some cases, the cornea exhibits some astigmatism but the rest of the optical system is correcting the deficits [MOR01]. In such cases, the person does not notice any problems with their vision.

A difference in refractive powers between the principal meridians causes light rays, originating from a point in the direction of one meridian, to be focused differently than these in direction of the other meridian [MOR01]. This creates two principal foci, of which one may lay on the retina, although this is not necessarily the case. By using a spherical correction, at least one of the foci can be placed on the retina. To correct the distance between the foci, toric lenses may be used [DM14]. The effects of astigmatism can be seen in 5.1.

If one focus point is located on the retina, an image of a line in the direction of the focused meridian is perceived correctly, with the exception of its ends. However, a line in a different angle is perceived as blurred. This effect can be used to determine the axis of an astigmatism with patients, as explained by Goersch [Goe93]: By presenting the patient an astigmatic fan, as shown in 5.3, one can identify the part of the fan that is the least blurred, which has the same rotation as the principal meridian of the astigmatic lens, that is best focused.

As stated by Denniston and Murray [DM14], astigmatism can be measured by the majority of people, but in most cases the imperfections in the eye are small enough such that no treatment is required. About 20% of the population have an astigmatism of more than 1dpt.

2.2.2 Strabismus

In this section, we describe strabismus as taught by Denniston and Murray [DM14, ch 17] and use this as a basis for our implementation later on. Before we can describe the strabismus itself, we first need to give a bit of context regarding binocular vision and eye movement. The human eye is moved by a set of muscles, sling structures and pulleys. One can distinguish between monocular and binocular eye movements. Binocular eye movements can further be distinguished whether they are movements in the same direction (versions) or in opposite directions (vergences). Vergences are limited to convergence and divergence, meaning inward and outward movements of the eyes. These are linked to accommodataion, resulting in convergence when near objects are focused.

With a healthy set of eyes, one can perceive the two separate images of the eyes as one fused image. For this to work, the muscles controlling the eye position need to adjust and maintain correct directions to enable fusion of images.

Strabismus describes problems with binocular vision. There are many reasons for strabismus to occur, like problems with the muscles that should control the eye or one-sided blurred vision, preventing correct fusion.

One can distinguish between manifest (tropias) and latent (phorias) deviations from the correct eye positions. Whilst tropias are manifest, they may or may not occur only in conjunction with accommodataion. Phorias can become manifest if fusion is suppressed and the remaining eye begins to focus.

In most cases, heterophoria does not cause any problems for the patients and is detectable in about 80% of the population [WP17] Heterophoria can however cause problems in some cases, like headaches or nausea. Since we assume it is hard to correctly simulate such effects, especially due to the complicated causes for heterophoria, we will focus the implementation and hence the further discussion in this chapter on heterotropia.

In the following, we describe heterotropia as taught by Walter and Plange [WP17], since they focus more on this type of strabismus as Denniston and Murray [DM14]. Heterotropia describes a manifest misalignment of the eyes. This includes concomitant strabismus, in which cases the severity of the misalignment is not related to the direction of the eye and incomitant strabismus, in which case the severity of the misalignment relates to the direction of the fixation. Incomitant strabismus can also include a completely immobile eye, due to nerve or muscle damage.

Heterotropia can cause problems when it newly develops. Although these symptoms may fade over time, it is important to treat them to reduce the risk of amblyopia [DM14, p 842]. The immediate effects caused by heterotropia are mostly confusion and diplopia. Confusion is the effect that two images appear to be layered on top of each other because different objects stimulate the same regions of the retina. Diplopia is the generation of double vision caused by the same object being projected onto different areas of the retina. When not taken care of, heterotropia can lead to amblyopia, causing the brain to suppress one eye.

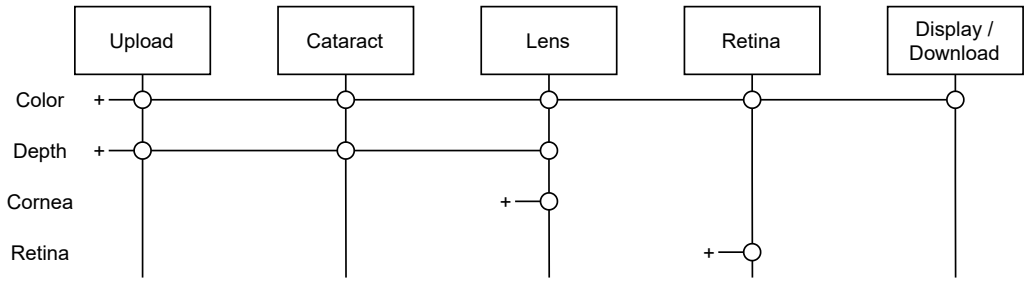


Figure 2.3: The order of nodes and the connections of their inputs and outputs. A '+' glyph indicates creation of a resource in this node.

2.3 Visual System Simulator

The VSS was developed by Schulz et al. [SRA+19] to provide a framework to facilitate fast prototyping of experiences with simulated visual deficiencies. Since we base our work on their simulator, the inner workings of the VSS are described in this section.

2.3.1 Simulator Stages

The simulator processes images in several stages to produce the desired output image. These stages are organized into nodes, that bundle a shader as well as logic for preparing user input and settings. The nodes further set up the input and output textures for the shader, so the nodes can be chained together.

A chain of nodes forms a flow. Since the simulator can simulate more than one eye, there can exist multiple flows at the same time. Each node, with the exception of the first and last node, uses the output resource of the previous node as its own input resource.

The first node is used to provide the flow with an initial input image. This input image can be obtained from a static image file or a video file. The input node further might adjust the projection of the image and extract depth information if provided. If required, the input node has to adapt the color space to the RGB space used throughout the simulation.

The nodes are executed roughly in the order, a incoming light ray would pass their physical counterparts in a real eye. This facilitates easy adoption of aspects of vision currently not simulated, like the use of glasses. The default nodes of a flow, their order and the forwarding of their resources are pictured in figure 2.3.

In contrast to the real eye, the cataract node is the first one to be executed, due to technical reasons described below. After this, the lens shader is executed, simulating the cornea, the lens and the vitreous humor. Its output is then passed to the retina shader, simulating the rods and cones located in the retina. The last step is a display shader that shows the result in a window. Instead of doing so, the result can be rendered to file using a "Download" node. A specialized display shader can further be used to display the resulting image in head-mounted displays.

In the following, we present the individual nodes in more detail, since in later parts of this work, we report on how their functionality has been extended.

2.3.2 Cataract Node

The cataract node simulates scattering of incoming light rays due to impurities in the lens, a reduction of contrast as well as increased sensitivity to bright light sources. The scattering effect is achieved by using a Gaussian blur with configurable kernel size. The reduction of contrast is realized by reducing the difference between the strongest RGB value and the other two values. To simulate increased sensitivity to bright light sources, the cataract shader uses a bloom effect by multiplying the color channels with a value based on their combined, perceived brightness. Although the effects of a cataract are caused in the lens of a real eye, they are simulated as the first node in the simulator. This is due to the incompatibility of this comparably easy implementation with the ray-based approach used in the lens shader.

2.3.3 Lens Node

The lens node simulates refractive errors in the eye. This is achieved by tracing rays through a simulated eye, refracting them at the four surfaces between the following media: the vitreous humor, the lens, the anterior chamber, the cornea and air. Note that the lens simulation does not take into account further factors regarding refraction like the tear film.

The above-mentioned surfaces are simulated in that order, from the inner-most surface to the outer. This is due to the way the simulation traces the ray of light: It starts from a location on the surface of the retina, that corresponds to the image coordinate and is directed through the above-mentioned surfaces onto an image plane, perpendicular to the view direction, that contains the input image. Although this is the opposite direction as light enters a real eye, this approach has some benefits compared to modeling the rays in their natural direction. The main one is that each value in the output buffer can be directly computed in a single shader pass and there will be no holes in the output image due to missed pixels.

When simulating one ray that originates a single photoreceptor, the lens simulation can create distortions, like the stretching caused by an astigmatic lens, but it cannot create the effect of defocused vision. In a perfect lens system, each light ray originating from a point in the regarded scene, hitting the lens, is focused into one single point in the retina. If the lens is defocused, the rays of one single point hit the retina in an area that has a size depending on the degree of defocus. In a natural scene, these areas overlap. In the overlap, the photoreceptors detect a mixture of the information coming from different points of origination.

To model this behavior, the simulation sends rays from the retina towards the image plane: Each simulated photoreceptor is the start point of several rays. These rays hit the image plane in different locations if the corresponding part of the scene is out of focus. This results in several color samples per photoreceptor. The lens simulation calculates the average of these samples to use as its output information. The path a sample ray takes throughout the simulated lens is shown in figure 2.4.

Since the lens simulation is aware of the distance of each pixel in the image plane thanks to the depth map, one can calculate the intersection point of the ray with the viewed scene while taking the distance of objects into account. Although not being perfect, this mechanism allows the VSS to simulate hyperopia and myopia.

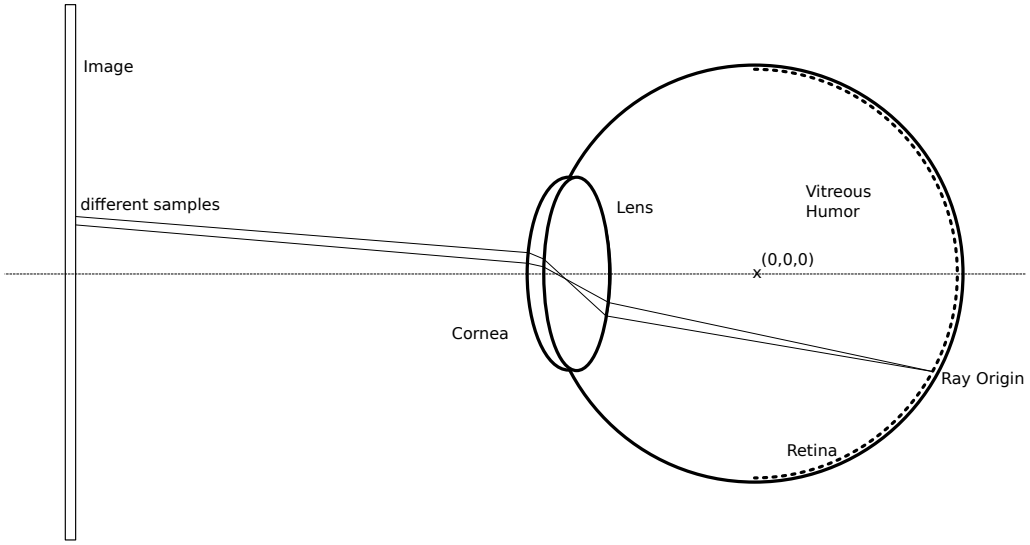


Figure 2.4: Path of a ray through the simulated lens. In this case, the rays to not meet each other in the same place on the image, hence the resulting image will be blurred. Note that the retina extends from $-90°$ to $+90°$ on the sphere.

The lens simulation has more than 25 parameters, ranging from the refractive index of a medium to the length of the vitreous humor, making it complex but flexible. One can adjust these parameters to adapt the lens simulation to special needs or to implement new eye diseases. One such disease, astigmatism, was implemented in this work, as described in 4.2.

2.3.4 Retina Node

The output of the lens node can be seen as the image reaching the retina. Based on this information the retina node simulates the retina of the eye. Each pixel corresponds to four cells: a rod and three cones. The retina node assumes an equal density of these cell types across all of the surface of the retina.

The retina node allows the user to supply a retina map, encoding whether each type of the four cells is present and whether their reception or signal output is dampened. With this, vision deficiencies like color-blindness and achromatopsia or glaucoma can be simulated.

Depending on the presence of cones, this node might introduce another blur operation: Although rods are capable to detect contrast in low-light situations, their resolution is lower than that of the cones. If this is the case, the simulator does not only blur the image but also creates a bloom effect to simulate the increased sensitivity to bright light.

2.3.5 The Configuration of the Visual System Simulator

The VSS can be configured via command-line parameters and a configuration file. The configuration file defines the simulated vision deficiencies. Whilst it is possible to simulate a single deficiency, the VSS is also capable of simulating multiple vision deficiencies and their interplay simultaneously. It

is worth noting here, that not all configurations are sane: An eye cannot have myopia and hyperopia at the same time. The configuration options are geared towards users that do not necessarily have an extended understanding of the simulator or the visual system and promote experimentation and exploration.

Whilst the configuration provides parameters to automatically generate retina maps and cornea maps, custom-made ones can be supplied too. Their creation requires a more in-depth understanding of the inner workings of the simulator, hence parameters, as used in 4.6, are the simpler option.

3 A Perception-Based Error Metric

In this chapter we describe the requirements for our error metric and outline why existing image quality measures have insufficient information to meet these requirements. We then describe our proposed error metric, which we then implement in chapter 4.1.

3.1 Image Quality Measures

Image quality measures are a widely used tool to assess the quality of image reproduction. Their origin lies in the evaluation of the quality of image or video transmission or compression [WR05], but they are also used as measures in different domains, e.g. in deep learning [DLHT14] or computer vision [RRKS19]. The goal of image or video compression is to create high-fidelity reproductions of an original image. If applied locally, image quality measures can be used in this context to assess problematic areas in an image. For the VSS, we need an error metric that is able to identify problematic areas and allows us to reason about the impact of vision deficiencies. But in contrast to image reproduction, we acknowledge there to be errors in the simulation, we deliberately introduce them in the vision deficiency nodes. This requires our metric to be able to make use of the knowledge it has about the image creation process so that one can later filter out changes in the image, that are not important for the evaluation. Although this requirement only partially aligns with the design goals of image quality metrics, we discuss the Mean Squared Error (MSE) and Structural Similarity (SSIM) as two basic image quality metrics here, so that we can later contrast them to our implementation.

3.1.1 MSE

The MSE is a simple but widely used metric to assess image quality [WR05, Ch. 5.4]. Its use in image and video coding quality analysis is different from the requirements for our use case: Image and video coding aim to create high-fidelity representations of input data in its output data with lower bit rates. In contrast, when developing the error metric for the visual system simulator, we acknowledge the existence of changes in the images. This allows us to cope with artifacts like changes in position of image features. A change in the position of a feature might not lead to a loss of information. This can be achieved by weighting down or completely ignoring the contribution of translation to the total error in our model. Since such a transformation would oppose the goal of high-fidelity reproduction, the behavior of the MSE is more suitable than our metric for that use case.

Even for image and video coding, the MSE has its shortcomings: Neither does it take the nonlinearity of perceived luminance exhibited by the HVS into account, nor is there a mechanism to model later components in the HVS like its ability to detect patterns [Dal92; WB09; WBSS04; WR05].

3.1.2 VDP and SSIM

Daly [Dal92] criticized the usage of image quality metrics solely based on linear differences in resulting pixel values early on. He proposed the Visible Differences Predictor (VDP) as a perception-oriented model to describe the impacts of differences in images. His model consists of several steps closely modeled after assumptions about the HVS. These steps include, amongst others, operations like adapting for luminance nonlinearity and a contrast sensitivity function. There are many assumptions made, based on real-world experiments, that might need to be calibrated for each individual [Dal92]. These encompass parameters like thresholds of noticeable differences.

Further, VDP makes assumptions about the display medium, which might need to be adapted for newer technologies [MMS].

Wang et al. [WBSS04] have developed an image similarity measure, SSIM, as an alternative to VDP. Instead of closely mapping their understanding of the HVS to a similarity measure, the authors chose a more simple approach, greatly reducing the number of parameters that need to be calibrated. Since they question whether the assumptions of the more complex approaches can be generalized, they developed a similarity measure with next-to-none tuneable parameters. SSIM can be computed pixel-wise as the product of three sum metrics: comparisons of luminance, contrast and structure. Although SSIM can be used to compute a similarity index across the whole image, it also can serve as a local similarity measure, if applied in patches.

VDP and SSIM only compare two images, to compute a measure of differences between them. In this work, we know the procedure that creates the differences and therefore can incorporate that information in our error measure. One type of such information is the exact amount of displacement of pixels in the xy plane, that can be extracted in our case but is impossible to infer for VDP and SSIM if only presented two images.

Image quality metrics are not reflecting the processing in the eye: Whilst one can focus on any part of a playing video at any time, the relation between the cornea, lens and retina are mostly fixed. This means, that you cannot move your eye in a way, such that the fovea would focus on the edges of the lens. Since the quality of human vision decreases with eccentricity, errors introduced by the edges of the lens will only be visible to parts of the retina with lower visual quality [Bar99]. By combining the error metric presented in the following with the simulation of retinal ganglion cells, following this model of reduced visual quality with increasing eccentricity, one could weight errors in the periphery differently than those in the center of the visual field.

3.1.3 The Simulation as a Measurement Process

To quantify changes introduced by vision deficiencies, we define an error metric in this section. While we exactly know the changes introduced by some components, like multiplications of color channels, there are operations that introduce uncertainty. We therefore design the error metric to be able to track the introduced uncertainty across all operations.

We regard our simulation as a process of measuring values. The simulated eye measures values on the input image and produces the output image from the measurement results. This approach allows us to use frameworks like the “Guide to the Expression of Uncertainty in Measurement”[Sta08]. Each pixel of the output image is generated by a measurement process, independent from the other

pixels. A perfect eye would produce the same value at an output pixel as observed in the input image at the same position. The simulated eye might not be perfect and therefore produce an output image that is different from the input image. We need to distinguish between error and uncertainty here. If the value is distorted by a known amount, e.g. because the simulated eye is unable to detect the color red, then the measurement introduces some error. If the value cannot be exactly measured, e.g. because multiple rays of light contribute to the output value, we use uncertainty to express the dispersion of values.

If compared with a real-world measurement instrument, our notion of uncertainty corresponds to the expected or observed measurement uncertainty (tightly related to the precision of the instrument), whilst the error corresponds to a known offset. A sampling process of one of the pixels is regarded as a series of repeated measurements of the same input value with possibly different results. We use the sample variance and standard deviation as measures for the observed uncertainty.

3.2 An Error Metric

We define an error e between the pixels of the input image and the output image.

Steps of the simulation may introduce errors in five dimensions: r, g, b, x, y. These represent the errors in the three color dimensions r, g, b and the error in image coordinates x and y.

For each step i of the simulation, a partial error e_i is computed. For the color dimensions, e_i is the difference of the output color c_{out} and the input color c_{in}. For x and y, the error equals the displacement $p_{out} - p_{in}$ applied in that step of the simulation. Following this scheme, e_i is a vector of component-wise differences.

These errors are calculated in each step for each pixel independently. That way, each pixel has an error value and these n partial errors e_i are added to form the cumulated error e of that pixel. The initial error e_0 is 0.

$$e = \sum_{i=0}^{n} e_i \tag{3.1}$$

$$e_i = \begin{pmatrix} c_{i,out} - c_{i,in} \\ p_{i,out} - p_{i,in} \end{pmatrix} = \begin{pmatrix} \Delta r_i \\ \Delta g_i \\ \Delta b_i \\ \Delta x_i \\ \Delta y_i \end{pmatrix}$$

3.2.1 Uncertainty Model

Some operations, like the lens simulation, might introduce uncertainty. This is caused by sampling different pixels to generate one output value. Before we lay out our method for tracking uncertainty across the simulation, we want to provide some intuition.

Intuition Some operations like the lens simulation achieve their functionality by simulating multiple rays of light. In a perfect lens, each ray originating from one spot on the retina going through the lens and hitting the image plane should result in the same position. If the target is defocused or the lens has a vision deficiency, these rays may hit different positions. By sampling the pixel values at the target locations, the simulation calculates the resulting color value.

One can interpret the sampling operation as multiple measurements of the same measure. Since the perfect lens will always measure the same pixel, all measurements will have the same color value and therefore the output value equals the input value of that pixel. If the individual rays target different positions, the color values may be different. The lens simulation averages the measured values to compute the output value. To get a measure of the dispersion of the sampled values, we use the standard deviation as a measure for uncertainty. The way the standard deviation is calculated in the simulation is described in the following sections.

It is worth noting here, that the lens simulation may produce a non-zero e, although there is no uncertainty at play. This is due to the possibility, that the lens simulation may not map input pixels to output pixels in the same location: If the simulated lens exhibits some sort of magnification whilst maintaining focus, the error vector e described above is non-zero, since positions and optionally color values change. In this case, all rays still meet in the same location. Further, e can be zero but uncertainty was introduced: If the eye looks at a uniformly colored plane that is out of focus and the lens has no further distortions, the output image equals the input image but the simulated rays do not meet in one point on the image plane.

Our definition Like the error, the uncertainty u has five dimensions: r, g, b, x, y. To simplify calculations, we use u^2 throughout the simulation and calculate u at the end.

$$u^2 = \begin{pmatrix} u_r^2 \\ u_g^2 \\ u_b^2 \\ u_x^2 \\ u_y^2 \end{pmatrix} \tag{3.2}$$

The uncertainty of a sampling operation has to be calculated for each component of u^2. To have a measure of the uncertainty of a sampling operation of n elements, we resort to the sample variance s^2 around the sample mean $\bar{x}$. We chose to base our uncertainty calculation on $\bar{x}$, since the calculations in the simulator always use the mean of the samples to form the output value.

Each of the sampled pixels in the current step might already contain some amount of uncertainty from previous steps. We summarize the input uncertainties as their mean m^2.

$$s^2 = \frac{1}{n}\sum_{k=1}^{n}(x_k - \bar{x})^2 \qquad m^2 = \frac{1}{n}\sum_{k=1}^{n}u_k^2 \qquad \bar{x} = \frac{1}{n}\sum_{k=1}^{n}x_k$$

Note that the u_k^2 in the equation above are the values from the previous step, saved alongside the pixel values x_k, each being the result of another sampling operation or the initial value. Subsection 3.2.2 details on the interpretation of this mean.

