Towards real-time ray tracing through foveated rendering

Erik N. Molenaar
University of Utrecht
ICA-3985725

February 5, 2018

Abstract

Foveated rendering is a technique that leverages the rapidly declining perceived quality of human vision towards the periphery to speed up 3D rendering. In this thesis a method is proposed to apply foveated rendering to ray tracing in order to bring ray tracing to the realm of real-time interactive applications. The proposed method consists of sampling according to a perceptually based distribution and the reconstruction of the image through interpolation.

An experiment with 17 participants was performed in order to look for the optimal interpolation method and the experiment showed similar performance with each interpolation method, but revealed the foveation method's sensitivity to high contrast when using lower density distributions due to the precomputed nature of the distribution. Without changes to the method, such as generating a content-aware distribution each frame, this eliminates the possibility of using lower density distributions that would require fewer samples. However, even with the base density distribution the method still provides a base speedup of 4.3 times.

Acknowledgements

I would like to thank the people that helped me throughout the thesis project. For starters, I would like to thank my supervisor at Utrecht University, Jacco Bikker, for helping me set up this project, thinking along and his feedback. Additionally, I would like to thank Nina Rosa at Utrecht University for helping in the design of the experiment. I would also like to thank Fove and Remco Kuijper who worked at Fove during this project for the gaze-tracking HMD, help with the headset and a discussion about the project about what to look for and keep in mind. Thanks to Michael Stengel at the Technical Universtiy of Delft for a fruitful discussion about the state of the method and suggestions for improvement. Lastly, I would like to thank my family and friends for their continued support.

Contents

1 Introduction

The field of 3D rendering concerns itself with constructing a generally visually appealing 2D image from a 3D scene. The most prevalent uses of this are seen in the entertainment industry, in both films and video games. Of these two, the video games are more strictly bound in what is possible given the limits of hardware and the time in which the rendering has to be completed, since it is a real-time process that needs to yield an interactive result. The most widespread algorithm across the industry to accomplish this is the *rasterisation* algorithm, thanks to the impressive performance it achieves with commonly available dedicated hardware. Its alternative is the *ray tracing* algorithm, which is slower, but generally achieves better results by trivially leading to accurate global effects like shadows and reflection, whereas with rasterisation implementing these effects is both costly and solely an approximation of the actual effect [Bik12]. Then again, a drawback of ray tracing is its computational cost. Significant changes need to be brought about before it can become widely available with real-time performance.

This change might come in the form of a certain type of hardware that is rapidly growing in popularity and usage: *head-mounted displays* (*HMDs*), more commonly referred to as virtual reality headsets. These devices generally come in the form of goggles that the user puts on that then shows the image on screens magnified by lenses to cover most of the user's view, resulting in a very immersive experience. The newest development for these HMDs is *gaze-tracking*, where the device tracks the gaze direction of the user in order to utilise this in the application. There are KickStarter funding projects for such hardware [Kic15], big companies like Google investing in the technology [Bus16], established VR companies like Oculus seeing the potential and working on the technology as well [Upl15]. There are various uses for gaze tracking that can already be seen in a select number of existing video games, such as aiming by looking, moving the camera where you look, getting blinded by looking at the sun or making eye contact with in-game characters [Tra17]. Yet the most exciting use for gaze tracking is *foveated rendering*, also referred to as *gaze-contingent rendering*, differing the rendering quality across the image based on the user's gaze direction. The perceived quality of human vision decreases rapidly towards the periphery and this principle is exploited by foveated rendering by decreasing quality and, by extension, the work done to render the areas of the image that end up in the user's periphery to speed up the overall rendering process significantly.

If we combine the concept of foveated rendering and its potential to significantly speed up the rendering process with the fact that the ray tracing algorithm is preferable to the current dominant rendering algorithm, but is

held back by its performance, there is a definite match between the two. Should foveated rendering bring that needed development to achieve real-time performance with ray tracing, then that would spell out a giant leap in the field of 3D rendering and, as such, the video game industry as a whole.

1.1 Problem statement

This thesis sets out to explore the possibilities that foveated rendering provides to speed up the ray tracing rendering process. The primary aim of this thesis is to explore the relationship between speed and quality and to find the maximum achievable speed with correctly functioning foveation. Since the essence of foveated rendering is to trick the human vision into perceiving everything happening in the periphery without accurately portraying it, the threshold of achievable speed is set by the properties of the human visual system, as it is imperative that the user is not bothered by the foveation and it does not break the user's immersion.

These essential points lead to the formation of the following research question to be answered throughout this thesis:

> What is the achievable speedup for ray tracing when utilising gaze-tracking and foveation without users being able to notice any differences with a regular image?

The general form of the foveation proposed in this thesis is sampling the scene non-uniformly based on the properties of human vision and then reconstructing the image using these samples through an interpolation algorithm. Different ways or densities of sampling and different interpolation algorithms will lead to varying results and speeds. The following two sub-questions explore these results in order to best answer the overarching research question.

1. What are the effects of the different sampling and interpolation options on the performance of foveated rendering?
2. Which settings allow for the biggest decrease in sampling resolution while keeping the foveation unnoticeable to the user?

1.2 Thesis Structure

Section 1 provides an introduction to the issue explored in this thesis. Following that, Section 2 gives some background information on the different concepts involved in the implementation of foveated rendering for ray tracing. Next, Section 3 provides an overview of earlier research into related

subjects, such as foveated rendering for rasterisation as well as for ray tracing, the generation of a distribution to sample from and interpolation for scattered data as resulting from that distribution. Afterwards the method used for this thesis is explained in Section 4. Using this implementation, the resulting performance is examined in Section 5. Section 6 then deals with the experiment performed for this thesis to research the qualitative results of the settings in the method and to what extent these settings allow for a decrease of resolution in the periphery, which leads to an increase in performance. The findings of the performance review, the results of the experiment and the overall properties and assumptions in the method are then discussed in Section 7. Finally, the thesis is concluded in Section 8, which summarises the thesis and proposes future work.

2 Background

This section provides an overview with some required theoretical background on the different concepts related to the proposed method. It highlights the ray tracing rendering algorithm, provides a background on the workings of the human eye, touches upon the current state of gaze tracking hardware and issues it needs to deal with, conveys the concept of foveated rendering, explores temporal anti-aliasing and finishes with an overview of psychophysics.

2.1 Ray tracing

Rendering a three dimensional scene to a two dimensional image lies at the basis of a big part of the computer graphics field. There are two methods to do so that dominate the field: *rasterisation* and *ray tracing*. The principle behind ray tracing is to simulate light physics, using ray optics to traverse the scene and calculate the resulting colour for individual pixels. Computing the result of all the branching ray paths for more intricate materials, such as those containing reflection and refraction, makes the algorithm fairly computationally expensive. This has a big impact for its potential implementation in real-time applications, since the algorithm will need to output an image within a reasonable timeframe in order to meet the required framerate.

Rasterisation, on the other hand, is currently the most widely used rendering algorithm for real-time applications thanks to its speed and the availability of specialised hardware. The speed of the rasterisation algorithm stems from its nature of approximating the result throughout separate occlusion and lighting stages. Its available methods for the different global aspects of rendering, like shadows and reflections, are mere approximations of the actual global effects, rarely work well together and only do so in specific cases, and are relatively expensive to add to the rendering [Bik12]. In contrast, these effects follow trivially in ray tracing due to the algorithm's physically-based nature. This comes at a cost, however, since the ray tracing algorithm is significantly more expensive to fully compute, comparatively even more so due to the lack of specialised hardware for ray tracing. This generally makes ray tracing very suitable for pre-rendered, realistic imagery such as in movies, but not so much for real-time applications until significant speed-ups can be achieved.

2.2 The human eye

By leveraging the properties of human senses, algorithms can become more efficient without notable difference to the user, as seen for example in MPEG

compression of audio and video files [Bra99]. In the case of computer graphics it is possible to exploit properties of the human visual system to focus computational power on the areas that matter most. In order to accomplish this, however, it is imperative to first understand the workings of the human visual system.