The m^2 contributes to the total u^2 of the new pixel alongside the sample variance s^2 as defined by equation 3.3.

$$u^2 = s^2 + m^2 \tag{3.3}$$

Covariance The components of u^2 might not be independent and therefore might exhibit some covariance. This becomes clear especially when thinking about grayscale images: In such cases, all the color channels share the same value.

We therefore need to calculate the variances and covariances of the sample. If we assumed covariances between all five components, the matrix in equation 3.4 would capture the full covariance matrix.

$$s^2 = \Sigma_s = \left[\begin{array}{ccc|cc} u_r^2 & u_{rg} & u_{rb} & u_{rx} & u_{ry} \\ u_{gr} & u_g^2 & u_{gb} & u_{gx} & u_{gy} \\ u_{br} & u_{bg} & u_b^2 & u_{bx} & u_{by} \\ \hline u_{xr} & u_{xg} & u_{xb} & u_x^2 & u_{xy} \\ u_{yr} & u_{yg} & u_{yb} & u_{xy} & u_y^2 \end{array}\right] \tag{3.4}$$

If we assume r, g, b to be independent from x, y, we can split the above covariance matrix in two separate matrices Σ_{rgb} and Σ_{xy}. Although there might exist input images with covariance between color and position (e.g. an image with some desaturation gradient), there are no calculations in the simulation that would make use of the covariances between Σ_{rgb} and Σ_{xy}. This is due to the fact that no implemented operation is modifying color and position with the one depending on the other. Therefore the covariances in the first and third quadrant in 3.4 will never be used and the values in the second quadrant won't affect those in the fourth quadrant and vice versa.

We define the used covariance matrices Σ_{rgb} and Σ_{xy} as shown in 3.5.

$$\Sigma_{rgb} = \begin{pmatrix} u_r^2 & u_{rg} & u_{rb} \\ u_{gr} & u_g^2 & u_{gb} \\ u_{br} & u_{bg} & u_b^2 \end{pmatrix} \qquad \Sigma_{xy} = \begin{pmatrix} u_x^2 & u_{xy} \\ u_{xy} & u_y^2 \end{pmatrix} \tag{3.5}$$

3.2.2 Interpretation of u_k^2

Before we explain how the propagation of uncertainty is implemented, we present a more in-depth look on how the sampled uncertainty u_k^2 can be interpreted. To get an understanding of the meaning of u_k^2, we construct it for a few steps of the simulation, beginning with the initial value.

Initial value The initial uncertainty of all pixels is 0. We thereby assume that the input image is free from uncertainties created in its capture: $u_0^2 = 0$.

First step creating uncertainty In the first step, u_1^2 is simply the sample variance of the current operation.

$$u_1^2 = s_1^2 + m_1^2 \qquad m_1^2 = \frac{1}{n}\sum_{k=1}^{n} u_{k,0}^2 = 0 \qquad u_1^2 = s_1^2$$

Second step In the second step u_2^2 consists of its own sample variance, as well as the average sample variance of all elements from the first step.

$$u_2^2 = s_2^2 + m_2^2 \qquad m_2^2 = \frac{1}{n}\sum_{k=1}^{n} u_{k,1}^2 = \frac{1}{n}\sum_{k=1}^{n} s_{k,1}^2 \qquad u_2^2 = s_2^2 + \frac{1}{n}\sum_{k=1}^{n} s_{k,1}^2$$

Subsequent steps As shown here with a third step, the uncertainty of a step is added to the average sample variances of all previous steps.

For the third step, we get the the following:

$$u_3^2 = s_3^2 + m_3^2 \qquad m_3^2 = \frac{1}{n}\sum_{k=1}^{n} u_{k,2}^2 = \frac{1}{n}\sum_{k=1}^{n}\left(s_{k,2}^2 + \frac{1}{n_k}\sum_{l=1}^{n_k} s_{l,1}^2\right)$$

Given that each sampling operation is done for all pixels, we can assume that all n_k are the same, say n'. This gives us

$$m_3^2 = \frac{1}{n}\sum_{k=1}^{n}\left(s_{k,2}^2 + \frac{1}{n'}\sum_{l=1}^{n'} s_{l,1}^2\right) \qquad m_3^2 = \frac{1}{n}\sum_{k=1}^{n} s_{k,2}^2 + \frac{1}{n}\sum_{k=1}^{n}\frac{1}{n'}\sum_{l=1}^{n'} s_{l,1}^2$$

$$m_3^2 = \frac{1}{n}\sum_{k=1}^{n} s_{k,2}^2 + \frac{1}{nn'}\sum_{k=1}^{n}\sum_{l=1}^{n'} s_{l,1}^2$$

$$u_3^2 = s_3^2 + \frac{1}{n}\sum_{k=1}^{n} s_{k,2}^2 + \frac{1}{nn'}\sum_{k=1}^{nn'} s_{k,1}^2$$

To summarize, the uncertainty u_n^2 at a step k can be computed as the sum of the averages of the sample variances of all previous steps, as well as the own.

3.2.3 Propagation of Uncertainty

When the values are altered in a simulation step, like being multiplied with a scalar, their uncertainties need to be altered too. This is done according to the propagation of uncertainty, we follow the recommendations of the GUM [Sta08, ch 5.2.2] here.

If a function f with N input variables x_i alters a pixel value, the corresponding uncertainty is propagated according to 3.6 [Sta08, ch 5.2.2].

$$u^2 = \sum_{i=1}^{N} \sum_{j=1}^{N} \frac{\partial f}{\partial x_i} \frac{\partial f}{\partial x_j} u_{x_i,x_j} \tag{3.6}$$

With $\frac{\partial f}{\partial x_i}$ being the partial derivative with respect for parameter x_i and u_{x_i,x_j} representing the input-variance of that parameter.

To ease calculation, we express the above formula in matrix notation according to [Bar93, eq. 4.19], see equation 3.7. We reuse the covariance matrix as defined in 3.5 for this. The calculation in equation 3.6 equals the multiplication with the Jacobi matrix J_f of the function f and its transpose.

$$\Sigma_f = J_f \Sigma_x J_f^\intercal \tag{3.7}$$

Example To illustrate this calculation, we give a concrete example: The full calculation for the propagation of uncertainty used in the modification performed by the bloom operation, as found in the cataract simulation. In this operation, the color value of the processed pixel is multiplied by its perceived brightness, scaled with some constant factor. We denote the brightness function as l and the used constant as c. We have $f(x_{rgb}) = x_{rgb} \cdot (1 + l(x_{rgb}) \cdot c)$. Whilst x_{rgb} is a vector of three components, $l(x_{rgb})$ only returns a scalar. We compute l as $l(x_{rgb}) = w_r \cdot x_r + w_g \cdot x_g + w_b \cdot x_b$ with w_{rgb} as weights to account for the individual contributions of the r, g, b channels to the perceived brightness. To save space, we use $d = x_{rgb} \cdot w_{rgb}$ as the dot-product between the two vectors.

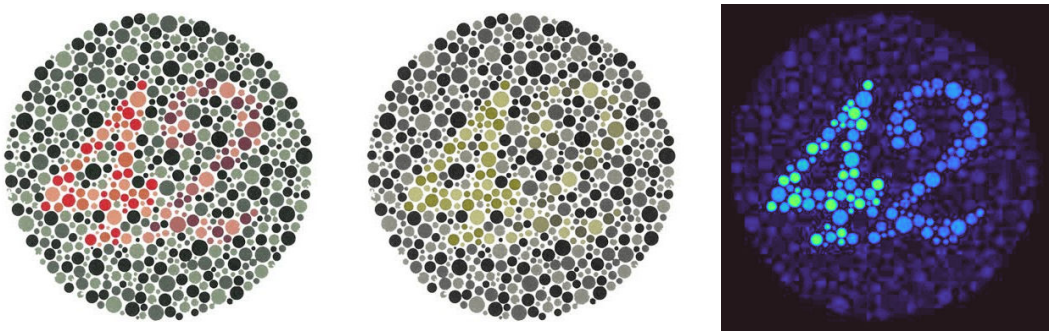
This gives us the Jacobi matrix J_f as follows.

$$J_f = \begin{pmatrix} c\,d + cw_r x_r + 1 & cw_g x_r & cw_b x_r \\ cw_r x_g & c\,d + cw_g x_g + 1 & cw_b x_g \\ cw_r x_b & cw_g x_b & c\,d + cw_b x_b + 1 \end{pmatrix}$$

We can then use J_f and the below Σ_{rgb} as described in equation 3.7.

$$\Sigma_{rgb} = \begin{pmatrix} u_r^2 & u_{rg} & u_{rb} \\ u_{gr} & u_{g^2} & u_{gb} \\ u_{br} & u_{bg} & u_b^2 \end{pmatrix}$$

We can use Equation 3.7 for all operations in the simulation that do not introduce additional uncertainty. In the operations that create new uncertainty, Equation 3.3 is used.



(a) Input image **(b)** Output image **(c)** Visualized $\|e_{r,g,b}\|$

Figure 3.1: Visualized $\|e_{r,g,b}\|$ of simulated red-blindness as a heatmap with the turbo color scheme

3.2.4 Combination function

We define a combination function $f(e, \Sigma$ that is able to process the vectors of e and u.

To allow the user to further process the resulting vector field, it makes sense to expose the error and uncertainty vectors to the user. They can then define their own f, according to the specific needs of the studied scenario. This can include filtering for specific properties like a threshold of $\|e_{x,y}\|$ or advanced weighting of the components.

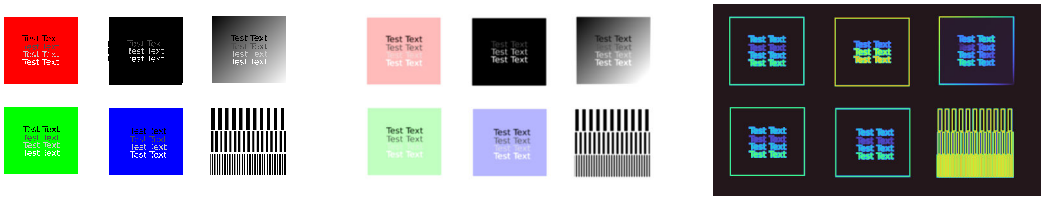
Nonetheless, to be able to provide a general overview of the severity of the simulated deficiency, we define six default combination functions, implemented as separate view modes in the VSS. A set of sample images and a detailed discussion of the results can be found in 5.4.1. In figure 5.9 we present sample renderings using the individual combination functions on a scene.

The Euclidian Length of the Color Error Vector The first combination function being $f(e, \Sigma) = \|e_{r,g,b}\|$, simply calculates the length of the RGB components of the error vector. We calculate this distance in the RGB space, other distance metrics like Δe_{2000} might be worth considering to model the human perception more closely. Figure 3.1 shows $\|e_{r,g,b}\|$ for a simulated red-blindness applied to a test scene. A more detailed discussion of this figure can be found in 5.4.1.

The Aggregated Standard Deviation of the Color Vector The second predefined combination function is $f(e, \Sigma) = \|u_{r,g,b}\|$. This is the euclidian length of the standard deviation vector in the color dimensions. This is easy to compute, since Σ_{rgb} already has the variances in the diagonal and we want to have the length of the vector with the standard deviations as components. We threfore can comupte this combination function as $\|u_{r,g,b}\| = \sqrt{\text{trace}(\Sigma_{r,g,b})}$.

The resulting value has the same unit as the color values themselves. It can be interpreted as the square root of the total sample variance of the color values [Ren12, eq. 3.78].

Figure 3.2 shows a visualization of $\|u_{r,g,b}\|$ of the cataract simulation. While 3.2a shows the original test image, 3.2b shows the simulator output and 3.2c shows $\|u_{r,g,b}\|$ as a heatmap with the turbo color scheme. Figure 5.9f shows $\|u_{r,g,b}\|$ for the classroom scene.



(a) Input image **(b)** Output image **(c)** Visualized $\|u_{r,g,b}\|$

Figure 3.2: Visualized $\|u_{r,g,b}\|$ of a simulated cataract as a heatmap with the turbo color scheme

The Euclidian Length of the XY-Error Vector We define a third combination function $f(e, \Sigma) = \|e_{x,y}\|$. This combination function shows the euclidian length of the displacement introduced to a pixel.

If used with the VSS, this combination function only produces interesting results if used with the lens simulation and some depth information. For all other implemented diseases, the XY error is constant across the whole output image. If the simulator is extended to include new diseases or existing diseases are implemented in more detail, this metric might become more relevant for images without depth information. A candidate for such a disease with interesting displacement information without manipulation by the lens simulation is the macular degeneration that can in some cases cause ripple effects.

Compared with the length of the color change vector, the length of the positional change vector is small: Whilst $e_{r,g,b}$ might span a large portion of the RGB space, e.g. if a color deficiency removes one color, a change in position of a few dozen pixels may already be perceived as a strong distortion. If the change in position is expressed in resolution-independent screen coordinates, this value is so small in some scenes, that features become visible only after a multiplication by more than 20. To cope with this problem, we decided to normalize the vector components relative to a maximal component length $\hat{e}_{x,y}$. We chose $\hat{e}_{x,y}$ to be the maximum observed displacement in our test scenes, in this case $\hat{e}^{-1}_{x,y} = 5.4$, what roughly equals 200 pixels in an image with a resolution of 1080 pixels. This value has to be adjusted for the observed scene if the values are outside of an acceptable range.

In figure 5.9d we present the visualization of $e_{x,y}$ for the classroom scene. A detailed discussion of the results can be found in section 5.4.1.

The Aggregated Standard Deviation of the XY Vector We define another combination function $f(e, \Sigma) = \|u_{x,y}\|$. This is the euclidian length of the standard deviation vector in the position dimensions. $\|u_{x,y}\| = \sqrt{\text{trace}(\Sigma_{x,z})}$.

As with $\|e_{x,y}\|$, $\|u_{x,y}\|$ only produces interesting values with the currently available nodes, if depth information is provided. One can observe the same scaling problems in $\|u_{x,y}\|$ as with $\|e_{x,y}\|$. We therefore use a $\hat{u}^2_{x,y}$ to scale the values to be in a useable range. For our test scenes, $\hat{u}^{-2}_{x,y} = 43.2$ produced useable results.

$\|u_{x,y}\|$ corresponds to the aggregated standard deviation. Since this metric does not have a direction, $\|u_{x,y}\|$ describes a circle with a radius of the produced value. Although this circle might be a measure for the blurriness of an image region, it does not necessarily has to correctly describe the loss of information. If an image of a perfectly white surface is simulated with a vision deficiency

that causes the surface to be defocused, we face the following problem: Whilst there might be depth information included in the scene, one cannot see any blur, since the scene itself consists of uniform color. As we will see in the results chapter later, $\|u_{x,y}\|$ is an interesting measure, although not being sufficient to assess the loss of information through blur. Figure 5.9g shows $\|u_{x,y}\|$ for the classroom scene.

The Euclidian Length of the RGBXY-Error Vector Instead of using only parts of e as in the previous sections, we define this combination function as $f(e, \Sigma) = \|e_{r,g,b,x,y}\|$. The error vector e exposes all information about absolute changes gathered throughout the simulation. Exposing e enables processing in external tools for more in-depth analysis, for example principal component analysis. Using $\|e_{r,g,b,x,y}\|$, we provide a combination function that gives a general overview of the total distortion, in the color and position dimensions. Figure 5.9e shows $\|e_{r,g,b,x,y}\|$ for the classroom scene.

The Aggregated Standard Deviation of the RGBXY Vector This last combination function $f(e, \Sigma) = \|u_{r,g,b,x,y}\|$ is the aggregated standard deviation of all tracked dimensions. Although we map from the available five dimensions to a single scalar value and thereby lose a lot of information, the length of the five-dimensional vector u allows for a general overview. Despite its simplicity, $\|u_{r,g,b,x,y}\|$ allows for interesting analysis, as discussed in a later chapter. Figure 5.9h shows $\|u_{r,g,b,x,y}\|$ for the classroom scene.

4 Extension of the Simulator

One goal of this work is to extend the functionality provided by the VSS. For this, we implement the previously described error metric, implement new vision deficiencies, extend the display capabilities and add a basic attention guidance system.

4.1 Implementation of the Uncertainty Visualization

Now that we have laid out the theoretical foundations of the error metric, we cover the implementation details in this section. As outlined in 2.3, the VSS is organized into independent flows that are comprised of nodes. These nodes pass shader resources in one direction.

To be able to track the changes done to images, we needed to extend this mechanism. The combination functions described above require covariance matrices for RGB and XY values, as well as vectors for absolute changes in color and position. Since covariance matrices are symmetric, one only needs to save half of the values outside the diagonal. Instead of using one large covariance matrix as shown in 3.4, we use two smaller, independent covariance matrices as shown in 3.5. This allows us to decrease the required space to pass the values from shader to shader from 15 to 9. Since the used graphics framework 'gfx'[1] has only implemented a small number of texture buffer types, we chose to split the matrix values into several vectors. As shown in 4.1, we introduced the following resources:

- The change of the color channels
- The variance of the color channels. This equals the diagonal of Σ_{rgb}.
- The covariance of the color channels. These are the off-diagonal values of Σ_{rgb} and Σ_{xy}. We merged these values in one texture since gfx supports at most four values in a texture buffer and the covariances of the color channels only take up three values.
- The change and variance of the position. The only reason these are grouped together is that they need two values each and we can save on textures if we place them in a texture buffer with four values.

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

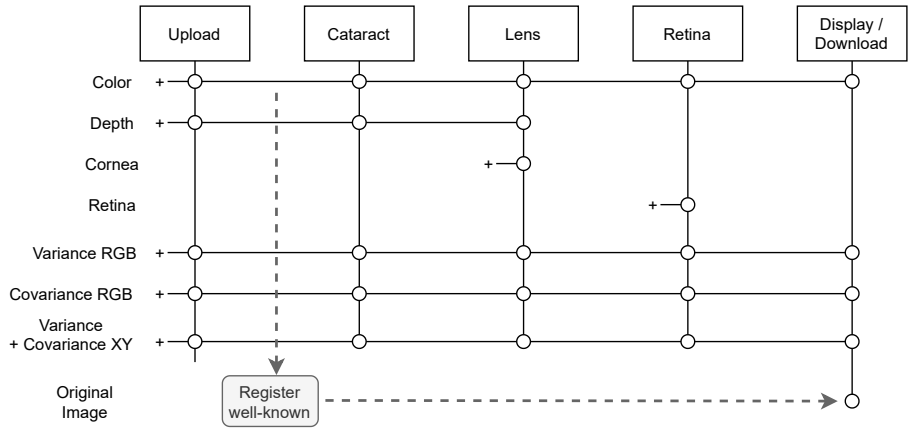


Figure 4.1: Flow of nodes with additional resources

4.1.1 Implementation of Error and Uncertainty Tracking

To implement tracking of changes and uncertainty throughout the simulator, there are several distinctions to be made. We use the shader resources shown in 4.1 to pass error and uncertainty values from node to node in the flow execution. To increase performance of the simulator if neither error nor uncertainty needs to be tracked, we only enable tracking if the according view modes are selected.

Tracking of error Since we regard the simulation as a measurement process, each change introduced in the simulation can be seen as an error in the measurement. To get the total changes, we track the individual changes in the simulation steps and add them to a buffer. That way, the changes accumulate with the shader execution and can be passed from node to node. Within the shader of a single node, we add the difference between the value before a step and the value after the step to an accumulator that is written to the texture buffer in a last step to make it available for the next node. This procedure is also shown in equation 3.1.

Tracking of uncertainty We distinguish between operations that introduce new uncertainty by sampling more than one value and operations that do not introduce new uncertainty but change the value and hence require us to propagate uncertainty.

In the first case, we apply equation 3.3 to build a new uncertainty value from past values included in the sampled pixels and the newly created dispersion of values in the current step. Since the shaders only write one pixel, we cannot support multiple sampling operations in the same shader without writing the data to a texture. Due to the loops required to determine the variance of the current value and to calculate the contribution of past values, these operations are comparably expensive.

If an operation changes the color or position values of a pixel, we need to propagate the contained uncertainty. This is done according to 3.7. Since these operations are mostly only a few matrix multiplications they are not gated by an if-condition checking for the view modes to keep the source code cleaner.

4.2 Astigmatism

The initial version of the VSS assumed the refractive surfaces of the simulated optical system to be spherical caps. Real eyes might exhibit different curvatures along several principal meridians. As described in 2.2.1, this may lead to astigmatism.

To allow for such lenses to be simulated with the VSS, we extended the simulator as described in this section. Astigmatism can be caused by unequal refraction in the cornea, the lens, as well as by the vitreous humor. Since measuring these astigmatisms individually requires special skills and equipment, optometrists resort to state the astigmatism of an eye as a whole: The spherical equivalent to correct the difference between the two foci and the axis of the principal meridian with the strongest refraction. We therefore only simulate astigmatism in one surface. Since it is the last surface in the calculation of ray refraction in the simulation and its manipulation does therefore not influence the previous steps, the outer cornea was chosen as the source of the astigmatism.

Although the simulator already is able to simulate imperfections of the cornea with the use of a cornea map, we decided to implement the astigmatism in a less cumbersome way. Instead of simulating the outer cornea surface as a part of a sphere, we simulate it as a part of an ellipsoid. In the case of a healthy cornea, all axis of the ellipsoid are equal and hence the simulation will still use it as a sphere. Instead of being required to manually calculate deflection vectors for each part of the cornea as required by the cornea map, the user of the simulator can now specify the eccentricity of the ellipsoid as well as the angle of the main axis. This corresponds to the way of specifying astigmatism by optometrists.