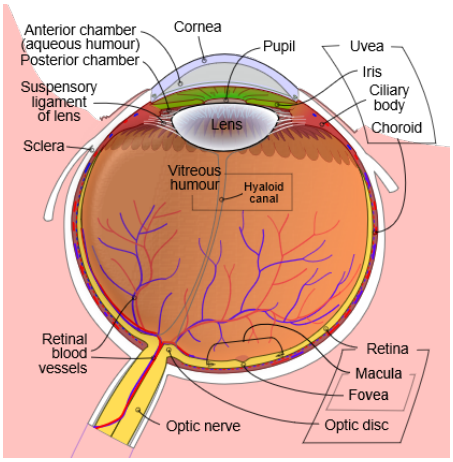

Figure 1: *An illustration of the human eye [Wik07]*

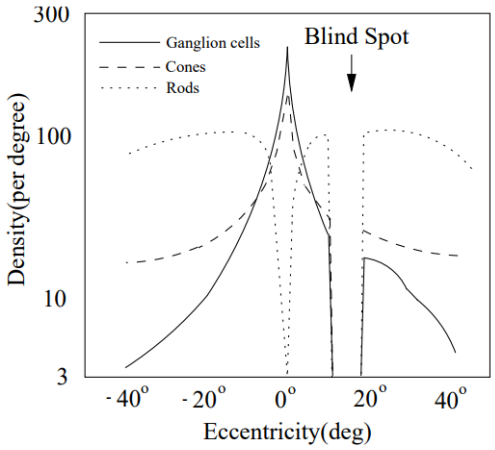

Figure 2: *The distribution of cells in the retina by eccentricity [SEB03]*

When looking out into the world, the optical image of that world is formed on the membrane lining the back of the eye, called the *retina*. This retina contains two types of photoreceptor cells, the *cones* and the *rods* that send their signals through *ganglion cells* that each receive signals from multiple cones and rods. The cones provide clear, colour vision, while the rods perceive less intense light, give us colourless vision and are primarily sensitive to changes in light. In the centre of the retina lies a little pit, about 1.5 millimeters in diameter, called the *fovea* [SEB03]. This fovea has the highest concentration of cones and is where the human vision provides the highest resolution [SEB03]. A full figure of the human eye can be seen in Figure 1, while Figure 2 shows the relative density of the cells across the retina. It is the varying densities shown here, the decrease in ganglion cells and cones, but an increase in rods towards the periphery, that make that peripheral vision relies mostly on the less accurate rods and the signals are aggregated by much fewer ganglion cells. This is the reason for the low visual acuity in human peripheral vision when compared to the fovea.

The human vision spans 160° horizontally and 135° vertically, but of this field there is only a circle of 5° with which we see fine detail [GFD+12]. This area is composed of the fovea, which makes up the centre circle of 2°, and the surrounding *parafovea*, which has a lower density of cones than the fovea, but

higher than the rest of the eye. Beyond that area, visual acuity diminishes quickly, meaning that the area with which we see fine detail is only about the size of a thumb nail at arm's length. Anything beyond the 30° circle, which is better known as *peripheral vision*, is mostly motion perception via the rods [Lin16], whose density decreases with distance from the fovea. This means that the distance from the centre of the eye, or the *visual axis*, which is measured in *eccentricity* or angle with the visual axis, has a direct impact on the detail of the perception. This effect is bigger than just the cell density suggests, since contrary to the cones in the fovea which have a one-on-one connection to ganglion cells, the rod cells farther from the fovea have a many-to-one relation, leading to aliasing [PSK+16]. Figure 3 displays a graph of human visual acuity relative to the maximum possible acuity per degree eccentricity. From this graph the rapid decline in visual acuity is clear, as the maximum is clearly at the fovea and at 10° from the fovea the acuity is down to 20 percent and 10 percent at 30°. After just $^1/_6$th of a degree the loss in acuity is already 25 percent [SAS16].

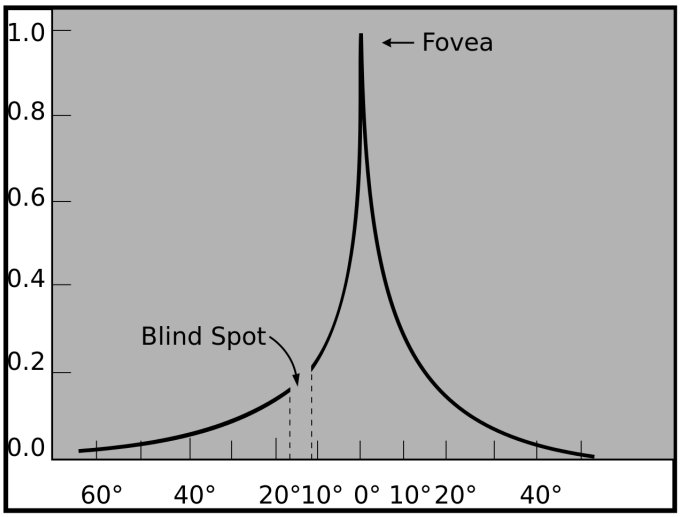

Figure 3: *Graph of relative visual acuity per degree eccentricity. The left side is towards the nose, the right side is towards the temple [Wik09].*

2.3 Gaze tracking

Since the area with the highest resolution in the user's perception depends on the gaze direction of the user, it is necessary to obtain that gaze direction. Technology limits the possibilities in this field, since performing this in a real-time application requires high framerate cameras with a resolution that

is high enough to identify and read the user's eyes. For a video game, the required framerate is 30 frames per second at the least, but is often set to 60 frames per second. Since received eye tracking results can only be used for the frame after the results have been processed, the refresh rate of the camera should be higher than the framerate of the application to limit latency. Another reason the refresh rate has to be high is human *saccade*, which is the most common, rapid movement of the eyes to focus on a new location, which the camera will have to be able to keep up with and detect. Consumer cameras for computers, such as webcams, do not offer this kind of performance yet.