4.2.1 Location of the Foci

In our simulated optical system, we assume that at least one of the foci of the astigmatic system is correctly placed on the retina. This assumption is made because of two reasons:

- If we would not place this constraint, spherical errors could be created by the user in several places in the simulator: The configuration of the astigmatism and the configuration of the Myopia and Hyperopia parameters. This redundancy would make it hard to reason about the resulting simulation.
- Further, this assumption greatly simplifies the calculation and specification of the ellipsoid representing the cornea.

We assume this assumption to be correct, since one can always adjust the system with a spherical lens to force one of the foci onto the retina. This assumption is in line with the procedure used by optometrists to correct astigmatism: One of the first steps is to place one of the foci on the retina to measure the required spherical correction [Goe93].

4.2.2 Ellipsoid for the Astigmatic Cornea

An ellipsoid of three dimensions can be defined by the position of its center in addition to three vectors representing its three main axes. We use a, b and c as the main axis here. We then can define the ellipsoid by equation 4.1.

$$\frac{x^2}{a^2} + \frac{y^2}{b^2} + \frac{z^2}{c^2} = 1 \tag{4.1}$$

In the simulation, we model the ellipsoid in such a way, that the main axis b and c are the same and equal to the original cornea radius used in the spherical simulation. This assures that one focus point is located on the retina and leaves a as a variable parameter. Further, rotation along the a axis are irrelevant, since b and c are equal. This leaves rotation along b and c as options. We can fix c in such a way that it is parallel to the z axis, since b and c can be rotated freely along the a axis. With such a position, a rotation alongside the c axis, hence along the z axis, equals what an optometrist would denote the angle of the astigmatism: The angle of the principal meridian.

The only other possible rotation is along the b axis. Such a rotation would create a protruding part of the cornea, which can be observed in diseases like keratoglobus [WD13]. Since this is not to be simulated in the case of an astigmatism, we prohibit rotation along the axis. To conclude, the only important axis for the rotation of the ellipsoid is the z axis.

This leaves only two parameters to be configured by the user: the eccentricity of the ellipsoid in direction of the a axis and the rotation along the z axis.

The vectors a, b and c can be written as scalars if the ellipsoid is rotated so that the directions of the main axis coincide with the coordinate axis. This rotation be denoted by a matrix R^{-1}. By scaling the ellipsoid in each direction of its axis, we can create a sphere with a radius of $r = 1$. The scaling can be described by a matrix S^{-1}. A third matrix T^{-1} can be used to move the ellipsoid to the origin.

If the transformations mentioned above are applied to the vectors describing the simulated ray of light, we can calculate its intersection point as the intersection point with a sphere. This transformation is computed as described in 4.2, with p_{ray} denoting the start of the ray vector at the last refraction surface and d_{ray} denoting its direction.

$$\begin{aligned}
p_{\text{scaled}} &= S^{-1} \cdot T^{-1} \cdot R^{-1} \cdot p_{\text{ray}} \\
d_{\text{scaled}} &= S^{-1} \cdot R^{-1} \cdot d_{\text{ray}} \\
p'_{\text{target}} &= \texttt{intersect}(p_{\text{scaled}}, d_{\text{scaled}}, 0, 1) \\
d_{\text{target}} &= R \cdot \texttt{normal}(S \cdot p'_{\text{target}}, a, b, c) \\
p_{\text{target}} &= R \cdot T \cdot S \cdot p_{\text{target}}
\end{aligned} \tag{4.2}$$

Here d_{target} is the resulting direction of the normal of the ellipsoid surface at the location of the intersection point with the ray. The position of that intersection point is denoted by p_{target}.

4.2.3 Changes to the Simulator

The calculation of the path a ray takes in the simulated optical system has to be adopted to use the ellipsoids described above. The only surface that we assume to be affected by the astigmatism is the outer cornea. Hence, the intersection point and refraction have to be calculated only for the last surface in the refraction process.

To configure the shape of the cornea, the user has to supply two parameters: The angle of the astigmatism, corresponding to the rotation of the ellipsoid. Optometrists calculate the angle starting from the vertical up direction counterclockwise [Goe93], so we also adopted this notation for the parameter.

The other configurable parameter is the eccentricity of the ellipsoid. This parameter is commonly given in dpt, as taught by Goersch [Goe93], denoting the difference of the optical powers of the two principal meridians. To get the corresponding eccentricity, we use the following calculations.

To get the refractive power D_1 of a thin lens with the outer radius r_a and inner radius r_b, we use the following equation.

$$D_1 = \frac{n' - n}{n} \cdot (\frac{1}{r_a} - \frac{1}{r_b}) \tag{4.3}$$

The difference between two lenses of a material with refractive index n' in a medium with refractive index n can be calculated according to 4.4.

$$\Delta D = D_1 - D_2 = \frac{n' - n}{n} \cdot ((\frac{1}{r_{a_1}} - \frac{1}{r_{b_1}}) - (\frac{1}{r_{a_2}} - \frac{1}{r_{b_2}})) \tag{4.4}$$

Both meridians share the same inner cornea surface radius, hence $r_{b_1} = r_{b_2}$. We further define d as the difference in the principal meridians, $d = r_{a_2} - r_{a_1}$. If we assume the eye to be facing air, we have that $n = 1$.

$$\Delta D = \frac{(n' - 1) \cdot d}{r_{a_1} \cdot r_{a_2}} \tag{4.5}$$

Equation 4.5 is heavily dependent on the exact refractive index and cornea radius of the individual eye. To be able to transfer results between different eyes, we assume there is some serious effort required for calibrating these parameters. We therefore resort to the much simpler approximation provided by Goersch [Goe93] with d in mm and D in dpt:

$$\Delta D = 5 \cdot d \tag{4.6}$$

4.3 Retinal Ganglion Cells

The VSS has to be extended to include capabilities to simulate and visualize activation of RGC. In their work, Aleman et al. [AWS18] implemented their RGC simulation by defining center-surround receptive fields across the retina. To analyze the count of activated ON-Cells and Off-Cells, they

varied the size of the simulated receptive fields and plotted the results. To visualize the activation of RGC, they colorized the simulated retinal ganglion cells according to their type (red/blue for ON/OFF). They used a constant size for their receptive fields for this visualization. In their case, examining the effect of contrast polarity when reading text, this model might be enough: The gaze of the reader will follow the text flow across the whole screen.

They implemented the visualization by deciding for each simulated RGC, whether the stimulus would activate an ON or OFF cell by comparing the value of the center of the receptive field with the average of the surrounding pixels in a defined distance.

Our requirements contrast the assumption of Aleman et al. [AWS18]: We simulate the retina at any given time with the gaze of the user. This means, that we cannot simply assume a constant size of RGC across the whole image. As hinted in 2.2.1, the distribution and size of RGC is far from trivial with variable eccentricity. There have been many decades of research concerned with the position, size and density of RGC. We therefore summarize some of the major challenging factors in implementing a simulation of the cells.

While there have been earlier attempts to describe the position of RGC, Dacey [Dac93] found that the coverage of photosensitive cells with RGC is roughly 1 across the whole retina. This greatly simplifies definition of formulae, since one can easily convert densities to sizes with a fixed coverage of 1 but assumes no holes in coverage and next to no overlap. Kolb and Dekorver [KD91] on the other hand found that their observed midget ganglion cells are connected to two bipolar cells. Balasubramanian and Sterling [BS09] state that some RGC have three-fold overlap in their dendritic trees to achieve a better signal to noise ratio in these areas. This conflicts with the findings of Dacey [Dac93] in at least some areas of the retina.

Drasdo et al. [DMKC07] studied the displacement of RGC from their receptive fields. In the fovea, the ratio of RGC to cones is nearly one. This does not conflict with the observations of Kolb and Dekorver [KD91], since the ratio of ganglions to cones does not rule out connectivity between one ganglion cell and multiple cones. The cells themselves are displaced towards the parafovea to avoid obstructing incoming light, leaving nearly no RGC in the fovea itself. This displacement necessitates a distinction between the position of the RGC itself and the position of the connected receptive field.

Wienbar and Schwartz [WS18] summarize findings of over 50 years, hence we resort to their work to summarize the challenges concerned with the shape of RGC. Many attempts to define a formula for the stimulation of RGC assume a model of receptive fields as a two-dimensional Difference of Gaussians to predict activation of the ganglion cell. These assumptions reject uniform influence of photosensitive cells in the receptive fields and instead model their influence as two Gaussians with inverted polarity, scaled according to the presumed size of the receptive field. Wienbar and Schwartz [WS18] point out that this model has several weaknesses, one of which is that retinal RGC are not necessarily round.

Whilst Drasdo et al. [DMKC07] assume a circular shape of RGC and use a hexagonal packing to express their coverage, Bloomfield [Blo94] already found that there exist RGC that exhibit non-circular shape and are sensitive to orientation. They could not identify a clear correspondence between the structure of the cell and its sensitivity to orientation.

Venkataramani and Taylor [VT16] observed that RGC sensitive to orientation also have a center-surround structure, like their orientation-less counterparts, despite being not circle-shaped.

Wienbar and Schwartz [WS18] provide an overview of research that is concerned with the activation of RGC over time. This includes decreasing activation output over time when presented with a constant stimulus, as well as a contribution of orientation-aware cells to the detection of motion.

4.3.1 Distribution of Retinal Ganglion Cells

This section is mainly concerned with the density of retinal ganglion cells. We use $\frac{RGCf}{deg^2}$ as a measure for the density of the receptive fields of RGC. Here, a degree means a degree of angle on the retina, not the visual field. To be exact, one needs to distinguish between angles from the optical center and the visual center on the retina, but since the simulator has no notion of such a difference, we ignore this distinction for the implementation. This means, that we measure degrees from the optical center in the simulated retina.

Since there has been a lot of work done attempting to map the density, size and position of RGC, as well as the distribution between ON-Cells and OFF-Cells and P-Cells and M-Cells, we refer to the summaries of Watson [Wat14]. The shape of RGC is most often approximated to be circular or hexagonal, whilst acknowledging the existence of the exceptions described above. The distribution of RGC varies differently along the four main meridians. Although there are reports of asymmetries between ON-Cells and OFF-Cells, Watson [Wat14] neglect these findings and simplify their model by assuming an even distribution between the two types. When talking about eccentricity, it is worth noting that there exists a disparity between the optical axes and the axis of the visual field. Since the VSS does not distinguish between the two, we ignore the distinction in our simulation and refer to coordinates based on the visual axes, as if they were identical with those of the optical axes. Based on the observations of Dacey [Dac93] and Drasdo et al. [DMKC07], Watson [Wat14] computed several formulae to model density and spacing of midget retinal ganglion cells outside of displacement zone. Within the displacement zone, RGC density cannot be used as a measure of receptive field density, since the cells are displaced from their photoreceptors and hence their receptive fields. The authors derive the Retinal Ganglion Cell receptive Fields (RGCf) density from the cone density within the displacement zone, by assuming one cone to be connected to two Midget Retinal Ganglion Cell (mRGC), as observed by Kolb and Dekorver [KD91].

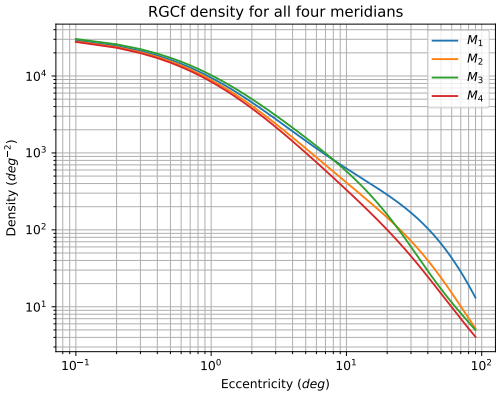
Whilst being the most numerous cells, the midget retinal ganglion cells are not the only ones. While the authors use the ratio of mRGC to cones to find the density of mRGC in the fovea, no such easy relation exists for RGC in general. The authors use factors found by Dacey [Dac93] and Drasdo et al. [DMKC07] to compute the density of ganglion cells in the fovea. These factors vary depending on the study and 1.12 is chosen by the authors.

4.3.2 Adaption of the RGC Placement Formula

In this section, we describe the formulae of Watson [Wat14] and explain the modifications we made to adapt them for our use case. In the fovea, the density of mRGC follows the density of the cones, if one assumes that one cone is connected to two mRGC. This is expressed by $d_{mf}(0) = 2d_c(0)$. With a midget cell fraction of $f(r)$ at eccentricity r, Watson et al use $d_{gf}(0) = f^{-1}(0)d_{mf}(0)$ as the equation for the density of all ganglion cells. We do not distinguish between the cell types and assume a constant $f(r) = f(0)$. A more elaborate distinction would be justified if we would simulate the individual cell types differently.

Meridian	k	a	r_2	r_e
Temporal	1	0.9851	1.058	22.14
Superior	2	0.9935	1.035	16.35
Nasal	3	0.9729	1.084	7.633
Inferior	4	0.996	0.9932	12.13

(a) Parameters for calculation of $d_{gf}(r,k)$ according to Watson [Wat14]



(b) RGCf density as a function of eccentricity, using the values from 4.2a

Figure 4.2: Receptive field density parameters and plotted function

For $f^{-1}(0)$, we follow the usage of 1.12 by Watson [Wat14]. With a peak cone density of 14804.6 deg^{-2} they get $d_{gf}(0) = 33,163.2\ deg^{-2}$ as the peak density of receptive fields in the fovea.

They further define an equation to compute the density of receptive fields based on the eccentricity r and weighting factor a_k depending on the meridian k. The factors for reducing density $r_{2,k}$ and scaling the exponential $r_{e,k}$ are also dependent of the meridian k.

$$d_{gf}(r,k) = d_{gf}(0) \cdot \left[a_k \left(1 + \frac{r}{r_{2,k}}\right)^{-2} + (1 - a_k) exp\left(-\frac{r}{r_{e,k}}\right) \right] \tag{4.7}$$

They optimized the formula to fit reported data outside of the displacement zone. Within the displacement zone, this function greatly differs from observed locations of retinal ganglion cells. Since we are interested only in the location of the receptive fields, this displacement is irrelevant for us. We therefore use 4.7 without a function compensating cell displacement.

Watson [Wat14] further expand their function by introducing a scaling function $f(r)$ that represents the distribution of M-Cells and P-Cells. Since we do not distinguish between the two, we leave this part for future work.

Table 4.2a lists the parameters used for 4.7 by Watson [Wat14], onto which we base our model.

To calculate the spacing between RGC, Watson [Wat14] use the formula 4.8. Since we do not use a hexagonal grid, we can simply use the area of our rectangular grids to calculate spacing, as defined in 4.9.

$$s(r,k) = \sqrt{\frac{2}{\sqrt{3} d_{gf}(r,k)}} \tag{4.8}$$

$$s(r,k) = \sqrt{\frac{1}{d_{gf}(r,k)}} \tag{4.9}$$

Since 4.9 only defines spacing along the four meridians, one needs to be able to compute the spacing in an arbitrary point from the calculation from the meridians. The authors define $r_{x,y} = \sqrt{x^2 + y^2}$ with x and y as coordinates of the retina. They then approximate the spacing function by using 4.10, where m_1, m_2 are the indices for the enclosing meridians.

$$s(x, y) = \sqrt{\left(\frac{x}{r_{xy}} s(r_{xy}, m_1)\right)^2 + \left(\frac{y}{r_{xy}} s(r_{xy}, m_2)\right)^2} \tag{4.10}$$

Although the authors include calculations to extend spacing to arbitrary points in the binocular field of vision, we do not implement these. This is due to the limited accuracy, with which stereo vision is mapped to the simulated retina. This might be an interesting area for future work.

The approach outlined above does come with some limitations for our simulator. Given an image resolution of 1920×1080 pixels to work on, it is quite obvious that it would be hard to accurately simulate nearly 100 Million photoreceptors [CSKH90] per retina.

The same applies to the density of ganglion cell receptive fields: With a peak density of 33, $162.3 \frac{1}{deg^2}$, achieving realistic granularity would account for 16389 pixels in the direction of one meridian, if we only have access to constant-spaced pixels.

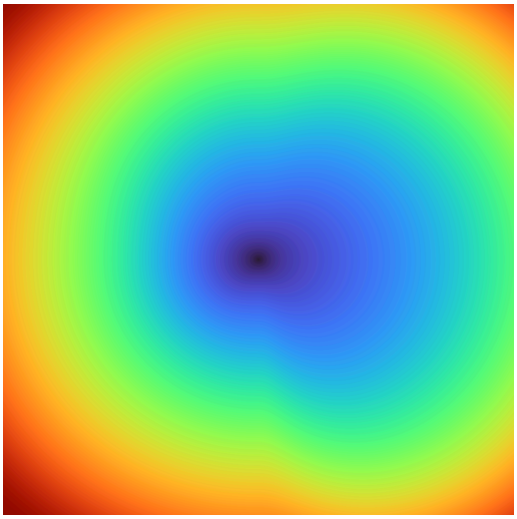
Whilst the model we described above can calculate the size of RGCf, it does not give us the number of cones in the receptive fields. Whilst this is limited in the real retina by the actual amount of cones present, we have a constant density of cones in the simulator.

4.3.3 The Grid of Simulated Retinal Ganglion Cells

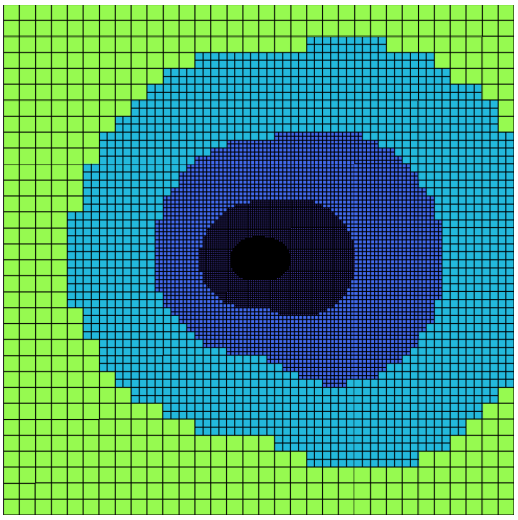
We follow the assumption of Watson [Wat14], that the ganglion cells are organized in two independent grids in separate layers, one for ON-Center or an Off-Center activated RGC. While human vision has later processing steps that can cope with the special structure of RGC in the fovea, we only simulate the ganglion cells in this work. We therefore decide to extend the center-surround properties of the RGC in the rest of the retina to those in the fovea. The center-surround model we use does not hinder the receptive fields in the fovea to be modeled with high overlap. This is achieved by clamping the minimal size of a receptive field to be of the size of an area covered by 9 cones. In the periphery, we follow Daceys observation, that the spacing of the cells roughly equals their size and therefore cover the retina with a factor of about 1.

To calculate the spacing between ganglion cells, we use the formula based on the work of Watson [Wat14] as described above. Figure 4.3a shows the spacing of RGCf in the right eye. We normalized the values to be visible in the color map. The increased densities and hence decreased spacings in the right side of the image correspond to the larger density of M_1 as seen in 4.2a. The strongest decline in density can be seen in the lower left portion of the figure, corresponding to the nasal, inferior quadrant. The transition from the low nasal values to the higher temporal values in the inferior lead to a dip in the figure, running from the center towards the bottom.

Instead of using a hexagonal grid to pack the ganglion cells, we opt for the choice of Aleman et al. [AWS18] to use rectangular cells. This greatly simplifies implementation whilst not diverging too much from the approximation by a hexagon. So instead of having to resort to the area of a hexagon, we can calculate the required length of the rectangular cells from the square root of the density.



(a) Spacing of RGCf with turbo colormap. Values normalized for readability



(b) The grid subdivision, scaled for readability. We added colors to mark areas with equal cell sizes.

Figure 4.3: The RGCf distribution of the right eye and the resulting grid size. Both scaled to be visible in print.

To achieve full coverage in our rectangular grid, with different cell sizes, we use a quadtree. This quadtree is subdivided for each cell until the cell size is smaller than or equal to the cell size calculated from the spacing function. This limits us to be able to only simulate rectangles with a length of a power of 2.

The resulting subdivision is shown in 4.3b. We scaled the density values up to produce a grid visible in the graphics and added colors to highlight regions of the grid with equal cell size. The same shape as shown in 4.3a can also be observed in this graphic. Especially the dip in the region of values transitioning from the inferior nasal to inferior temporal quadrant is prominent.

4.3.4 Retinal Ganglion Cell Receptive Field Sampling

The model of Watson [Wat14] is not concerned whether there is a weighting function to decide if a cell is active based on the individual stimuli of photoreceptors in its receptive field. We discuss the usage of a weighting function in this section.

Since the simulation of the ganglion cells is dependent on the diseases implemented in the retina shader, the RGC calculations have to happen after applying the cone-based transformations. To decide whether ON-Cell or Off-Cell is activated, one needs to take more information into account than is provided by a single photoreceptor and hence more than by the shader execution in the retina. Therefore, the earliest possible place for the ganglion simulation is after the retina shader, although the RGC are still a part of the retina.

How do the sampled pixels, representing photoreceptors, contribute to the decision of a RGC to be activated?

Wienbar and Schwartz [WS18] summarized several models for RGCf activation. While it was found early on, that the ganglion cells exhibit non-linear summation of stimuli [SV79], there is no readily available formula to describe that behavior [WS18]. Some problems with linear spatial models are described by Schwartz and Rieke [SR11] but we want to highlight the following one: Inverse stimuli should evoke opposite reactions. A stimulated surround may not produce any response on its own, but it can completely suppress a stimulated center.

As summarized by Wienbar and Schwartz [WS18], temporal non-linearities can be observed. This encompasses effects like gain control, reducing the firing rate over time if presented with a constant stimulus. Other effects, like firing on the leading edge of a detected movement to anticipate motion are also part of temporal non-linear effects. Due to the complexity of the topic and the missing notion of time in the VSS, we decide not to implement these features and leave their realization for future work.

Necessitated by the problems listed above and concerns regarding performance, we opted not to use all cones available in the receptive field to decide whether a RGC is firing. Instead, we followed the implementation of Aleman et al. [AWS18], using a single, centered sample to represent the center receptive field and a set of 8 samples in a rectangular shape to represent the surround. This sidesteps the need to model the non-linearities observed in RGCf activation. Further, having only 9 samples, this step should not degrade performance much.

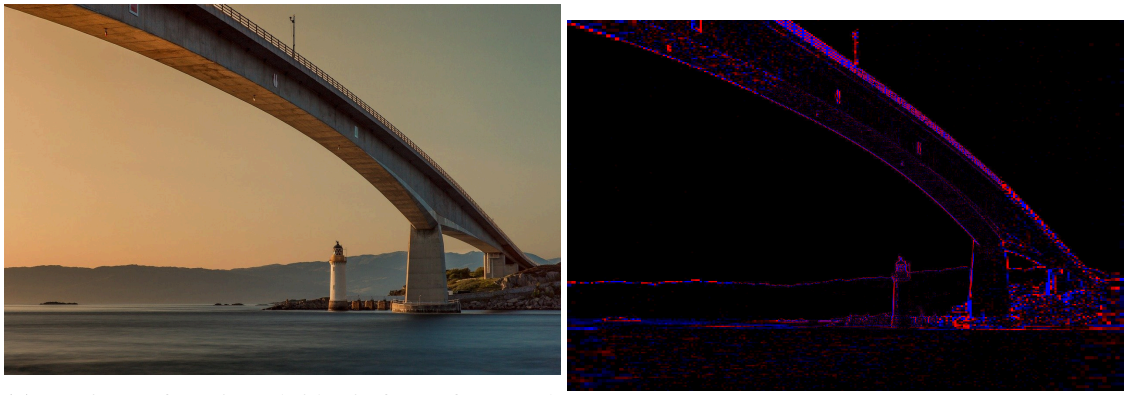
4.3.5 Visualization of Retinal Ganglion Cell Receptive Field Activation

With a set of pixels $p_s = \{p_0, .., p_7\}$ sampled from the surround, and a central pixel p_c, we can decide whether an ON-Cell or an Off-Cell would be activated. In the case that $8 * p_c > \sum p_s$, the ON-Cell is stimulated, otherwise the Off-Cell would be stimulated. Since these are mutually exclusive and we follow the assumption of Watson [Wat14] to have two independent grids of ON-Cells and Off-Cells, only one of the cell types can be active for any given simulated location.

To visualize activated ganglion cells, we follow the model of Aleman et al. [AWS18], indicating On-Cells with a red quad and Off-Cells with a blue quad. To visualize the strength of activation, we also follow their model of multiplying the color value with the relative contrast difference. That way, low contrasts are nearly completely black and one does not has to resort to an arbitrary threshold to prevent noise.

Figure 4.4 depicts a bridge in front of a sky with a smooth gradient and the resulting RGCf activation pattern. The strong contrast between the bridge and the sky, as well as the horizon and the sky, create clearly visible activation patterns. The gradient in the sky is so smooth, that it does not trigger ganglion cell activation. A more detailed discussion of the results alongside many more rendered scenes can be found in 5.3.

[2]Image from https://pixabay.com/photos/bridge-lighthouse-sunset-sea-lake-192986/



(a) Test image featuring a bridge in front of a smooth sky [2]

(b) The ganglion activation pattern for 4.4a.

Figure 4.4: Natural sample scene and the RGCf activation pattern. Due to the smooth gradient in the sky, the activation pattern is nearly completely black in that area.

4.4 Strabismus

We extended the VSS to be able to simulate strabismus. Since the effects and causes of heterophoria are mainfold, we decided to focus our implementation on the simulation of heterotropia. This circumvents the need to make assumptions about many physical properties of the simulated eye like velocities for changing the leading eye or the distance and timing of saccades made by an affected eye. Further, since we do not need to dynamically change the leading eye, we do not need to rely on a mechanism for the user of the simulation to detect or change the leading eye whilst preventing simulator sickness.

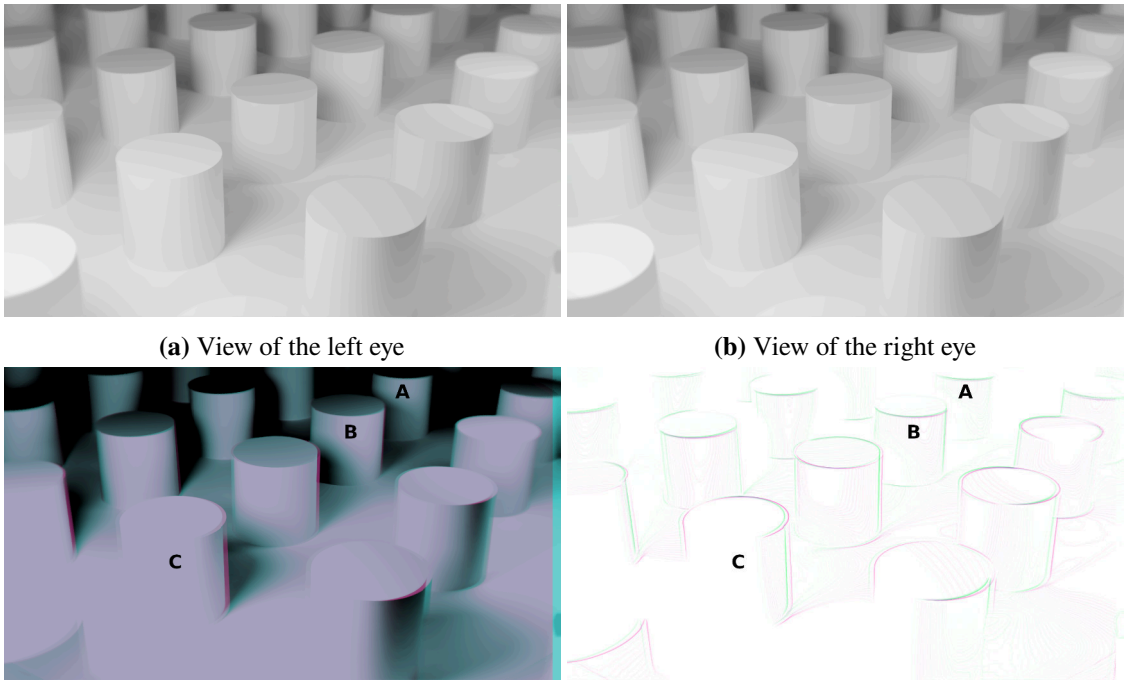
This leaves us with a simulation of heterotropia. As described in 2.2.2, one needs to distinguish between concomitant strabismus and incomitant strabismus. The former describes cases in which the misalignment between the eyes remains constant regardless of the angle of a focused object, whilst the latter includes cases where the misalignment varies with the angle of a focused object. When using a head-mounted display, the simulator has information about the head position but not necessarily about the gaze. Since gaze information is not yet fully supported in the VSS, we have no easy way of obtaining the angle at which the user focuses an object on the screen.

Therefore we restrict our simulation to an implementation covering concomitant strabismus. Since we have no gaze information but only head rotation information, we regard the eye position as fixed. This way, the user can still move their head and with it the simulated eyes. Use cases where the user looks at objects by changing their gaze in a head-mounted display are not covered.

4.4.1 Stereopsis for the simulator

In most cases, strabismus hinders stereopsis. To simulate this effect, we need to first implement stereopsis in the simulator at all.

Although the simulator has depth information available, the provided images are created from a single point of view. This means, that if we move the simulated eye, we will encounter artifacts.



(a) View of the left eye

(b) View of the right eye

(c) The images merged at cylinder 'A'

(d) The edges of the images merged at cylinder 'A'

Figure 4.5: Sample scene showing the views of two eyes.

More involved technologies might enable the reduction or removal of such artifacts. One approach is "LightFields", e.g. as proposed by Marwah et al. [MWBR13], which require an involved capturing mechanism. For working on already captured 360°RGBD videos, Serrano et al. [SKC+19] propose an approach to improve motion parallax and reduce artifacts. Implementing such advanced methods is out of scope for this thesis, so we leave this for future work.

Instead, we resorted to simply moving the eyes in the simulator. We used two independent flows for the individual eyes, allowing completely independent configuration of diseases in the eyes. This is done by running the lens simulation as in the original version but as a last step, after the ray has passed the refractive surface between the cornea and the air, we move the ray according to the eye position. The distance is implemented as a 2-dimensional vector that moves the start position of the last ray segment on the x,y-plane, relative to the head position. The offset in x-direction can be configured in the configuration files. If an offset in y-direction is required, this can easily be added.

Stereopsis scene In figure 4.5 we present a sample scene with two simulated eyes with an Interpupillary Distance (IPD) of 64mm. The value of 64 mm was taken from the mean IPD found by Gordon et al. [GBB+89, p. 150].

Subfigure 4.5a shows the output of the left eye, subfigure 4.5b shows the output of the right eye. Since the difference is hardly noticeable, we merged the two images with the following procedure, resulting in subfigure 4.5c. The outputs of the right and left eye have been imported in an image manipulation program and the image of the right eye has been moved to the left so that it aligns with the left one on the cylinder marked 'A'. We increased the contrast of both images to allow for

more easy detection of features. Then, the image of the left eye has been colorized in a cyan tone and the right one in a magenta one. In a last step, the magenta image was layered on top of the cyan one with 50% transparency.

In cylinder ‘B’, one can see that on the right side, there is a slight pink line and on the left side, there is a cyan one. Since cylinder ‘B’, has a lesser distance from the viewpoint than cylinder ‘A’, we expect the cylinder to have moved more when changing eyes. This effect is stronger the nearer an object is to the camera. Cylinder ‘C’ has a clear pink phantom outline.

However, this visualization only works for bright columns on dark background. If we have dark cylinders on bright background, there appear artifacts This can be observed on the right-most row of cylinders where the shadows cause the cylinders to be darker than the foreground. To verify the implementation is in fact correct, we applied edge-detection and colorized these edges in the above images. The result can be seen in 4.5d. One can verify that the distance between the two colored edges increases, the lower the distance of the object to the camera is.

4.4.2 Implementation of Strabismus

Since we now have the ability to simulate two independent eyes and use the provided depth information to simulate depth perception, we can now implement strabismus.

Strabismus can occur with deviation in vertical and horizontal axis. Although a rotation along the axis of the view direction is also possible, we neglect this case for two reasons: This sort of strabism is rare compared to the other ones and either requires surgery or leads to the output of one eye to be supressed [NJC96].

This leaves us with a configurable rotation along the x axis and the y axis. We use the following rotation matrices to rotate along these axis independently:

$$R = R_x(\alpha)R_y(\beta) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\alpha & -\sin\alpha & 0 \\ 0 & \sin\alpha & \cos\alpha & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \cos\beta & 0 & \sin\beta & 0 \\ 0 & 1 & 0 & 0 \\ -\sin\beta & 0 & \cos\beta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{4.11}$$

We rotate the view after we apply the translation of the head, so that the rotation only rotates the eye and not a part of the head, modifying the eye position. Further, the rotation is done in the rust code, so that it has to be done only once and not for each and every pixel. We expose parameters α, β in equation 4.11 as configuration parameters. Using the command line parameters `--config_right` and `--config_left`, the user can provide these parametes for each eye independently. Note that this allows to define simulations in which none of the eyes focuses correctly. It remains for the user to decide whether this is applicable. Rendered sample images of exotropia are shown in figure 5.5.

4.5 View Modes

To enable easier detection of problematic areas in the investigated scene, we added several new view modes to the simulator. We required the following viewing capabilities:

- Unmodified input image in the current projection
- Rendered simulation
- RGC activation
- Visualization of error and uncertainty with different combination functions
- Overlay of the error and uncertainty over another image

For this, we reworked the way the image is created in the display shader. We identified three base image types, the unmodified input image, the rendered simulating output and the RGC activation pattern. Since the calculation for the ganglion cells is situated in the display shader anyway, this data can easily be acquired. The rendered simulation output is forwarded between the nodes in a flow and since the display node is positioned after the simulation nodes, the color input of the display node is the desired simulation output. The unmodified input image is forwarded between the shader nodes. Fortunately, the input image is not modified after the perspective has been applied to it.

We therefore implemented a shortcut for a node to register a 'well-known' shader resource, in this case, the original image, such that later nodes can use the resource directly as an input. This way we circumvent the need to forward unmodified data in all visual deficiency nodes. The mechanism of registering and retrieving special resources is shown in 4.1 as the gray, dashed lines.

Since all combination functions that we implemented produce a single scalar value for a pixel, we allow the user to specify the combination function as a parameter. Depending on this parameter a value is filled by the corresponding combination function, that can later be visualized. If instead of the current scalar fields, one requires vector fields, a customized solution would be required.

The result value of the combination function can now be visualized in the scalar field. We provide three color maps, that map the scalar value to some RGB vector to achieve a more user-friendly heat map. We implemented the turbo color map and Viridis, as well as a linear grayscale color map.

In a last step, we merge the base image and the color vector. We implemented three modes: Only the base image, only the heatmap and an overlay of the heatmap on top of the base image.

For the overlay, a simple threshold is used. If the result value of the combination function is larger than the threshold, the heat map color is used. Otherwise the color of the base image is used. This approach removes the need of an advanced blend function but produces results with rough edges along iso-values.

The resulting user-configurable parameters are the following:

- Base image: Original image, simulation output, RGC activation
- Combination function: One of the combination functions described in 3.2.4
- View mode: base image, heatmap, overlay
- Color map: Turbo, Viridis or grayscale

The above parameters can be configured by the user at runtime using the keyboard or in advance by supplying them as an argument at application startup. The latter option is especially useful for rendering to file, since in this case, no interaction is possible.

4.6 Attention Guidance

Inspired by the work of Lange et al. [LSGB20], we implemented an attention guidance system based on moving particles. We wanted to produce a proof of concept for this sort of highlighting and therefore refrained from implementing the more complex swarm motion algorithms used by Lange et al. [LSGB20]. Instead, we resorted to a more simple approach. We use the vertices of a cube as the positions for the individuals of our swarm. To give a feeling of swarm motion whilst keeping the relative position of the individuals of the swarm constant for simplicity, we rotate the cube along all three main axes at different speeds. This gives the swarm the illusion of random, unpredictable movement whilst not relying on actual random numbers.

The guidance system is implemented as one of the last steps in the simulator and therefore works with all available view modes. We add the particle color to the current color of the shader, if the current pixel is a part of a particle.

To achieve better integration of the swarm particles into the existing scene, we use smooth Hermite interpolation for the particle. This creates more convincing simulations compared to particles with sharp edges. This is implemented using the `smoothstep` function of glsl.

We implemented manual placement of the guidance marker, so that an operator can highlight an area of an image for the user of the system. It might be desirable to automatically highlight problematic areas of an image according to the values generated by a combination function, as described in 3.2.4. One possible use case for this would be to place the guidance marker at the global maximum of the values generated by a combination function. Unfortunately, the VSS has no way to access information about global maxima from within a shader. We therefore resort to only manual placement of the marker. Figure 4.6 shows the attention guidance pattern in the classroom scene with (4.6b) and without (4.6a) smoothing. Although the implementation without smoothing appears to more easily capture attention in this sample image, we decided to keep the smoothed implementation for two reasons: It blends better into the scene providing a more aesthetic appearance and since the swarm is the only moving part of an otherwise still image, we have no doubts regarding its potential to quickly draw attention.

(a) Attention guidance without smoothing **(b)** Attention guidance with smoothing

Figure 4.6: The attention guidance pattern in the classroom scene. The images show the guidance patter at roughly the same position, with and without smoothing.

5 Results

In this chapter, we describe and discuss the results we obtained based on the extensions made to the simulator. We start by reviewing the newly implemented vision deficiencies and the simulation of retinal ganglion cell activation. Afterwards, we show and discuss visualizations of our uncertainty model in several configurations. The chapter closes with an overview of observed performance characteristics.

5.1 Astigmatism Simulation

In figure 5.1, we present a test scene bases on a real photograph. In figure 5.1a, the test object has been photographed with the normal camera lens. The image in figure 5.1b was made by placing an astigmatic lens of 4dpt in front of the camera lens. Using the original from 5.1a, we configured the VSS to simulate a lens with 4dpt of astigmatism and the same axis as used in the image created with a physical lens. The results are shown in figure 5.1c. Although there is quite a difference between the real astigmatic lens and the simulated one, some of the observed effects are comparable: Whilst the text on the ring became unreadable in both cases, both images retain a clearly distinguishable line in the direction of the astigmatism. Further, the top and lower edges of the circle are comparably defocused, in both images they are less distorted than the right and left corners.

In contrast to the real astigmatic lens, there is no stretch observable in the simulated lens. We are quite unsure why this effect occurs. One possible explanation are effects connected to the way the image was captured: The resulting image was highly sensitive to the lens being placed in the wrong distance. Perhaps the camera lens did interfere with the astigmatic lens in this case.

Especially in the top and bottom regions of the image, the wood texture of the background is bent outwards in the simulated image. This effect is caused by the lens simulation and does occur, even if no astigmatism is simulated.

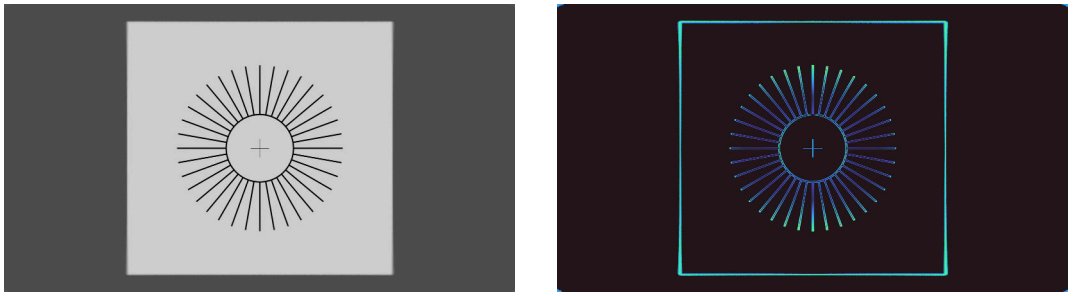
figure 5.2 shows a test scene with different strengths of astigmatism and a constant rotation of its axis. The original image shows a collection of rays typically used by optometrists to determine the axis of an astigmatism. To determine the axis of the astigmatism, one has to identify the least-blurred ray. Simulations with different angles are depicted in 5.3.

On the left side of each subfigure, we show the resulting output image and on the right, we show the heatmap encoding the aggregated standard deviation of the color channels $\|u_{r,g,b}\|$ (see Section 3.2.4). The images in 5.2 use an axis of 0° and strengths of 0, 0.5, 1 and 2 dpt. The images in 5.3 use a strength of 1dpt and axis rotations of 0, 45 and 90 degrees.

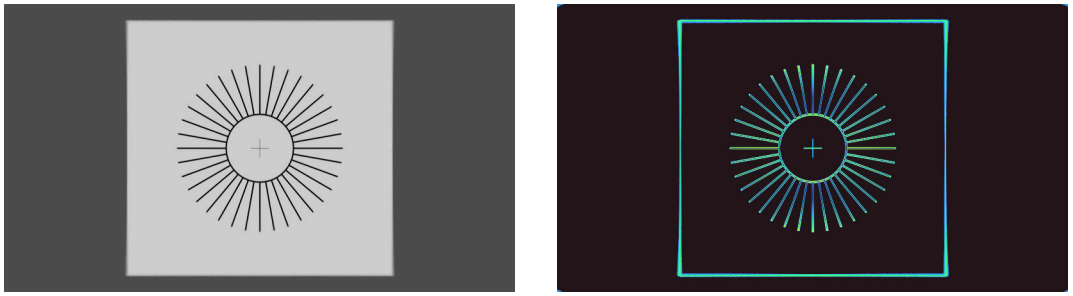
Note that even if we do not simulate any astigmatism like in 5.2a, there is a small amount of uncertainty throughout the image. This is caused by the imperfections in the lens simulation, resulting in a slightly defocused image.

(a) Normal image **(b)** Image with real astigmatic lens **(c)** Simulated astigmatism

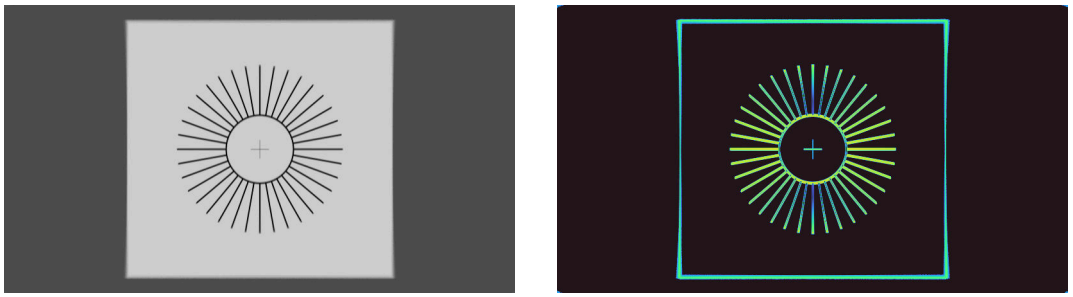
Figure 5.1: A photograph without modifications 5.1a and with an astigmatic lens in front of the camera 5.1b, as well as the simulation based on the original 5.1c



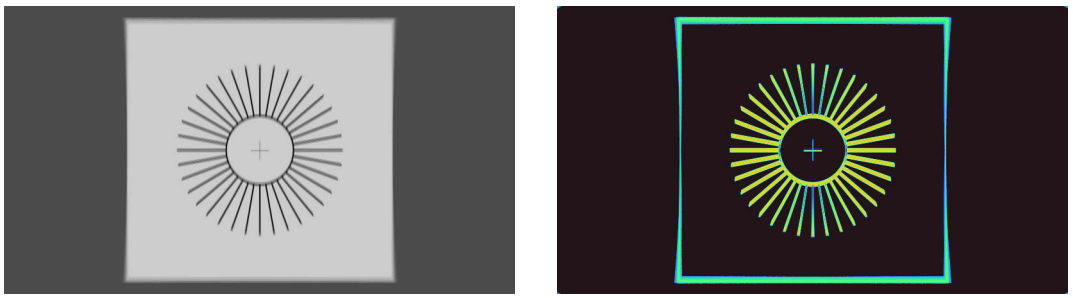
(a) The test scene without astigmatism



(b) The test scene with an astigmatism of 0.5 dpt and an axis of 0°



(c) The test scene with an astigmatism of 1 dpt and an axis of 0°



(d) The test scene with an astigmatism of 2 dpt and an axis of 0°

Figure 5.2: The same test scene with different strengths of astigmatism and a constant axis of 0°. Each subfigure shows the rendered test scene on the left and $\|u_{r,g,b}\|$ (see 3.2.4) on the right as a measure for the strength of the blurred regions. Due to slight inaccuracies in the simulator, 5.2a shows blurred regions although no astigmatism is simulated.