A more feasible method for now would be to use *head-mounted displays*, or HMDs for short. These are commonly known as *virtual reality* (VR) headsets and generally provide, next to one or two actual displays and lenses, some hardware to track the orientation of the user's head. Eye tracking is not commonly part of these devices, although more and more companies are starting to pick up the trend, often looking at the potential for gaming, such as aiming or scanning by looking at an object or in-game characters responding to eye contact [FOV15]. The KickStarter project for the eye tracking HMD FOVE has amassed over $480,000 from backers to develop their hardware [Kic15]. Google bought the start-up company EyeFluence which specialises in eye tracking [Bus16] and Oculus acquired the eye-tracking startup The Eye Tribe [Tec16]. The founder of Oculus Rift called eye tracking a "critical part" of the future of VR and Oculus Rift is researching it themselves as well [Upl15], most likely leading to the later inclusion of the technology in their HMD. SensoMotoric Instruments provided extensions for a time to existing well-known HMDs, such as the Oculus Rift [Sen14], Samsung Gear VR [Sen16] and the HTC Vive [Upl16] to integrate their eye-tracking hardware into them.

The performance requirements increase with the decision to utilise head-mounted displays, however, since in order to prevent *cybersickness*, a form of motion sickness which is caused by a conflict between perception and the user's expectation in an immersive environment, the frame rate needs to be higher than on general monitors and 95 frames per second is the established target as that should eliminate noticeable flickering of the screen [Val14].

Even though peripheral vision is not as accurate as the central part of human vision, motion and flickering sensitivity are uniform across the entire visual field [PSK+16]. It is important then to avoid artifacts such as temporal aliasing that cause such phenomena. This is especially the case with head-mounted displays, which cover a large part of our visual field with a high field of view and a low pixel density [Val14].

2.4 Foveated rendering

Combining the availability of eye tracking and the knowledge of the properties of the human eye makes it possible to leverage visual acuity in rendering. This technique is called *foveated rendering*, but is sometimes also referred to as *gaze-directed* or *gaze-contingent rendering*. The principle of foveated rendering is to mimic the visual acuity graph around the user's fixation point, generally rendering an area around that point at full resolution rather than solely at that point. Beyond that, with increasing eccentricity, the rendering resolution diminishes to match the decrease in visual acuity. The distance from the fixation point and the corresponding lower visual acuity makes the lower quality imperceivable to the user so the resulting image is indistinguishable from a full resolution image. However, for rendering purposes being able to render big parts of the image at a lower resolution has the potential of increasing rendering speed significantly. Foveated rendering for rasterisation, for example, can reduce the number of shaded pixels by up to a factor 10 to 15 [GFD+12].

2.5 Temporal anti-aliasing

Sampling one frame of a continuous world with continuous motion can lead to aliasing and *temporal artifacts* such as fast, jerky motions and flickering. One way of eliminating these issues to create a *temporally stable* output, is *temporal anti-aliasing* or TAA for short [Kar14a, Kar14b]. The concept of temporal anti-aliasing is to generate the output from samples taken from multiple frames, not just the current one. This allows for an estimation over time rather than just capturing an instant.

Since locations of objects change over multiple frames, the first step is generally reprojection to find out where the object shown in a particular pixel was before and if it was even in view in the first place, so as to find out which pixel from the earlier frame, if any, contains the earlier colour data. This can be done through the same velocity buffer calculations used for motion blur to trace back the earlier position.

Simply rendering this different frame data together leads to *ghosting*, showing the objects in their previous positions as well, while only the current positions are interesting. This can be solved using *neighbourhood clamping*, in which the historical samples' usage is restricted to the neighbourhood of the current location, making the anti-aliasing a more local problem without large smearing effects.

2.6 Psychophysics

Measuring stimuli thresholds based on human senses requires some insight into the field of *Psychophysics*, the scientific study of the relation between stimulus and sensation, of which Gescheider provides a clear overview [Ges13]. These thresholds based on human senses are not set, but vary between subjects and even within subjects. Within this field are three classical methods to measure a threshold while eliminating this variability as much as possible. On top of that there are two types of thresholds that can be measured: *absolute thresholds* that represent when a stimulus is or is not perceived and *difference thresholds* which show when a difference between two stimuli can or cannot be perceived.

The first method is the *method of constant stimuli* in which a subject is shown each of the possible stimuli multiple times. Together, the results can be used to fit an s-shaped curve called an *ogive* to represent the psychometric function, as seen in Figure 4. This function can then be used to read the probability of a positive response at a certain stimulus intensity.

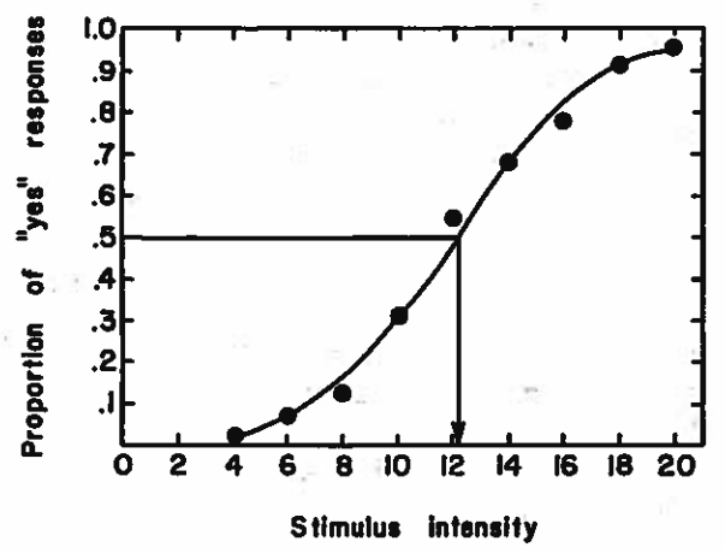

Figure 4: *An ogive mapping the relation of the stimulus intensity against the proportion of "yes" responses [Ges13].*

The next method is the most frequently used method thanks to its efficiency. This is the *method of limits*, in which the first test value is well above or below the threshold value and in each successive test the stimulus value moves towards the threshold in set steps. Once the subject switches answers from perceiving to no longer perceiving the stimulus, which is called a *re-*

versal, the test ends, although generally multiple such series are performed to approximate the threshold. One way to do so is the *staircase method* in which the test is not terminated, but is instead continued from the reversal point in the opposite direction until the subject switches answers again. The reversal points are noted and once enough values have been acquired, the mean of those values represents the 50% threshold, at which subjects will respond positively half the time and negatively the other half.

Although the method of limits is more efficient than the method of constant stimuli, it is also less accurate and provides less information. It can be used as preparation for the method of constant stimuli to make that method less costly. The method of limits is also sensitive to the human tendency to predict desired answers or repeating previous answers. The errors introduced by these tendencies are called the *error of expectation* and the *error of habituation* respectively, although there are ways to minimise these errors, such as varying the starting point and avoiding overly long trial series.

The last method is the *method of adjustment* in which the subject gains control over the presented stimulus and attempts to find the threshold themselves based on what they perceive. This is generally even more efficient than the method of limits, since trials can be quickly completed and it is not necessarily needed to complete any full series. The method of adjustment is limited in its possible applications however.

3 Previous work

This section contains an overview of previous work on different topics required for the proposed method and shows the groundwork out of which this method came forth. It deals with the work on foveated rendering for rasterisation, previous research into foveated rendering for ray tracing, methods for generating a distribution based on human vision and finally discusses several interpolation methods for scattered data.

3.1 Foveated rasterised rendering

With the emergence of the required technology to perform foveated rendering, research is widely being performed on the possibilities of foveated rendering and how to optimally implement it.

A method for foveated rendering for rasterisation was developed by Guenter et al. [GFD+12] using a desktop eye tracking setup. This method proposes rendering three distinct, rectangular layers with decreasing sampling rates. The rendered rectangles are then interpolate up to native resolution and smoothly composited with circular blending masks into a final image as seen in Figure 5. The layers are sized according to findings by Aubert and Foerster in 1857, who found that the minimum discernable angular size increases approximately linear to the eccentricity. The smallest layer is centred around the user's focus and is rendered at full quality with the highest level of detail (LOD), while the bigger layers behind that render a lower quality image. The outermost layer also uses a lower level of detail, using about half as many triangles as the other layers. Their relatively simple LOD approach saved only 0.5ms in their proposed method, but shows the feasibility of using a more profitable LOD system while hiding the LOD transitions from the user.