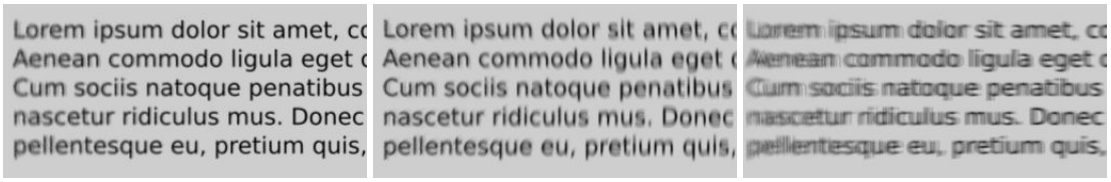
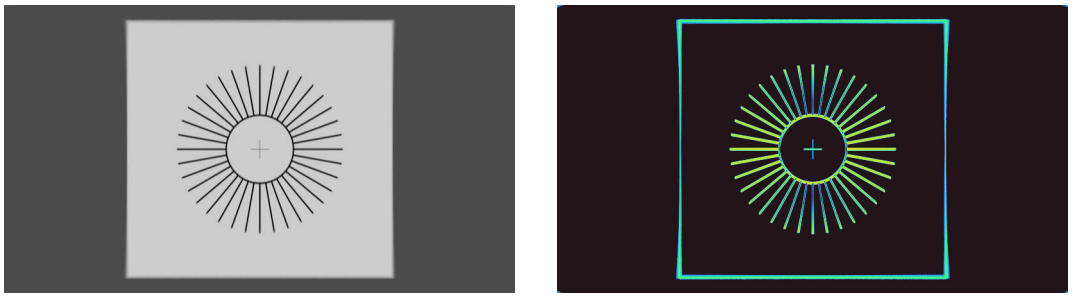
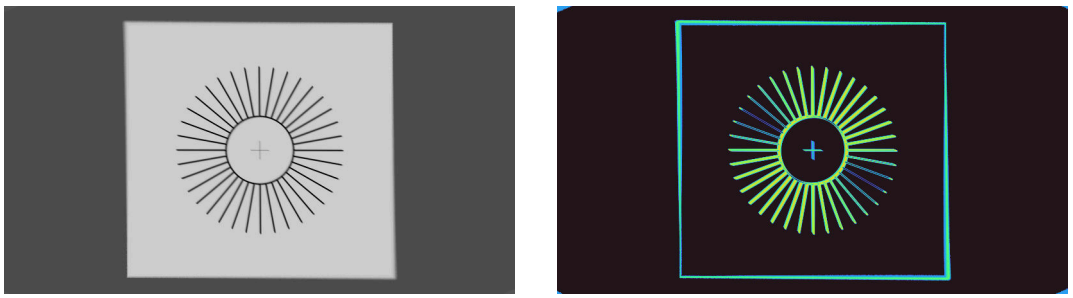


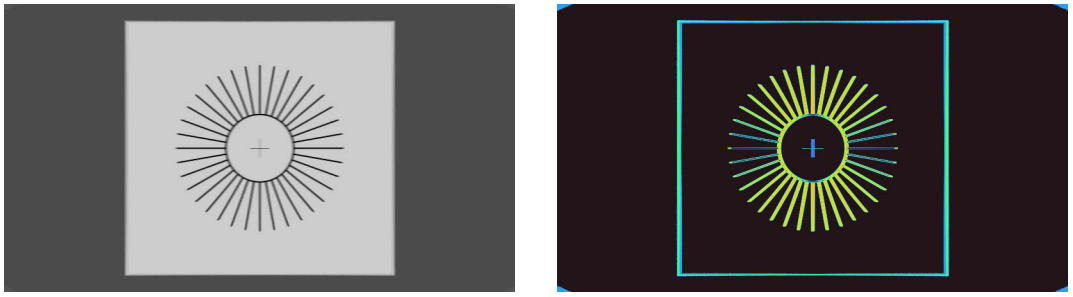
Figure 5.4: A sample scene containing text (magnified). The following configurations of astigmatism are simulated: left no astigmatism, middle 1 dpt 0° , right 1 dpt 90°. On the left side of each image, the readability decreases due to the lens simulation.



(a) The test scene with an astigmatism of 1 dpt and an axis of 0°



(b) The test scene with an astigmatism of 1 dpt and an axis of 45°



(c) The test scene with an astigmatism of 1 dpt and an axis of 90°

Figure 5.3: The same test scene as 5.2 but with astigmatism of constant 1 dpt and a variable axis.

Figure 5.4 shows a portion of an image containing text with different configurations of astigmatism applied. We only show the left part of the image, the right corner marks the middle of the original image. This explains why the text at the left side of the image is barely readable, even if no astigmatism is active: The image quality of the lens node decreases with eccentricity and the left side of the images have higher eccentricity than the right side which corresponds to the center of

the simulated image. Whilst the text in the left subfigure in 5.4, without astigmatism, is clearly readable, it gets blurred if we apply astigmatism. For the subfigure in the middle, we applied 1 dpt at 0°, resulting in a vertical blur. This does however not strongly decrease readability. If we apply 1 dpt at 90°, as shown in the right subfigure, the strongly reduced readability becomes visible quickly. This is in line with reports of patients with the corresponding diseases [Goe93].

5.2 Strabismus Simulation

Figure 5.5 shows the classroom scene as seen by two eyes. The view of the left eye, 5.5a, is in direction of the head, while the right eye has an exotropia of 10°. Although being not that impressive, the images also show the artifacts created by the translation caused by the IPD, especially at the desk directly in front of the camera. To more easily detect the differences in these images, we created an anaglyph in 5.5c.

(a) View of the left eye without exotropia

(b) View of the right eye with 10° exotropia

(c) View of the right eye with 10° exotropia

Figure 5.5: The classroom scene as seen by both eyes with an IPD of 64 mm and an exotropia in the right eye of 10°

It remains to test how well people using head-mounted displays tolerate the simulated strabismus. A study with real participants using the simulator to complete a task might point towards areas needing improvement.

5.3 Retinal Ganglion Cell Simulation

In this section, we present images and their corresponding retinal ganglion cell activation patterns, as rendered by the simulator. Except otherwise noted, we used the simulation as described above with a peak density of $33163 \frac{1}{deg^2}$, clamping the size to be at least one pixel.

Artificial scenes Figure 5.6 shows some renderings of artificial test images and their RGCf activation patterns. The artificial images have been created specifically to showcases some properties of the ganglion cell simulation. Therefore, 5.6a and 5.6c are divided in an upper and lower part, with inverted contrast polarities. All of these images have a size of 1000×1000 pixels.

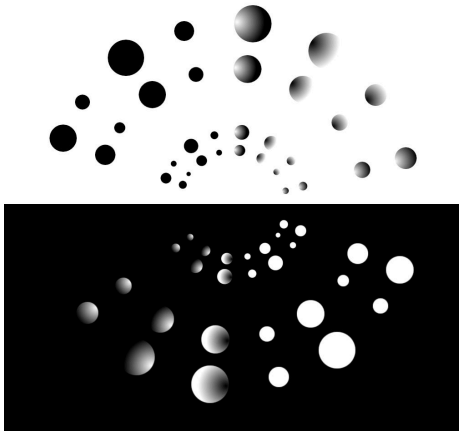
Figure 5.6a shows a set of circles in a circular pattern. Since the circles are larger than the RGCf, we can observe activation solely at their edges. The circles filled with a gradient are not visible as full circles in the RGCf pattern, since the gradient is too shallow to trigger the activation of the

cells. On the edge from the upper part of 5.6b to the lower part, one can observe the variable size of the ganglion cells. Note the inverted polarity of the activation patterns of black circles on white background and white circles on black background.

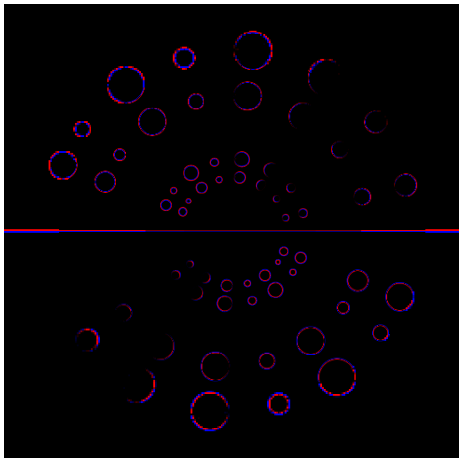
Figure 5.6c shows some text in different font sizes. While the smallest text is readable in the RGCf pattern in the central regions, it becomes unreadable even after the first increase in RGCf size. Despite the strong contrast polarity, the overstimulation observed by Aleman et al. [AWS18] is visible in some cases: An overwhelming amount of active OFF-Cells can be observed with black text on light background, depending on the size of the font and the size of the ganglion cells. For example, the word ‘degree’ in 5.6c exhibits the same off-overstimulation, as observed by Aleman et al. [AWS18]. The word ‘high’ 5.8a does not exhibit this phenomenon.

This problem gets more severe, when we decrease the image size to 500×500 pixels, as shown in 5.8a and 5.8b. In this case, the ON-Cells are overstimulated at the word ‘high’ and the small text begins to merge with the edge between the black and white regions. Due to the limitations of our grid implementation, we argue that the approach of Aleman et al. [AWS18], using multiple uniform RGCf resolutions is more suitable to detect ON-Cell or Off-Cell overstimulation. This is due to their approach seeming to be not so sensitive to changes in text size, as well as reduced dependence on the location of the ganglion grid in relation to the text.

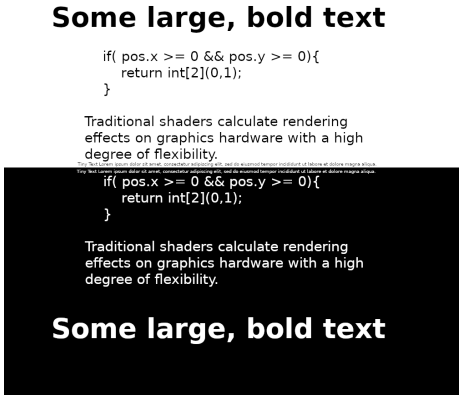
This dependence of our implementation on the location of features is especially visible in the purpose-made grid patterns in 5.6e. While the diagonal grid in the lower right produces sane results, the other three quadrants produce spurious results. The lower-left pattern has a grid size of 4 pixels, which is close to the Nyquist frequency of the ganglion cells, creating aliasing patterns. In the upper half, the grids with 16 and 32 pixels create a different kind of pattern: Depending on the exact sampled position, where the ganglion cell has its center, as well as the result of the operation to convert from the ganglion cell center to the actual pixel grid, the same line can be drawn as blue or red in the grid.



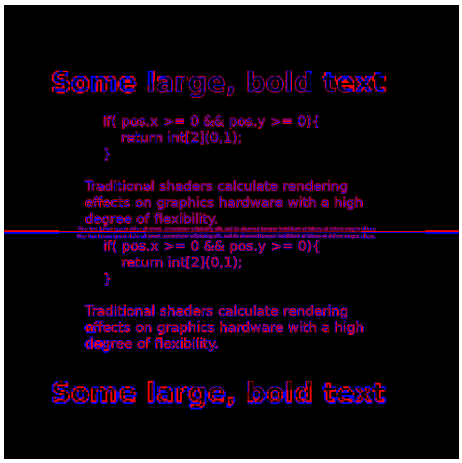
(a) A set of circles in circular patterns



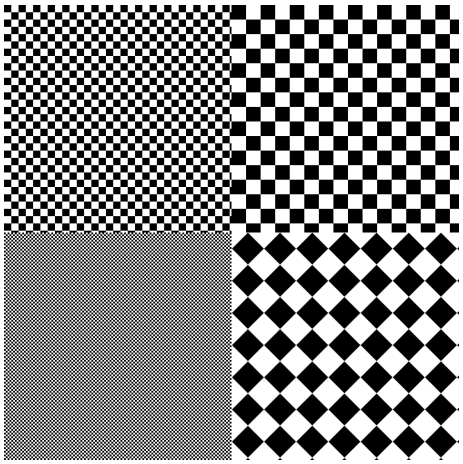
(b) The RGCf pattern for 5.6a



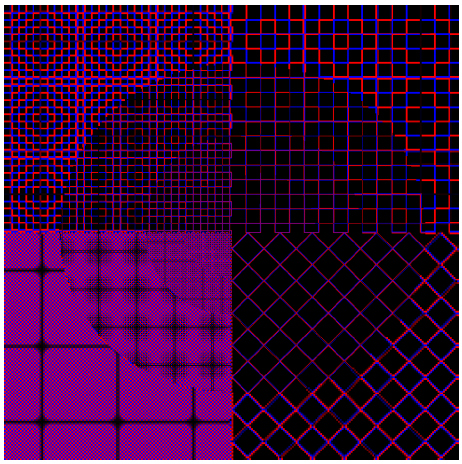
(c) Text of different sizes with inverted contrast polarity. Text from Wikipedia [1]



(d) The RGCf pattern for 5.6c



(e) A pattern of grids of different size



(f) The RGCf pattern for 5.6e

Figure 5.6: Collection of artificial sample images and their RGCf activation patterns

Natural scenes Figure 5.7 depicts several natural scenes with their RGCf activation patterns. These images have been selected for their high contrasts to discuss several properties of the ganglion cell simulation. Although it might look like there is a single, circular reduction in receptive field size, closer inspection will reveal that there are in fact several. We will quickly point them out, using 5.7d as an example: In the lower left, there is an area with the largest receptive field size. From there on, there is a constant size patch covering most of the image, up until the ears of the cat. The next falloff can be seen right below the mouth of the cat. There are more subdivisions according to the spacing function, but since these would require receptive field size smaller than a single pixel, we cannot use them.

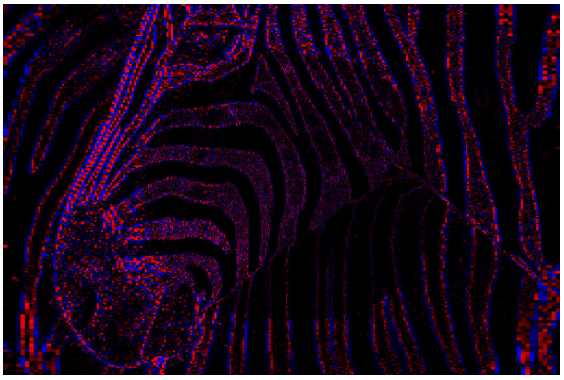
Figure 5.7a depicts a zebra which has high contrast transitions between its stripes. Due to the high granularity, one can even make out individual hairs of its fur in the central parts of the image. On the perimeter, especially in the right top corner, the dark portion of the stripes is so black, that it swallows most details, leaving next to nothing for the ganglion cells to be activated.

Figure 5.7c depicts a cat. Its fine fur at the ears gets picked up well by the ganglion cells. The higher the RGCf density is, the better the features of the cat are visible: Whilst the eyes are outlined by a thin line of active RGCf, the fur in the periphery produces a lot of noise. The black background does not generate any activation in neither cell type. With increasing cell size, the image gets more and more convoluted. This is due to the texture of the cat's fur, which has enough contrast to trigger the ganglion cells.

A differnt effect can be observed in figure 4.4. The gradient of the sky is so smooth, that it does not trigger the ganglion cells, even with large eccentricity. The output is comparable with the black background in the picture of the cat but the input image does in fact contain a gradient. This behavior is as expected, since the implementation of our RGC is focused on large contrasts.

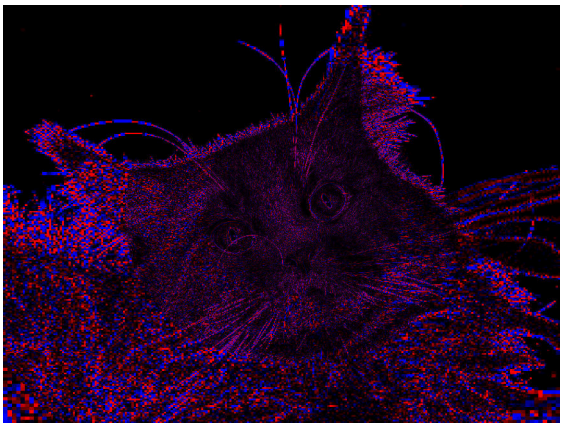
[1] Text from Wikipedia: https://en.wikipedia.org/wiki/Shader

(a) A zebra with large contrast at its stripes [2]



(b) The RGCf pattern for 5.7a

(c) A cat having fine fur with high contast [3]

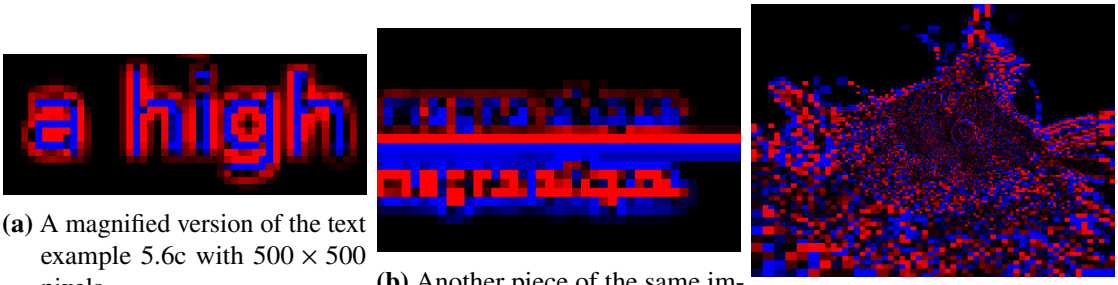


(d) The RGCf pattern for 5.7c

Figure 5.7: Collection of natural sample images and their RGCf activation patterns

Problematic scenes If one would scale the receptive field spacing function, in such a way that the highest field density matches the pixel density of the image, the peripheral ganglion cells reach an unnatural scale. The image of the cat in 5.7c would look like depicted in figure 5.8c, if the field density of roughly $33.000\frac{1}{deg^2}$ would be reduced to match the available 1440 pixels in the image. The same problem can be seen in 5.8a and 5.8b, as discussed in 5.3.

[3]Image from https://www.pexels.com/photo/black-and-white-zebra-47349/
[3]Image from https://www.flickr.com/photos/tambako/50641150937/



(a) A magnified version of the text example 5.6c with 500 × 500 pixels.

(b) Another piece of the same image as 5.8a

(c) The cat from 5.7c with scaled receptive fields

Figure 5.8: Collection of sample images highlighting some problems of the RGCf simulation.

5.4 Uncertainty Visualization

In this section we present visualizations based on our uncertainty model and discuss the benefits and drawbacks of the combination functions. We then demonstrate the propagation of uncertainty in our model based on two scenarios: propagation between the steps of one shader with only one sampling operation and propagation across two shaders with two sampling operations. To discuss possible use cases of the uncertainty visualization, we show the overlay functionality and detail on some of the problems coming with it. Since we expect our model to have an advantage compared to approaches based only on input and output image, we compare our uncertainty visualization to the output of SSIM.

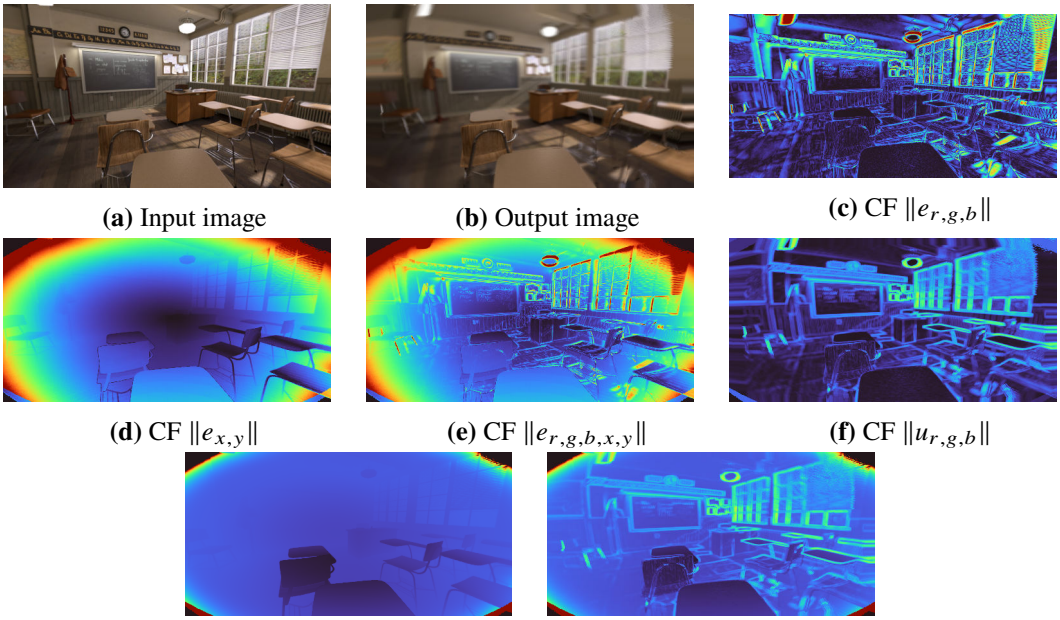
5.4.1 Combination functions

In this section, we present the results of our error metric and the uncertainty visualization. We discuss the individual combination functions in more detail as they allow for different analysis capabilities in different situations. We refer to figure 5.9 to disucss our findings.