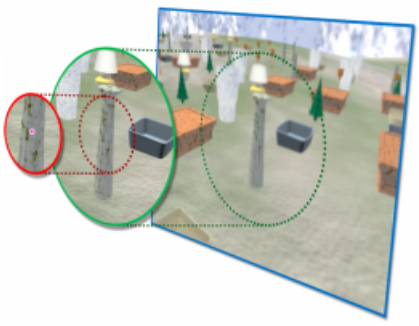

Figure 5: *The compositing of the three rendered layers into a final, foveated image [GFD+12].*

The smallest layer also refreshes at 120Hz, while the bigger layers alternatingly refresh at 60Hz. This is to be able to better account for saccades and to make the updating of the foveal region unnoticeable. A big factor in this is the system latency, the time between the capture of the eye gaze position and the rendered image being displayed on the screen. If that system latency is too big, the system lags behind and the user will be able to see the update of the foveal region of the image. All the systems in the method work asynchronously, making the exact latency hard to predict. Guenter et al. carefully chose their system to minimise that latency and made an analysis of the best and worst case latencies, which are 23ms and 40ms, respectively.

The lower sampling rate of the layers introduces distracting temporal artifacts that are clearly visible to the user. Simply brute-force supersampling the image back to native resolution reduces the artifacts, but greatly diminishes the performance gain of the method. Instead they combine three methods of low computational cost to decrease visible aliasing: *multi-sample antialiasing (MSAA)*, *temporal reprojection* and *whole frame jitter*.

The total cost reduction of this method compared to a full-resolution render amounts to 5 to 6 times. They estimate that as displays increase in size and field of view, the comparative benefit of foveated rendering compared to traditional rendering will become even greater, as traditional rendering cost grows exponentially and the cost increase for foveated rendering is roughly linear.

The contributions of this paper include the actual implementation of a foveated rendering system for general, interactive 3D graphics and showing the performance increase it can provide, their careful analysis of system latency in collaboration with eye tracking and that their foveated rendering system avoids distracting artifacts and will provide even greater benefits on larger, sharper displays.

A perceptually-based implementation of foveation in rasterisation for virtual reality is proposed by Patney et al. [PSK+16]. In their paper, they found that previously proposed foveated rendering techniques, such as the one from Guenter et al., were focused on reducing rendering cost with no regard for identifying and minimizing the resulting artifacts, which they did appear to suffer from. Instead, this paper works from the core question of what constitutes a good foveated image. They established this through a user test in which they gradually increased the aggressiveness of their Gaussian blur based on eccentricity for three different setups: aliased, temporally stable and contrast preserving. They found that filtering the outer regions of the image too aggressively induces a sense of tunnel vision, caused by the reduced contrast in the image, which is noticeable to the viewer. By adding a contrast

enhancement in post-processing, their viewers tolerated a two times larger blurring radius. User studies with this enhancement provided a base target image to use as a goal in the construction of a practical foveated renderer.

Basing their foveated renderer on human perception, the authors performed a literary study on human perception. This showed a number of essential points for application in a foveated rendering system. The first was the falloff in visual acuity with eccentricity. Another was *Cortical Magnification Theory*, which states that increased stimulus size at higher eccentricity provides a similar image. However, the existing foveated renderers based on this theory are lacking, since the theory does not cover all aspects of peripheral vision and is not applicable in all cases. Another finding was the existence of an *aliasing zone* in human peripheral vision, alluding to the mismatch between detection acuity and resolution acuity, the ability to detect a stimulus and the ability to discern its content, respectively. Detection acuity is significantly higher than resolution acuity in peripheral vision and filtering for foveated rendering can lead to loss of apparent contrast, which is detectable even if the object is not discernable. The authors intend to exploit this by enhancing contrast in peripheral regions in order to maintain apparent detail. The authors also found that motion detection is uniform across the visual field and that the same applies to flicker detection. Since the peripheral region is still quite sensitive to motion, foveated renderers need to be temporally stable or run conservatively in order to not break immersion. Lastly, tests with anisotropic blurring to exploit humans' unequal sensitivity between tangential and radial frequencies were inconclusive. Since specific hardware would be necessary to translate the results into computational savings, so were tests in exploitation of decreased colour perception in the periphery, despite promising perceptual results showing that subjects could not detect the color reduction.

The resulting method includes *temporal antialiasing* (TAA) to eliminate distracting aliasing artifacts that draw the users' attention. This makes the method temporally stable, although due to the nature of foveated rendering some more aspects need to be considered before it works as intended. The blurring in the periphery smears the same colours out over multiple pixels, reducing the information the surrounding pixels provide on what the colour should be. As such, the authors introduce *variance sampling* to exclude outliers and enabling TAA for subsampled areas such as the periphery in foveated rendering. Another aspect to consider is saccadic movement of the eye which can move areas out of focus into focus on the next frame. This will lead to a mismatch in the color data of the blurred outer region on the last frame and the detailed rendering of the current frame. The reuse of the old frame means that it might take several frames for the image to converge

to the detailed version. To remedy this *focus lag*, the authors change the convergence rate for such cases to have that area prefer the detailed image instead.

The user study between three rendering algorithms (no foveated rendering, the method from Guenter et al. augmented with the proposed TAA and their own proposed method) showed that they can render more coarsely up to 30° closer to the centre of interest than Guenter et al. without introducing gaze-dependent aliasing or blur. Through metrics, they also showed that their final result is highly similar to their perceptual target image. Among the key contributions of this paper is the revalidation of what constitutes a good foveated renderer through perceptual studies, specifically that temporal stability and contrast preservation are key requirements. They also provide a perceptually-validated target image to use as quality reference in development of a correct foveated renderer. Another contribution is their developed real-time gaze-tracked foveated rendering system with demonstrated performance and memory savings while closely resembling the target image quality and maintaining temporal stability. Lastly, they introduced improvements to temporal antialiasing for application in gaze-dependent, multi-resolution shading systems.

Figure 6: *A foveated image rendered with the proposed system by Patney et al. with the user's gaze directed at the clock in the upper-right corner [PSK+16].*

3.2 Foveated ray-traced rendering

While rasterisation is the industry standard, foveation is also beneficial to ray tracing.

Siekawa implemented a method for gaze-dependent ray tracing with the goal of increasing performance without perceptible quality loss [SM14]. In order to decrease the amount of rays cast, the author suggests to cast a number of rays proportional to the sensitivity of the human visual system. Based on the gaze-dependent contrast sensitivity function, they can create a distribution to sample with. Given that one degree of vision at a higher eccentricity covers more pixels than one degree at the centre of interest, but is not perceived as well, the author uses the same amount of rays per degree, automatically leading to a lower ray density in outer regions. The pixels in one perceptual degree are grouped together into a cell and each cell, smaller near the centre of interest and bigger near the periphery, receives the same amount of anti-aliasing rays. Although the effect of this will be less correct in the periphery, the viewer should not notice this as much as it is far from the gaze point.

Although this paper did not have a real-time setup to test with, their offline results allowed for quality tests. They rendered 1920 by 1080 pixel images using 32 samples for anti-aliasing to be viewed on a desktop setup. No user study was performed to verify the results. The author reports a more than 3 times speedup in rendering time while needing around 31 percent of the sampling rays. Based on the resolution that would be required to reach the maximum resolution perceivable by the human visual system, which according to the author is 5400 by 3900, the proposed method would lead to a 95 percent decrease in rays required to construct an image.

In a presentation, Harada suggested a method with a focus on performance for foveated ray tracing for VR using multiple GPUs [AMD14]. He discusses how the flexibility of ray tracing, specifically the ability to cast rays in any direction, is a great fit for VR, but the system is lacking in performance. The computational cost depends on the scene complexity and the number of rays cast. Whereas in a testing environment it is easy to reduce the scene complexity for performance sake, the scene complexity is outside of the algorithm's influence. Instead, the focus for a decrease in computational cost lies on the number of rays cast. Two methods are suggested: sample reuse between the right and left views and foveated rendering.

The sample reuse, which is more clearly explained in the related abstract [FH14], stems from their stereo rendering system seen in Figure 7. The followed principle is that if the scene scale is large enough, many of the