The Euclidian Length of the Color Error Vector The length of the color error vector $\|e_{r,g,b}\|$ is a metric that describes the strength of the absolute change in color values.

In figure 3.1c we visualized $\|e_{r,g,b}\|$ for a scene with red-blindness. The test pattern, an Ishihara color test plate used to detect color vision deficiencies in patients, consists of three components: a white background, a set of gray and green dots of varying luminance and the number 42 in red dots, with the dots of the 4 being more saturated. The simulated color deficiency maps the gray and green dots to various gray colors and the red dots to yellow, green or gray tones. The strongest change of the red tones map the color `#d52e3d` to `#8b8a34`, changing the color strongly in two dimensions. The white background is unaffected.

In the visualization of $\|e_{r,g,b}\|$, we can verify that the background is not affected by the color manipulation. Further, the slight modifications of the gray and green dots appear as small values in the heat map. Clearly visible are the dots of the number 42, indicating a large change in color. This way, we can quickly identify problematic areas of an image in regard to the used vision deficiency.



(a) Input image **(b)** Output image **(c)** CF $\|e_{r,g,b}\|$

(d) CF $\|e_{x,y}\|$ **(e)** CF $\|e_{r,g,b,x,y}\|$ **(f)** CF $\|u_{r,g,b}\|$

(g) CF $\|u_{x,y}\|$ **(h)** CF $\|u_{r,g,b,x,y}\|$

Figure 5.9: The classroom scene with a view position of $(457, 151)$, 2dpt of myopia (equals a setting of mnh:17). The subfigures present the input and output images alongside all implemented combination functions. In all heat maps, the turbo color scheme is used.

In figure 5.9c, the areas with the largest values of $\|e_{r,g,b}\|$ are the lamps at the ceiling, the metallic parts of the left-most chair and the upper parts of the windows. These areas correspond to areas with high contrast and strong distortion: each of the areas features a white or almost white area next to a dark brown or grey area. Since $\|e_{r,g,b}\|$ reaches its maximum when the color changes in all three dimensions, these areas are strongly highlighted.

The clock, chalkboard and the alphabet are mostly unreadable in the simulation due to the defocus caused by the simulated ametropia. This is reflected by highlighted areas in the heat map in these places. From this visualization one can quickly identify these areas as problematic ones. If the goal of the simulation was to detect problematic areas in the scene to create a more accessible classroom, increasing the size of the letters and the clock would be a possible approach.

Since the ametropia used to simulate the scene does not include deliberate changes of color values, all visible changes are due to distortion of the image. This effect has two components: The first one is mixing of colors. The ametropia causes areas to be not focused correctly, allowing the rays of the lens simulation to sample different color values. These values are then mixed together to produce the output value of that pixel. The lens simulation uses the average as its mix operation, so the resulting color equals the average of all sampled input colors. In general, $e_{r,g,b}$ is the difference of the input value to the output value of a pixel p_a: $c_{\text{out}}^{p_a} - c_{\text{in}}^{p_a}$. But since the image is defocused for color mixing to occur, $c_{\text{in}}^{p_a}$ is a set of pixels as described in the following. For now, we assume no change in position of the central ray occurs. This is not the case in the lens simulation, see the following paragraphs for more details. If we assume no change in position of the central ray, we have that $e_{r,g,b}^{p_a} = \frac{1}{n} \sum_{i=0}^{n} c_{\text{in}}^{p_i} - c_{\text{central}}^{p_a}$ with $c_{\text{central}}^{p_a}$ being one of the components in the sum and simultaneously the pixel at the same postion as the output pixel.

The second effect is a change in the position of the rays. This can happen with or without mixing of colors caused by a defocused image. Even when the regarded object is focused and no mixing occurs, it is possible that an object is shown in the output image at a different position than the input image. Reasons for this may be a slight magnification or distortion in the periphery. In such cases, $e_{r,g,b}$ is the difference of the colors of the pixels at the output location p_a (the currently processed pixel) and input location p_b : $e^{p_a}_{r,g,b} = c^{p_a}_{\text{out}} - c^{p_b}_{\text{in}}$. Note that p_b is the target location of the central ray as calculated by the lens simulation.

It might be tempting to try to use $c^{p_a}_{\text{in}}$ instead of $c^{p_b}_{\text{in}}$ in the previous formula. However, we have no way to find $c^{p_a}_{\text{in}}$ in the lens simulation, since the simulation without deficiencies does not map input pixels to output pixels at the exact same location. Therefore, given the currently processed pixel p_a, we cannot exactly determine p_a in the input image, let alone its color value. The same behavior can later also be observed with $e_{x,y}$.

This explains why the windows in the upper right corner of figure 5.9c show large $\|e_{r,g,b}\|$ across a large surface: In the original, the window blinds of the right-most window are roughly aligned with the top right corner of the image. In the simulation, they are wrapped towards the left, giving way to the brown color of the wall. This aligns with the bright highlighted area in the heat map. The area of the white window blinds was replaced by the dark brown color of the wall, causing large $\|e_{r,g,b}\|$.

We only implemented this distance metric on the RGB space. However, this color space does not correctly represent the human perception [PA97; XYS01]. It therefore might be desirable to convert $e_{r,g,b}$ to another color space like CieLAB. This might provide results that are closer to human perception [GL99]. To further adapt to the human perception one could use another metric for the distance in color values than the euclidian length. Frameworks like ΔE_{2000} [SWD04] are custom-tailored to solve this problem. This comes at the cost of greatly increased complexity in the required computations and the interpretation of the results is not as intuitive as the euclidian length. Further, ΔE_{2000} was designed to work with small differences in color, if large differences are supplied, ΔE_{2000} falls short [XYS01].

The Aggregated Standard Deviation of the Color Vector For the discussion of the results, we refer to figure 3.2. The cataract introduces some blur, applies a bloom effect and lowers the contrast of the image according to the configuration. In the heatmap, one can see that of the four left squares, the black one has the strongest outline. This is due to black (`#000000`) having the largest contrast compared to the white (`#FFFFFF`) background and hence the uniform blur causes the sampled color values to have the maximum possible range in each dimension. The other colors only have the maximum possible range in two dimensions, e.g. red (`#FF0000`) to white. The Gaussian blur kernel of the cataract operation then causes the sampled values at the edges to have a large variance in the black square. If magnified, one can see that in the heatmap, the boxes and text have no uniform outline but feature a gradient. This is due to the blur kernel reaching fewer pixels with a different color than the mean with increasing distance from the feature. This will be important later on, since a blurred output image will always produce a blurred heatmap using this combination function.

The gradient box on the upper right corner shows decreasing values in the aggregated standard deviation with decreasing contrast. This behavior is as expected since the variance in the color samples also decreases with the contrast.

The patterns in the lower right corner showcase the probably most important feature of this combination function: If a vision deficiency reduces or even destroys the visibility of image features by blurring them, a visualization using $\|u_{r,g,b}\|$ shows the problematic areas. The distance and size of the displayed stripes are decreasing in each row and the spatial intensity of the heatmap increases accordingly.

In figure 5.9f, we present a visualization of the aggregated standard deviation of the color vector for the classroom scene. The areas with the largest $\|u_{r,g,b}\|$ are mostly the same as those of $\|e_{r,g,b}\|$: the lamps, the chair legs, the windows as well as the letters and sheets of paper on the wall. In addition, this combination function also highlights the edges of the desks on the right side. Although one can verify that these are in fact defocused by comparing the input and output images, the impact of the caused blur is comparably small.

In contrast to $\|e_{r,g,b}\|$, this combination function does not suffer from undesired sensitivity to displacement and hence does not highlight the window shades on the right. Unfortunately, this also means, that all distortions in the resulting image are also contained in the heat map. Further, the heatmap exhibits the same blur as the result image, caused by the way the uncertainty is tracked. This distortion is even more severe in figure 5.12: The upper right cube exhibits large displacement and deformation.

The Euclidian Length of the XY-Error Vector The euclidian length of the xy error vector simply describes the distance a pixel was moved from the input to the output image. Due to the aforementioned problems with exact pixel positions in the lens simulation, this metric has some error on its own.

In figure 5.9d we show a visualization of $\|e_{x,y}\|$ for the classroom scene. The distortion of the lens when using myopia quickly becomes apparent at the edges. There is a gradient ranging from the center of the image to the corners with increasing severity of deflection. In the corners, the deflection eventually leaves the supported value range. The values of $e_{x,y}$ were normalized within a range of 0 to 200 pixels to achieve visible results. If this step would have been skipped, the values of e_x and e_y would be so small that interesting features would not show up on the heatmap at all and only the artifacts in the edges would be visible. The range of 0 to 200 pixels was chosen for this scene by experimentation and needs to be adjusted if used with other configurations.

If this metric is used without depth information, all currently implemented vision deficiencies of the VSS produce a uniform value of zero.

The Aggregated Standard Deviation of the XY Vector The aggregated standard deviation of the xy vector describes the radius of the standard deviation of a rotational symmetric sample distribution. The larger this radius is, the more dispersion was observed in the sampling process. However, this measure does not directly correspond to the perceived blurriness of a scene: $\|u_{x,y}\|$ is only concerned with the positions of the samples, not the color values. This means that even if $\|u_{x,y}\|$ might indicate a region with a large sampling radius, there might be no visible blur if the area is covered in a uniform color.

Figure 5.9g shows a visualization of $\|u_{x,y}\|$ for the classroom scene. One can observe the same distortions at the edge of the visual field as with $\|e_{x,y}\|$. To make the interesting features of the scene visible, we also scaled the vector components in this image. We chose to normalize the values of $u^2_{x,y}$ to 0.0185 which represents an experimental value.

Since the whole scene has textured surfaces, the visualization gives an intuition for blurred regions: The back of the chair directly in front of the one the camera is positioned on, alongside with the desk directly in front of the camera have the lowest values for $\|u_{x,y}\|$. If compared with the simulation output, one can observe that these regions indeed are not blurred. Since a low value of $\|u_{x,y}\|$ indicates a small dispersion in sample positions, this rules out blurred regions for all but the most high-frequency textures. We therefore conclude that a low value of $\|u_{x,y}\|$ is an indicator for unproblematic areas in a scene.

As with $\|e_{x,y}\|$, $\|u_{x,y}\|$ only produces interesting results, if depth information is provided. However, instead of producing a value of zero, diseases like the cataract produce uniform values proportional to their configured blur scale.

RGBXY Vectors Both, $\|e_{r,g,b,x,y}\|$ and $\|u_{r,g,b,x,y}\|$, combine properties of the combination functions using only a subset of their components. Figure 5.9e shows $\|e_{r,g,b,x,y}\|$ and figure 5.9h shows $\|u_{r,g,b,x,y}\|$ for the classroom scene. Whilst $\|e_{r,g,b,x,y}\|$ does not really provide much insight in this scene, the usage of $\|u_{r,g,b,x,y}\|$ adds some value to the visualization compared to using $\|u_{r,g,b}\|$ and $\|u_{x,y}\|$ alone. One benefit is that it highlights the distorted edges of the visual field, making it clear that problems located there are not important for the current scene. Further, it adds some offset to the whole scene, thereby reducing the value of the desks and chairs relative to the ground and walls. This adds more context to the visualization since the least affected regions have lower values than the rest.

The two 5 dimensional vectors might further be exported to be used in external tools to allow for more in-depth analysis.

5.4.2 Propagation of Uncertainty

We show the effect of the propagation of uncertainty by presenting the cube scene with a cataract vision deficiency with the individual operations of the cataract shader disabled. The cataract shader is comprised of three main operations: A blur operation, a bloom operation and a contrast modification. Figure 5.11 shows $\|u_{r,g,b}\|$ for the three stages: In figure 5.11a only the blur operation is enabled, in 5.11b the blur and bloom operations are enabled and in 5.11c all of them are enabled.

The sampling operation in the blur uses the sample variance to compute $u_{r,g,b}$ (see 3.3) without already existing uncertainty. The result of this operation is shown in 5.11a. The aggregated standard deviation in this image favors edges in the floor tiles, that exhibit large differences in the RGB space. The edge at ‘B’ from yellow (`bf9800`) to pink (`b21f76`) features a larger distance in color values than ‘A’ with purple (`271c5d`) to blue (`1f3264`). Since all tiles are mostly uniformly colored and there are not large differences in the size of the sampling areas, so that they would stretch more than one tile, the edge at ‘B’ has larger $\|u_{r,g,b}\|$, as observable in the visualization.

The bloom operation propagates the uncertainty using the Jacobi matrix J_{bloom} as in 5.10a. Since the bloom operation multiplies the color values with a factor greater one, we expect the resulting uncertainty to increase as well. This can be observed in 5.11b: Areas that already contain some uncertainty and are located on bright areas of the image have their values increased. This effect is especially prominent at the edges and corners of the large cube and the lower one but can also be observed on most lines on the floor grid. Some areas, like the inner parts of the floor tiles and the area marked as 'A' in 5.11b, the values do not increase at all or only by a barely noticeable amount. This is due to either their uncertainty being low, their image color being dark or both of them.

The contrast operation adjusts the value of the lower two channels to be closer to the value of the channel with the largest color value. For propagation, we use J_{contrast} as in 5.10b.

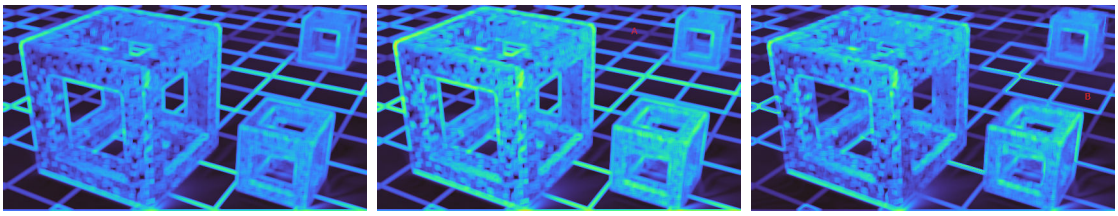
$$J_{\text{bloom}} = \begin{pmatrix} c\ (w \cdot x) + cw_r x_r + 1 & cw_g x_r & cw_b x_r \\ cw_r x_g & c\ (w \cdot x) + cw_g x_g + 1 & cw_b x_g \\ cw_r x_b & cw_g x_b & c\ (w \cdot x) + cw_b x_b + 1 \end{pmatrix}$$

(a) Jacobian matrix of the bloom operation. We use $\cdot$ as the dot product to shorten the equation. The current pixels color value is x, w are the weights used to simulate human perception and c is a constant configured by the user.

$$J_{\text{contrast}} = \begin{pmatrix} 1 & 0 & 0 \\ c & 1-c & 0 \\ c & 0 & 1-c \end{pmatrix}$$

(b) Jacobian matrix of the contrast operation if the red channel is the strongest one. c is a constant configured by the user

Figure 5.10: Jacobi matrices for the bloom and contrast operation



(a) $\|u_{r,g,b}\|$ after only the blur operation

(b) $\|u_{r,g,b}\|$ after the blur and bloom operations

(c) $\|u_{r,g,b}\|$ after the blur, bloom and contrast operations

Figure 5.11: Aggregated standard deviation of individual steps in the cataract shader

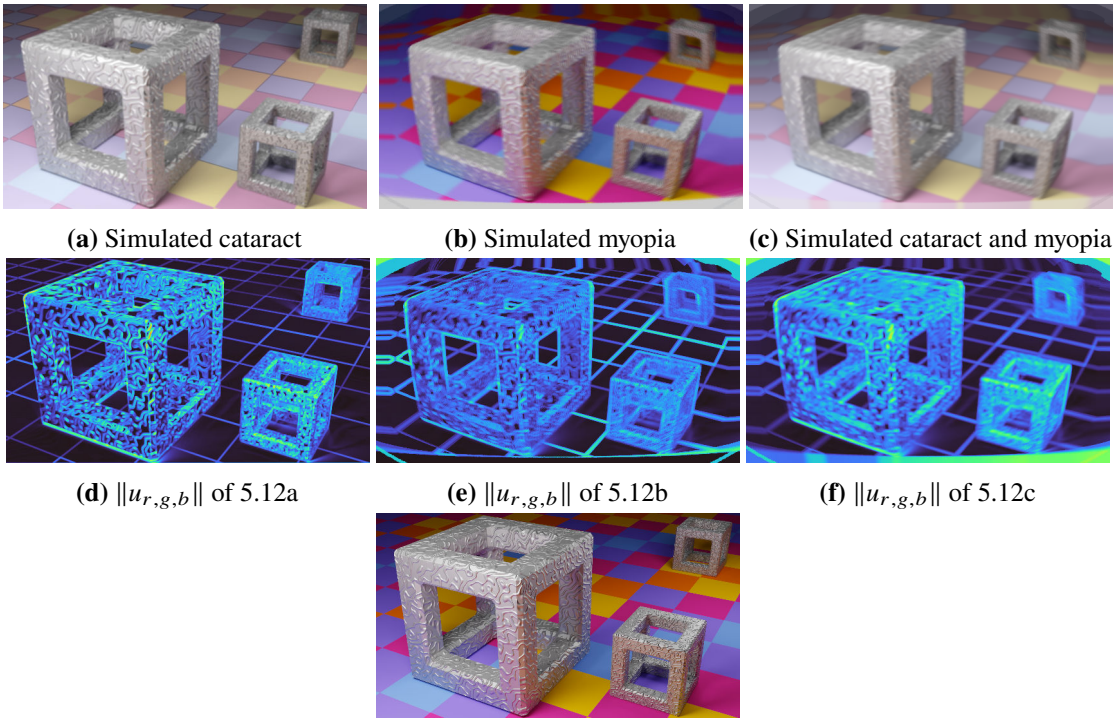
To showcase the propagation of uncertainty across multiple sampling operations, we use the same cube scene, but with a different configuration: A combination of cataract (weaker than in 5.11) and myopia is simulated. The result is shown in 5.12c. We also rendered the two visual deficiencies individually, as shown in 5.12a and 5.12b. The visualizations of the aggregated standard deviation are shown for the individual and combined simulations in 5.12d, 5.12e and 5.12f.

Both operations, the myopia and the cataract incorporate a blur operation that is implemented by sampling pixels from the respective input images. The myopia simulation, being a part of the lens node, is executed after the cataract simulation. Therefore, the pixels sampled by the myopia simulation already contain the uncertainty of the cataract simulation. This existing, observed uncertainty and the new uncertainty, created by the sampling operation itself, are combined according to equation 3.3.

We can observe the effects of this combination in 5.12f: Features of both heatmaps of the individual simulations are present in the combined image. Examples for this are the strong highlights on the camera-facing corner of the large cube originating from the cataract simulation and the strong highlights on the floor-tiles between the large cube and the cube in the background.

The image quality of the visualization for the combined simulation is worse than the image quality of the other two heat maps. This is caused by the propagation of uncertainty in the lens: The color and uncertainty values are read by the same sampling operation causing a blur in the color image. Therefore, this also causes a blur in the visualization.

The artifacts at the lower edge of the image are quite interesting: Since the original image has a slight one-pixel thick artifact line at the end of the image, the myopia simulation increases the thickness of this artifact by a lot. If the cataract is executed before the myopia, the blur of the cataract fills that line with the colors of neighboring pixels. If then the distortion of the myopia is applied, that color information creates a bleeding effect in both, the color image and the visualization of the aggregated standard deviation.



(a) Simulated cataract **(b)** Simulated myopia **(c)** Simulated cataract and myopia

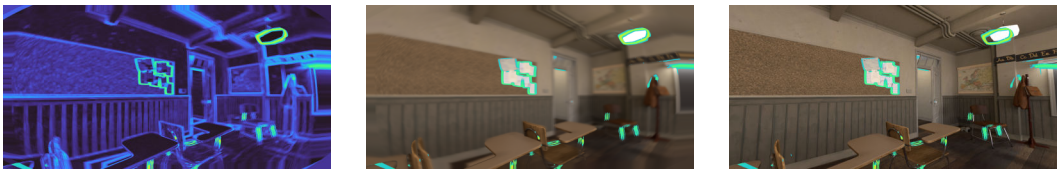
(d) $\|u_{r,g,b}\|$ of 5.12a **(e)** $\|u_{r,g,b}\|$ of 5.12b **(f)** $\|u_{r,g,b}\|$ of 5.12c

(g) The original image

Figure 5.12: Propagation of uncertainty across multiple sampling operations shown with the cube scene.

5.4.3 View Modes

The results of the different view modes are demonstrated in another test scene, shown in 5.13. We used a hyperopia of 2dpt to create these images within the classroom scene.



(a) The heatmap as a base for the overlays **(b)** Blended highlighting on top of the input image **(c)** Blended highlighting on top of the input image

Figure 5.13: Comparison of different view modes

Figure 5.13a shows the heatmap visualizing the aggregated standard deviation. Figure 5.13b shows the same heat map but displayed using the overlay view mode, using the output image as the base image. Figure 5.13c shows the overlay on top of the input image.

Regarding the overlay on top of the input image, whilst the highlights align with the problematic features in the center of the image, this is not the case in the periphery. This is caused by the distortion introduced in the lens. One can see the same distortion when comparing the input and

output image. Unfortunately, this problem can not easily be fixed, although the change in image coordinates is recorded in $e_{x,y}$. To illustrate, imagine the following scenario: Some pixel A is mapped to a pixel B in the lens shader. If we assume the lens node to be the only active visual deficiency node, the distance between A and B is saved in $B_{e_{xy}}$ and the sample variance in $B_{u_{RGB}}$. If later the display node processes pixel A, it does not know where to look for pixel B and its uncertainty. Further, when processing pixel B in the lens node, we could calculate the position of pixel A but we cannot influence the value of A from there. A possible solution that could be evaluated in future work, is the usage of the lens shader to reverse the refraction process when processing A, to determine the position of B, look up $B_{u_{RGB}}$ and use this as the output value. This approach might correct the misalignment between the original image and the overlay in presence of strong distortion but requires another costly execution of the lens shader.

Regardless of the misalignment, one can quickly determine the problematic areas in the scene. Depending on the combination of the scene, head position and simulated deficiency, it might be required to adjust the threshold to keep the highlights from getting too large and thereby breaking the immersion.

5.4.4 Comparison with SSIM

We expect our uncertainty model to have several advantages compared to approaches based on a comparison of the input and output image. Therefore, we compare our uncertainty visualization with SSIM and highlight advantages and drawbacks of both technologies.

Since SSIM outputs a measure for similarity, we inverted the value to easier detect dissimilarities. This aligns better with the visualizations produced by the simulator.

The first scene we use for comparison is again the classroom scene, with the previously used myopia. We did change the view angle to include more of the window front to have the window blinds take up a larger part of the image. Figure 5.14a shows the orignal image, 5.14b shows the simulated image. We compare the visualization of $\|u_{r,g,b}\|$ in 5.14c to the output of SSIM if fed with the input and output images in 5.14d.

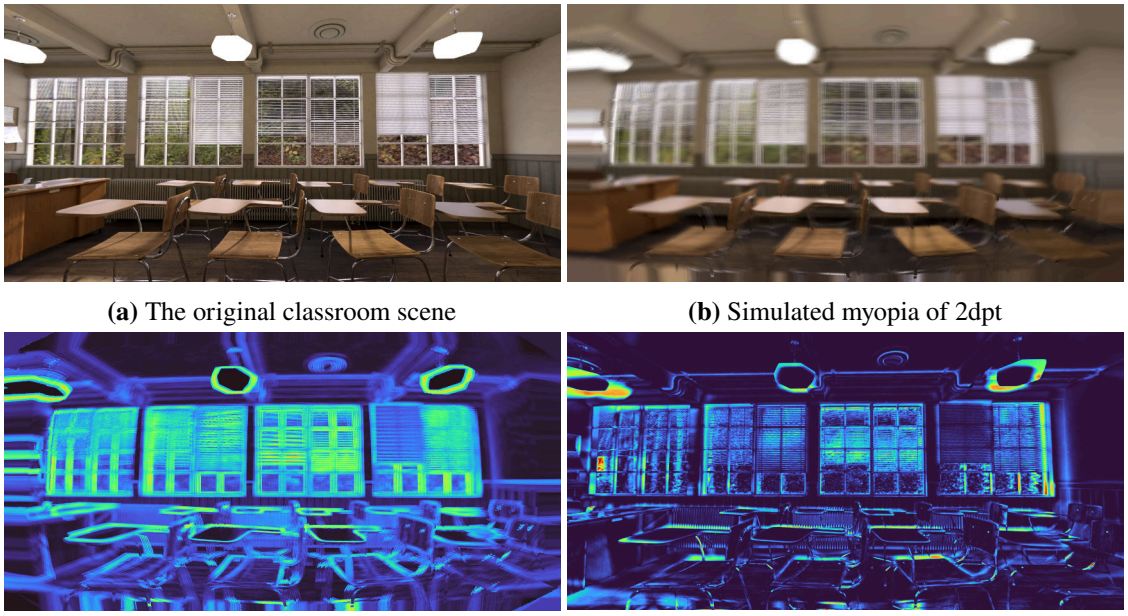
Our visualization suffers from the same blur that was observed in previous images. The visualization of SSIM however does not have this shortcoming.

There are three main features in this image that we want to discuss in the following: the lamps, the window blinds and the desks. Starting with the lamps, the strong offset of the lamp and its highlighting in both positions shows that SSIM has no information about the image creation process. This is not surprising, since SSIM only has the input and output images as reference. This shows that SSIM is sensitive to displaced features in a scene, what will become more important in the cube scene in the next example. Although this might produce artifacts, this is not a deficiency of SSIM: Since SSIM is used to detect differences in images and the lamps certainly are a difference between the input and output images, SSIM correctly reports it as such. Depending on the use case, this behavior might or might not be desirable.

The second feature to discuss are the window shades. In the original image, they form thin white stripes in front of a much darker background. In the output image, they are blurred but still exhibit a similar structure. We suspect this comes from the large area that is covered by comparably few rays in the lens shader, some of which hit a window blind and others do not. Since there is a high

contrast between window shade and background some of the sampled pixels produce large values for $\|u_{r,g,b}\|$. Although the areas with the largest values coincide roughly with those of SSIM, it remains to decide whether such a strong focus in the shades is appropriate for the desired use case.

The last feature to discuss with this scene are the desks and chairs in front of the windows. The first row of desks has highlights in both, the VSS and SSIM output: whilst SSIM shows large values directly above the desks due to the displacement, the VSS output highlights the whole shape of the desks. Especially the leftmost edge of the leftmost desk is nearly not highlighted at all in the SSIM output. Depending on the use case, this might be problematic: If one wants to identify objects within the scene that might be hard to navigate around by people with the simulated deficiencies, the SSIM is of little help. If the use case is to find areas of the image that were changed by the simulation, using SSIM is the better choice.



(a) The original classroom scene

(b) Simulated myopia of 2dpt

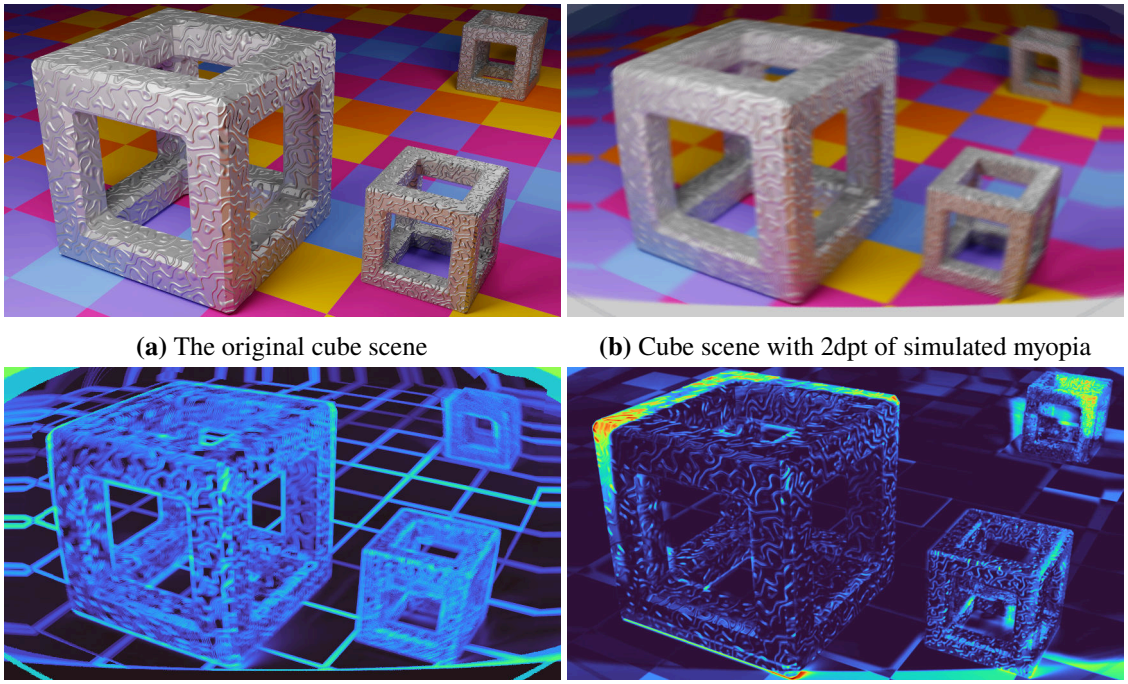
(c) Visualized $\|u_{r,g,b}\|$

(d) Visualized SSIM

Figure 5.14: Comparison of SSIM and $\|u_{r,g,b}\|$ in the classroom scene

We use the cube scene to further highlight the differences between SSIM and our metric. The orignal image and the simulated output are shown in 5.15a and 5.15b. The visualization of $\|u_{r,g,b}\|$ is shown in 5.15c, the corresponding output of SSIM is shown in 5.15d.

As with the classroom scene, we can observe the sensitivity to displacement of SSIM. This is especially prominent in the area of the background cube and the left edge of the large cube. Our metric does isolate displacement into $\|e_{x,y}\|$ and therefore allows to focus with $\|u_{r,g,b}\|$ on the problems caused by the object being out of focus. In certain conditions, this might not be desirable: Since displaced objects in an image can lead to situations that are hard to navigate for people with the simulated vision deficiencies, neglecting displacement might not be viable. In such cases, one can combine several of our tracked vector components, e.g. mix $\|e_{x,y}\|$ and $\|u_{r,g,b}\|$ in a suitable way.



(a) The original cube scene

(b) Cube scene with 2dpt of simulated myopia

(c) Visualized $\|u_{r,g,b}\|$

(d) Visualized SSIM

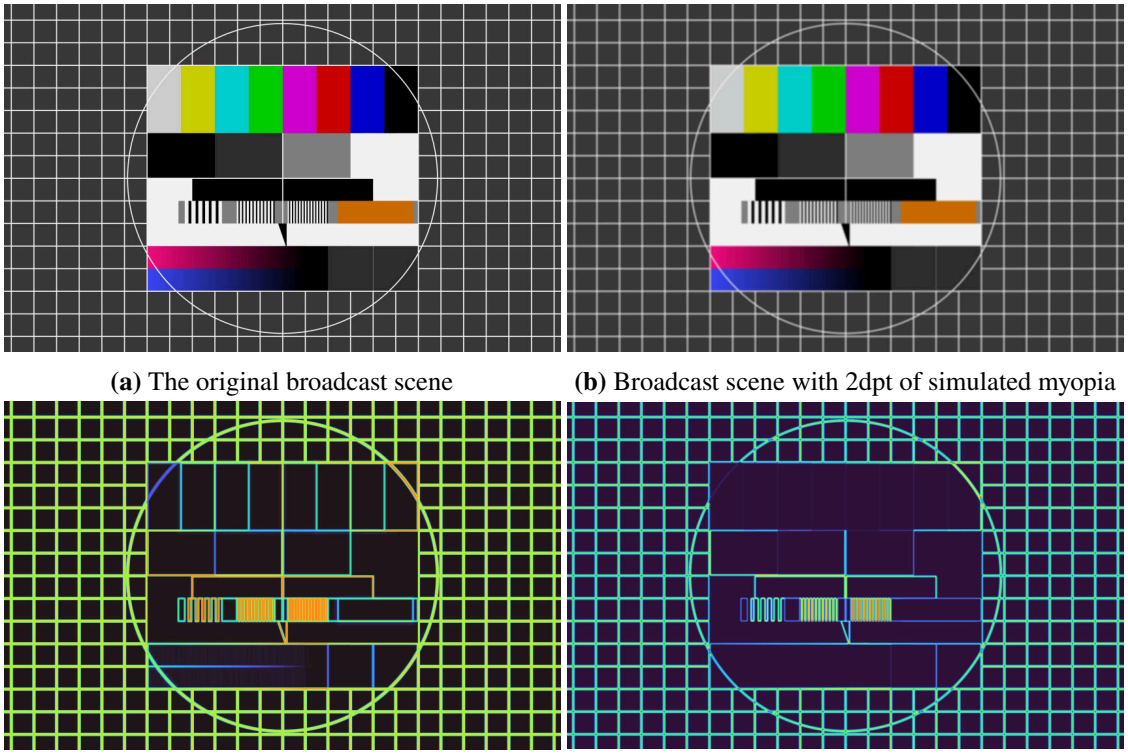
Figure 5.15: Comparison of SSIM and $\|u_{r,g,b}\|$ in the cube scene

Since SSIM is focused on comparison based on luminance, it has no notion for color in its original way. So if two colors have comparable luminance values, a blurred edge between them can not be detected by SSIM. This can be observed in 5.15d at the floor tiles between the left corner of the right cube and the cube in the background: The orange and purple tiles have nearly the same luminance values, hence the value picked up by SSIM is nearly the same and there is no dissimilarity detected between the input and output image, if their border is blurred. In contrast to this, our metric tracks uncertainty for each color channel separately and the combination function is only applied at the end. Hence, $u_{r,g,b}$ has a non-zero length in this area.

Following the results in the previous section, we created one last test scene to compare SSIM and our metric. Figure 5.16a shows a broadcast test pattern, to which we applied a slight blur in the cataract node. The output image is shown in 5.16b. Note that we deactivated all other operations of the cataract to achieve a simpler test case. Figure 5.16c shows the output of our metric and 5.16d that of SSIM.

The insensitivity of SSIM to color becomes apparent in several places: The border between the two gradients is missing in SSIM. Further, the orange bar is not separated from the grey surround in SSIM. There are several approaches to adapt SSIM to work better with color images, for example the work by Bhateja et al. [BSK14]. It is left for future work to compare our metric to these approaches.

Both images pick up the blurred lines of the grid well. Upon closer examination, it becomes apparent that SSIM splits the grid lines into their two edges. This leads to points at the intersection of lines, that have a value close to zero. A similar behavior can be observed in our metric: the



(a) The original broadcast scene

(b) Broadcast scene with 2dpt of simulated myopia

(c) Visualized $\|u_{r,g,b}\|$

(d) Visualized SSIM

Figure 5.16: Comparison of SSIM and $\|u_{r,g,b}\|$ in the broadcast scene

values at the intersections are lower than those on the rest of the line. We suspect that this behavior can be changed by tuning the parameters used for SSIM window size. It remains for the user to decide what behavior is desired in each specific situation.

Since our error metric allows the independent analysis of introduced errors and uncertainty, one can configure the visualization to be insensitive to changes in position of features. This is one of the main differences when compared to SSIM, which has no way to distinguish between the two. We therefore conclude, that our metric is especially useful if one desires to analyze a scene while acknowledging the existence of errors in color or position. Due to the way we gather and track uncertainty, we are not able to produce visualizations of uncertainty with a fidelity comparable to SSIM. If only errors are visualized, this problem does not exist. Given that one can extract the values of all ten tracked error and uncertainty dimensions, advanced filtering and analysis in external systems is possible but is left to future work to be explored.

Parameter:	Possible values:
Ray count	1, 5, 9, 17
Resolution (pixels)	1280x720, 1920x1080, 2560x1440, 3840x2160, 5120x2880
Error tracking	on, off

Table 5.1: Parameters for the benchmarks

5.5 Performance Characteristics

To evaluate performance characteristics of the VSS, we measured rendering times of the simulator for several configurations. Since the lens node is the one with the strongest impact on performance, we chose to base this benchmark on the myopia simulation. We measure frame timings with variable resolutions, enabled and disabled error tracking, monocular and binocular simulation, as well as different amounts of simulated rays.

The benchmarks were conducted on a PC with an AMD Radeon RX 480 GPU with 8192MB of video memory and an AMD Ryzen 9 3900X Processor. We used Linux 5.9.16 with an amdgpu driver of version 20.3.4.

We used the cartesian product of the parameters described in table 5.1 as configurations for the simulator. Unfortunately, the GPU did not have enough video memory to create all necessary buffers for a binocular simulation of $5120x2880$ pixels each. We therefore did not include that benchmark in the results.

Figure 5.17 shows the mean times the GPU needed for a single frame in each combination. The error bars show the observed standard deviation of the recorded times. Figure 5.17a shows the results for a single simulated eye, 5.17b shows the results for two simulated eyes, as they would be used in a VR setting. Due to the double buffering strategy in the VSS, we could not achieve frame rates above 60 frames per second without stability issues. We therefore left vsync active, limiting framerates in the benchmark at 60 frames per second.

The simulator generates several maps, e.g. the retina map at startup in a time-consuming first step. We therefore removed the measurement representing the first frame.

Depending on the selected view mode, the simulator tracks the introduced error. This is disabled, for example when viewing only the simulator output. By selecting the view modes accordingly, we activated and deactivated the error tracking. Note however, even with deactivated error tracking, some of our modification to the VSS are still active and may cause performance penalties when compared to the initial version.

The limit of 60 frames per second, corresponding to 16.67 ms per frame, can be seen in 5.17a as the lower limit of frame times. Although it might be interesting to analyze performance when running with more than 60 frames per second in more detail, we expect large amounts of work to be required to migrate the VSS to libraries that produce stable results when running with more than 60 frames per second.

As can be seen in both configurations, monocular and binocular, resolution has the strongest influence on performance. For the monocular configuration, we can observe that the time per frame is roughly proportional to the number of pixels, with the limitation of 16.67 ms being the minimum.

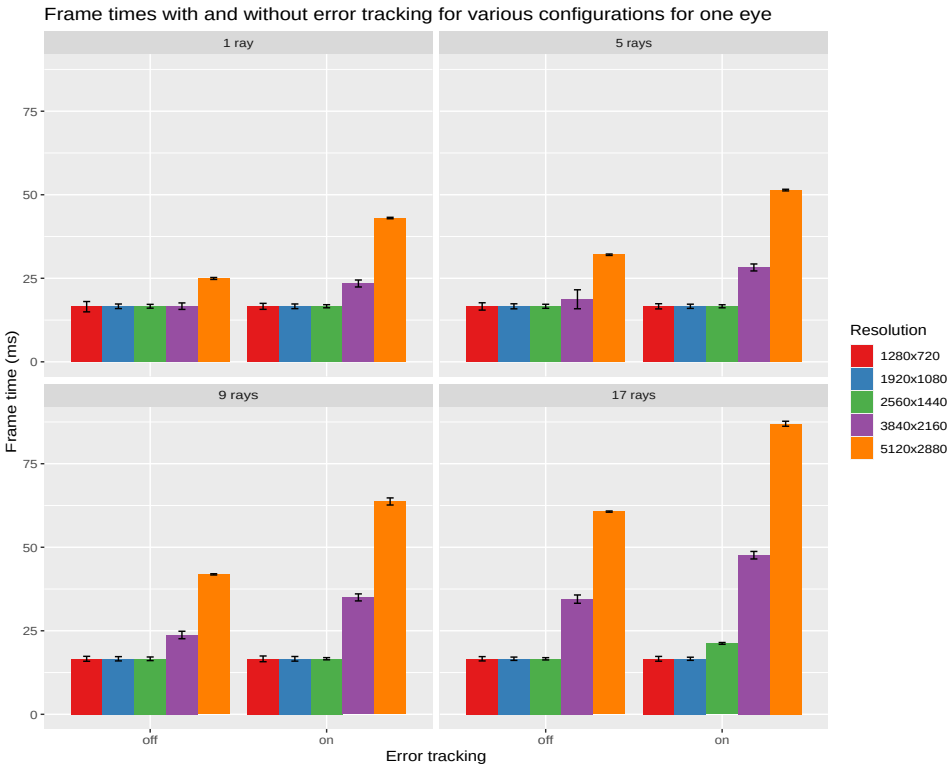
For the configuration with 17 rays, starting with 2560x1440 px, 2.25 times more pixels lead to 2.243 times more rendering time and 4 times more pixels require 4.097 times more time when error tracking is enabled.

The ray count does decrease the performance but since the ray shader only is a part of the whole simulation, other operations like simple forwarding of color values between the shaders are not affected. Thus, a 17-fold increase in simulated rays does not increase the time needed for the whole shader execution by 17.

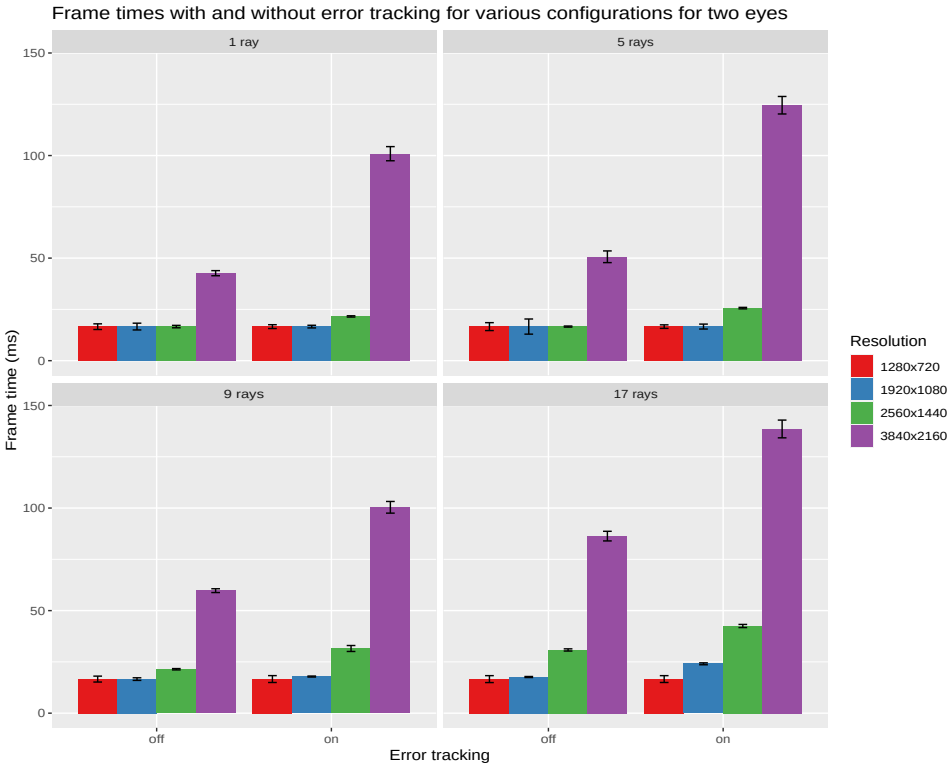
Activating error tracking creates costs in two places: Calculating the variances and covariances for all samples, which is directly linked to the number of rays and hence amount of samples. The other one is writing the absolute error and uncertainty values to textures so that following shaders can make use of them and this reading process. Whilst writing and reading the texture only depends on the size of that texture, the variance calculation depends on the size of the texture and the amount of rays. We therefore see the difference between rendering times for active and inactive error tracking to increase only slowly with increasing resolution.

Figure 5.17b shows the frame times for the same scene as used for figure 5.17a but with binocular vision. Note that the scale has been adjusted and we had to remove the simulation with $5120x2880$ pixels due to hardware constraints. When the second eye is activated, allowing for binocular simulation, we do not only require the simulator to render twice as many pixels, but execute two independent flows. These flows have independent configurations and textures. A final node, having access to the output textures of both flows, requires both of them to be rendered to completion and copy the results in an additional step. This imposes a large performance penalty. If used with a head-mounted display, one might be able to remove this last combination node and thereby greatly increase the simulator performance.

Regarding these performance characteristics, we conclude that the simulator may be used with 17 rays with resolutions up to 3840x2160 pixels, with and without error tracking in a desktop application in a monocular configuration. However, if binocular vision has to be simulated, the simulator becomes hard to use in this configuration. If used in a head-mounted display, further improvements are needed.



(a) Frame times for the classroom scene with one eye



(b) Frame times for the classroom scene with two eyes

Figure 5.17: Frame times for the classroom scene with different configurations

6 Conclusion

In this work, we developed a metric to quantify and visualize differences in perception caused by vision deficiencies. We extended the VSS, a simulator for visual deficiencies, to be able to visualize this metric based on absolute error and uncertainty tracked throughout the simulation. The simulator was extended to include two more vision deficiencies: astigmatism and strabism. With the addition of a ganglion cell simulation, we added a first stage of filtering and processing to the simulator. Using different view modes, we enabled users to view the output of the VSS in many configurations, like overlays of our metric on top of the simulated image. We implemented a simple attention guidance system to attract the users attention to a specific location.

We found that our error metric can be used to quickly highlight errors introduced by vision deficiencies. Using different combination functions, one can select the visualization that matches the current requirements best. Using propagation of uncertainty and a custom metric for combination of existing and new uncertainty, the simulator can cope with configurations in which uncertainty is introduced in several places and needs to be propagated troughout multiple nodes.

By comparing the metric to SSIM, we found that the complex and expensive error propagation has some benefits in certain use-cases. These are mainly the insensitivity to changes in position, made possible by analyzing uncertainty independent from error and the ability to choose the dimensions of RGBXY that are relevant for the current analysis.

By using the visualization of the retinal ganglion cells on natural and artificial scenes, we produced activation patterns similar to those of Aleman et al. [AWS18] but found that our implementation is susceptible to artifacts related to the used grid.

Although the used RGB color space does not portray human perception very well, we have shown that the uncertainty of RGBXY allows to highlight problematic areas in a scene. If a model closer to human percecption is needed, one can replace the usage of RGB by a color distance metric like Δe_{2000}. By tracking RGBXY throughout the whole simulation, allowing processing of the error und uncertainty vectors in a final node, more advanced analysis in external tools is possible. We want to end our conclusion with the observation that the architecture of the VSS allowed for mostly unproblematic extension of the application. We therefore hope for broader adaption of the simulator in research and education.

6.1 Outlook

Based on the insights gained in our work, we identify the following areas as possible topics for future work.

VSS for Virtual Reality Using head-mounted displays, optionally with see-through capabilities, the effects of simulated eye diseases may easier to understand. We expect the usage of virtual reality to make it easier to raise awareness for inclusive design and communicate problems encountered by people with eye diseases. Since we implemented an overlay view mode, we hope that a port of the VSS allows for analysis of a scene without breaking immersion. Unfortunately, we expect the performance of the simulator to be too low to support error tracking, convincing simulations and native binocular resolution for modern head-mounted displays without performance improvements.

Ganglion Cells and Myopia Depending on developments in the field of Aleman et al. [AWS18], regarding a possible link between contrast polarity and the development of myopia, our simulation of RGC might be a useful tool in further research. In combination with a head-mounted display with see-through capabilities, exploration of everyday scenarios with regard to contrast polarity might be simplified with the RGC.

Strabismus and Virtual Reality Although we implemented a simulation for strabismus, it remains to analyze tolerance of the simulation for wearers of virtual reality devices. Since we simulate similar reduced fusion capabilities as observed with real strabismus, we expect similar symptoms when using the simulator. We have decided agains implementing phorias, since they have to be simulated as involuntary eye movements. We suspect, since virtual reality devices are known to be prone to cause simulator sickness, phorias to cause at least nausea in the wearer. It might be interesting to test this with people wearing head-mounted displays.

Combination of RGC Simulation and Error Metric It might be of interest to combine the RGC simulation with the error metric. This is motivated by the reduced sensitivity of the human eye with increasing eccentricity. We therefore assume, weighting down the error metric depending on the position might be beneficial. Since this requires well-funded assumptions on how the perceived severity of errors follows the ganglion cell density, we leave this to be explored in future work.

Bibliography

[AWS18] A. C. Aleman, M. Wang, F. Schaeffel. "Reading and Myopia: Contrast Polarity Matters". In: *Scientific Reports* 8.1 (July 2018). DOI: 10.1038/s41598-018-28904-x (cit. on pp. 11, 17, 37, 38, 41, 43, 56, 75, 76).

[Bar93] R. J. B. Barlow. *A Guide to the Use of Statistical Methods in the Physical Sciences*. John Wiley and Sons, Nov. 30, 1993. 228 pp. ISBN: 0471922951. URL: https://www.ebook.de/de/product/3055878/barlow_roger_j_barlow_statistics.html (cit. on p. 29).

[Bar99] P. Barten. *Contrast sensitivity of the human eye and its effects on image quality*. Bellingham, WA: SPIE Optical Engineering Press, 1999. ISBN: 0819434965 (cit. on pp. 15, 24).

[Bha09] B. Bhattacharyya. *Textbook of Visual Science and Clinical Optometry*. Jaypee Brothers Medical Publishers, Aug. 1, 2009. 316 pp. ISBN: 8184485999. URL: https://www.ebook.de/de/product/16545311/bikas_bhattacharyya_textbook_of_visual_science_and_clinical_optometry.html (cit. on pp. 13, 17).

[Blo94] S. A. Bloomfield. "Orientation-sensitive amacrine and ganglion cells in the rabbit retina". In: *Journal of Neurophysiology* 71.5 (May 1994), pp. 1672–1691. DOI: 10.1152/jn.1994.71.5.1672 (cit. on p. 38).

[BS09] V. Balasubramanian, P. Sterling. "Receptive fields and functional architecture in the retina". In: *The Journal of Physiology* 587.12 (June 2009), pp. 2753–2767. DOI: 10.1113/jphysiol.2009.170704 (cit. on p. 38).

[BSK14] V. Bhateja, A. Srivastava, A. Kalsi. "Fast SSIM Index for Color Images Employing Reduced-Reference Evaluation". In: *Proceedings of the International Conference on Frontiers of Intelligent Computing: Theory and Applications (FICTA) 2013*. Springer International Publishing, 2014, pp. 451–458. DOI: 10.1007/978-3-319-02931-3_51 (cit. on p. 70).

[CC03] S. G. Crewther, D. P. Crewther. "Inhibition of retinal ON/OFF systems differentially affects refractive compensation to defocus". In: *NeuroReport* 14.9 (July 2003), pp. 1233–1237. DOI: 10.1097/00001756-200307010-00009 (cit. on p. 17).

[CGC14] K. Cornish, J. Goodman-Deane, P. J. Clarkson. "Designer Requirements for Visual Capability Loss Simulator Tools: Differences between Design Disciplines". In: *Universal Access in Human-Computer Interaction. Design and Development Methods for Universal Access*. Springer International Publishing, 2014, pp. 19–30. DOI: 10.1007/978-3-319-07437-5_3 (cit. on p. 11).

[CSKH90] C. A. Curcio, K. R. Sloan, R. E. Kalina, A. E. Hendrickson. "Human photoreceptor topography". In: *The Journal of Comparative Neurology* 292.4 (Feb. 1990), pp. 497–523. DOI: 10.1002/cne.902920402 (cit. on p. 41).

[Dac93] D. Dacey. “The mosaic of midget ganglion cells in the human retina”. In: *The Journal of Neuroscience* 13.12 (Dec. 1993), pp. 5334–5355. DOI: 10.1523/jneurosci.13-12-05334.1993 (cit. on pp. 38, 39).

[Dal92] S. J. Daly. “Visible differences predictor: an algorithm for the assessment of image fidelity”. In: *Human Vision, Visual Processing, and Digital Display III*. Ed. by B. E. Rogowitz. SPIE, Aug. 1992. DOI: 10.1117/12.135952 (cit. on pp. 23, 24).

[DLHT14] C. Dong, C. C. Loy, K. He, X. Tang. “Learning a Deep Convolutional Network for Image Super-Resolution”. In: *Computer Vision – ECCV 2014*. Springer International Publishing, 2014, pp. 184–199. DOI: 10.1007/978-3-319-10593-2_13 (cit. on p. 23).

[DM14] A. Denniston, P. Murray, eds. *Oxford Handbook of Ophthalmology*. Oxford University Press, Oct. 2014. DOI: 10.1093/med/9780199679980.001.0001 (cit. on pp. 17, 18).

[DMKC07] N. Drasdo, C. L. Millican, C. R. Katholi, C. A. Curcio. “The length of Henle fibers in the human retina and a model of ganglion receptive field density in the visual field”. In: *Vision Research* 47.22 (Oct. 2007), pp. 2901–2911. DOI: 10.1016/j.visres.2007.01.007 (cit. on pp. 38, 39).

[GBB+89] C. C. Gordon, C. L. Blackwell, B. Bradtmiller, J. L. Parham, P. Barrientos, S. P. Paquette, B. Corner, J. Carson, J. C. Venezia, B. M. Rockwell, M. Mucher, S. Kristensen. “2012 Anthropometric Survey of U.S. Army Personnel: Methods and Summary Statistics”. In: 1989 (cit. on p. 45).

[GL99] S.-S. Guan, M. R. Luo. “A colour-difference formula for assessing large colour differences”. In: *Color Research & Application* 24.5 (1999), pp. 344–355. DOI: https://doi.org/10.1002/(SICI)1520-6378(199910)24:5<344::AID-COL6>3.0.CO;2-X (cit. on p. 62).

[Goe93] H. Goersch. *Handbuch für Augenoptik*. 3. Aufl., 3. Nachdr. Carl Zeiss, 1993 (cit. on pp. 17, 35, 37, 55).

[Har03] P. T. Harvey. “Common Eye Diseases of Elderly People: Identifying and Treating Causes of Vision Loss”. In: *Gerontology* 49.1 (2003), pp. 1–11. DOI: 10.1159/000066507 (cit. on p. 11).

[HFW+16] B. A. Holden, T. R. Fricke, D. A. Wilson, M. Jong, K. S. Naidoo, P. Sankaridurg, T. Y. Wong, T. J. Naduvilath, S. Resnikoff. “Global Prevalence of Myopia and High Myopia and Temporal Trends from 2000 through 2050”. In: *Ophthalmology* 123.5 (May 2016), pp. 1036–1042. DOI: 10.1016/j.ophtha.2016.01.006 (cit. on p. 11).

[HLMM09] H. Hoshi, W.-L. Liu, S. C. Massey, S. L. Mills. “ON Inputs to the OFF Layer: Bipolar Cells That Break the Stratification Rules of the Retina”. In: *Journal of Neuroscience* 29.28 (July 2009), pp. 8875–8883. DOI: 10.1523/jneurosci.0912-09.2009 (cit. on p. 16).

[Kan13] E. Kandel. *Principles of neural science*. New York: McGraw-Hill, 2013. ISBN: 9780071390118 (cit. on pp. 15, 16).

[KD91] H. Kolb, L. Dekorver. “Midget ganglion cells of the parafovea of the human retina: A Study by electron microscopy and serial section reconstructions”. In: *The Journal of Comparative Neurology* 303.4 (Jan. 1991), pp. 617–636. DOI: 10.1002/cne.903030408 (cit. on pp. 38, 39).

[KG95] R. L. Kline, E. P. Glinert. “Improving GUI accessibility for people with low vision”. In: *Proceedings of the SIGCHI conference on Human factors in computing systems - CHI ’95*. ACM Press, 1995. DOI: 10.1145/223904.223919 (cit. on p. 11).

[LSGB20] D. Lange, T. C. Stratmann, U. Gruenefeld, S. Boll. “HiveFive: Immersion Preserving Attention Guidance in Virtual Reality”. In: *Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems*. ACM, Apr. 2020. DOI: 10.1145/3313831.3376803 (cit. on p. 48).

[MMS] R. Mantiuk, K. Myszkowski, H.-P. Seidel. “Visible difference predicator for high dynamic range images”. In: *2004 IEEE International Conference on Systems, Man and Cybernetics (IEEE Cat. No.04CH37583)*. IEEE. DOI: 10.1109/icsmc.2004.1400750 (cit. on p. 24).

[MOR01] N. MORLET. “Astigmatism and the analysis of its surgical correction”. In: *British Journal of Ophthalmology* 85.9 (Sept. 2001), pp. 1127–1138. DOI: 10.1136/bjo.85.9.1127 (cit. on p. 17).

[MWBR13] K. Marwah, G. Wetzstein, Y. Bando, R. Raskar. “Compressive Light Field Photography using Overcomplete Dictionaries and Optimized Projections”. In: *ACM Trans. Graph. (Proc. SIGGRAPH)* 32.4 (2013), pp. 1–11 (cit. on p. 45).

[NJC96] G. K. V. Noorden, R. H. JENKINS, M. W. CHU. “Horizontal Transposition of the Vertical Rectus Muscles for Cyclotropia”. In: *American Journal of Ophthalmology* 122.3 (Sept. 1996), pp. 325–330. DOI: 10.1016/s0002-9394(14)72058-6 (cit. on p. 46).

[PA97] M. R. Pointer, G. G. Attridge. “Some aspects of the visual scaling of large colour differences”. In: *Color Research & Application* 22.5 (1997), pp. 298–307. DOI: https://doi.org/10.1002/(SICI)1520-6378(199710)22:5<298::AID-COL3>3.0.CO;2-S (cit. on p. 62).

[PRS11] C.-W. Pan, D. Ramamurthy, S.-M. Saw. “Worldwide prevalence and risk factors for myopia”. In: *Ophthalmic and Physiological Optics* 32.1 (Dec. 2011), pp. 3–16. DOI: 10.1111/j.1475-1313.2011.00884.x (cit. on p. 11).

[Rem12] L. Remington. *Clinical anatomy and physiology of the visual system*. St. Louis: Elsevier/Butterworth-Heinemann, 2012. ISBN: 9781437719260 (cit. on pp. 16, 17).

[Ren12] A. Rencher. *Methods of multivariate analysis*. Hoboken, New Jersey: Wiley, 2012. ISBN: 9781118391655 (cit. on p. 30).

[RRKS19] H. Rebecq, R. Ranftl, V. Koltun, D. Scaramuzza. “Events-to-video: Bringing modern computer vision to event cameras”. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2019, pp. 3857–3866 (cit. on p. 23).

[SKC+19] A. Serrano, I. Kim, Z. Chen, S. DiVerdi, D. Gutierrez, A. Hertzmann, B. Masiá. “Motion parallax for 360° RGBD video”. In: *IEEE Transactions on Visualization and Computer Graphics* 25 (2019), pp. 1817–1827 (cit. on p. 45).

[SM15] J. R. Sanes, R. H. Masland. “The Types of Retinal Ganglion Cells: Current Status and Implications for Neuronal Classification”. In: *Annual Review of Neuroscience* 38.1 (July 2015), pp. 221–246. DOI: 10.1146/annurev-neuro-071714-034120 (cit. on p. 15).

[SR11] G. Schwartz, F. Rieke. “Nonlinear spatial encoding by retinal ganglion cells: when 1 + 1 ≠ 2”. In: *Journal of General Physiology* 138.3 (Aug. 2011), pp. 283–290. DOI: 10.1085/jgp.201110629 (cit. on p. 43).

[SRA+19] C. Schulz, N. Rodrigues, M. Amann, D. Baumgartner, A. Mielke, C. Baumann, M. Sedlmair, D. Weiskopf. “A Framework for Pervasive Visual Deficiency Simulation”. In: *2019 IEEE Conference on Virtual Reality and 3D User Interfaces (VR)*. IEEE, Mar. 2019. DOI: 10.1109/vr44988.2019.9044164 (cit. on p. 19).

[SSM86] P. H. Schiller, J. H. Sandell, J. H. R. Maunsell. “Functions of the ON and OFF channels of the visual system”. In: *Nature* 322.6082 (Aug. 1986), pp. 824–825. DOI: 10.1038/322824a0 (cit. on p. 16).

[Sta08] I. O. for Standardization. *Guide to the expression of uncertainty in measurement (GUM)-Supplement 1: Numerical methods for the propagation of distributions*. Geneva: International Organization for Standardization, 2008 (cit. on pp. 24, 29).

[SV79] R. M. Shapley, J. D. Victor. “Nonlinear spatial summation and the contrast gain control of cat retinal ganglion cells.” In: *The Journal of Physiology* 290.2 (May 1979), pp. 141–161. DOI: 10.1113/jphysiol.1979.sp012765 (cit. on p. 43).

[SWD04] G. Sharma, W. Wu, E. N. Dalal. “The CIEDE2000 color-difference formula: Implementation notes, supplementary test data, and mathematical observations”. In: *Color Research & Application* 30.1 (2004), pp. 21–30. DOI: 10.1002/col.20070 (cit. on p. 62).

[SZA16] S. Szpiro, Y. Zhao, S. Azenkot. “Finding a store, searching for a product”. In: *Proceedings of the 2016 ACM International Joint Conference on Pervasive and Ubiquitous Computing*. ACM, Sept. 2016. DOI: 10.1145/2971648.2971723 (cit. on p. 11).

[Vit09] S. Vitale. “Increased Prevalence of Myopia in the United States Between 1971-1972 and 1999-2004”. In: *Archives of Ophthalmology* 127.12 (Dec. 2009), p. 1632. DOI: 10.1001/archophthalmol.2009.303 (cit. on p. 11).

[VT16] S. Venkataramani, W. R. Taylor. “Synaptic Mechanisms Generating Orientation Selectivity in the ON Pathway of the Rabbit Retina”. In: *Journal of Neuroscience* 36.11 (Mar. 2016), pp. 3336–3349. DOI: 10.1523/jneurosci.1432-15.2016 (cit. on p. 38).

[Wäs04] H. Wässle. “Parallel processing in the mammalian retina”. In: *Nature Reviews Neuroscience* 5.10 (Oct. 2004), pp. 747–757. DOI: 10.1038/nrn1497 (cit. on p. 16).

[Wat14] A. B. Watson. “A formula for human retinal ganglion cell receptive field density as a function of visual field location”. In: *Journal of Vision* 14.7 (June 2014), p. 15. DOI: 10.1167/14.7.15 (cit. on pp. 15, 39–43).

[WB09] Z. Wang, A. Bovik. “Mean squared error: Love it or leave it? A new look at Signal Fidelity Measures”. In: *IEEE Signal Processing Magazine* 26.1 (Jan. 2009), pp. 98–117. DOI: 10.1109/msp.2008.930649 (cit. on p. 23).

[WBSS04] Z. Wang, A. Bovik, H. Sheikh, E. Simoncelli. “Image Quality Assessment: From Error Visibility to Structural Similarity”. In: *IEEE Transactions on Image Processing* 13.4 (Apr. 2004), pp. 600–612. DOI: 10.1109/tip.2003.819861 (cit. on pp. 23, 24).

[WD13] B. S. Wallang, S. Das. “Keratoglobus”. In: *Eye* 27.9 (June 2013), pp. 1004–1012. DOI: 10.1038/eye.2013.130 (cit. on p. 36).

[WP17] P. Walter, N. Plange. *Basiswissen Augenheilkunde*. Springer Berlin Heidelberg, 2017. DOI: 10.1007/978-3-662-52801-3 (cit. on p. 18).

[WR05] H. R. Wu, K. R. Rao. *Digital Video Image Quality and Perceptual Coding (Signal Processing and Communications)*. USA: CRC Press, Inc., 2005. ISBN: 0824727770 (cit. on p. 23).

[WS18] S. Wienbar, G. W. Schwartz. “The dynamic receptive fields of retinal ganglion cells”. In: *Progress in Retinal and Eye Research* 67 (Nov. 2018), pp. 102–117. DOI: 10.1016/j.preteyeres.2018.06.003 (cit. on pp. 38, 39, 43).

[XYS01] H. Xu, H. Yaguchi, S. Shioiri. “Testing CIELAB-based color-difference formulae using large color differences”. In: *Optical Review* 8.6 (Nov. 2001), pp. 487–494. DOI: 10.1007/bf02931740 (cit. on p. 62).

All links were last followed on April 11, 2021.

Declaration

I hereby declare that the work presented in this thesis is entirely my own and that I did not use any other sources and references than the listed ones. I have marked all direct or indirect statements from other sources contained therein as quotations. Neither this work nor significant parts of it were part of another examination procedure. I have not published this work in whole or in part before. The electronic copy is consistent with all submitted copies.

---

place, date, signature