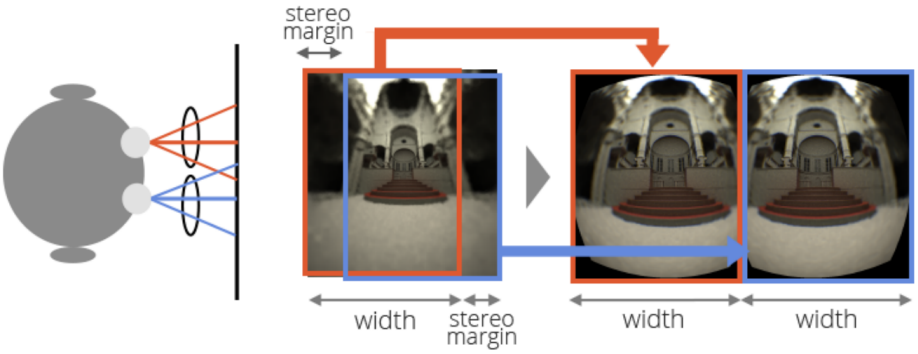

Figure 7: *The stereo rendering system that generates a right and left eye view from one central image [AMD14].*

pixels in the left and right view will be the same. Instead of calculating two views, the method uses one bigger image with *stereo margins* next to the part of the image shared by both views that will be used solely for the corresponding view. Although the presented method is an estimation and, as the authors say, still requires precise parallax correction, that will not significantly increase the number of required rays, thus maintaining the method's efficiency.

The suggested method starts by creating a density map containing cells that increase in size towards the periphery. Then one sample would be ray traced for each cell to decide the colour that cell will take. One sample consists of at least 16 rays to prevent distracting spatial and temporal luminance artifacts [FH14]. Only the cells that actually fall within the rendered area, which changes location on the density map depending on the gaze location of the user, will have their colour calculated. When all the cells in the rendered area have a colour, the remaining pixels have their colour estimated using a weighted blend of the k neighbour cells' colours. The author then performs *chromatic separation* and, in order to match VR screen output, he also applies *barrel distortion*.

The reported results are that the method uses only 5 percent of the samples required for full rate shading. However, the projected speedup of 10 to 30 frames per second is not enough for viable performance in VR. To fix this, the method uses multiple GPUs, splitting up the frame and dividing the workload of each frame between the GPUs instead of alternate frame rendering which would not decrease rendering time for any given frame. They report a framerate of more than 75 frames per second while preserving visual quality.

3.3 Generating a distribution

Sparse sampling is trivially possible with a ray tracer, since each primary ray can be individually and independently cast. This is not the case for rasterisation, since samples can only come from a regular raster. The ability to freely sample sparsely allows for sampling according to more accurate systems depending on eccentricity than those confined to rasters.

That leaves the choice of which system to choose to base the sampling on. Ideally, that should be a system that describes the human vision's capability to discern content at different eccentricity values. This function exists and is called the *contrast sensitivity function*. In order to translate this function to sample density on a 2D field, a sampling pattern is needed. A high quality sampling pattern is important for ray tracing, due to the visibility of the results and possible sampling errors, but also because it better represents the continuous function being sampled. Dunbar et al. [DH06] propose the blue-noise pattern of *Poisson-disc sampling*.

Poisson-disc sampling is a sampling pattern in which a sample is only used if there is no other sample in the results to which the distance is smaller than a given value. By making this distance dependent on the contrast sensitivity function, thus applying *adaptive sampling*, it is possible to sample according to that function while guaranteeing the desired density in all areas of the distribution.

The distributions used for sampling affect the final result and can cause some temporal effects in a real-time application, like persistently visible boundaries between samples, as a result of interpolation. Figure 8 shows a case in which this might occur. In this case it might be beneficial to use

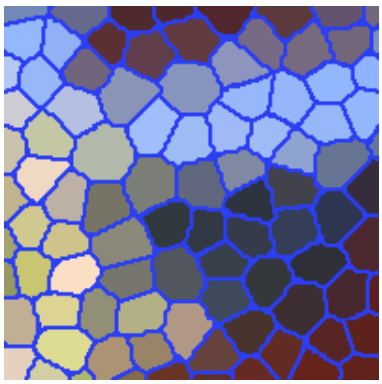

Figure 8: *A depiction of the boundaries (in dark blue) between the different cells. The cells change colour as the camera moves, but the separations remain in the same location, which can be noticeable to the user.*

a different distribution each frame, meaning the distribution would have to be regenerated each frame. Bridson proposed an extension to the method by Dunbar et al. to extend the method to arbitrary dimensions in *O(N)* time [Bri07]. In turn, Wei proposed a parallel implementation of the algorithm to speed it up even further [Wei08]. The reported results show that the method could generate up to 4.06 million samples per second in 2008. In the case of a real-time VR method at 95 frames per second, this means that per frame the method could generate about 42.7 thousand samples. The more recent Nvidia flagship GPU, the GeForce GTX 1080, is about 20 times faster than the GeForce 8800 GTX used in the paper by Wei, meaning the possible samples generated per frame today would be around 854.7 thousand samples with this method.

Assuming the resolution of the Oculus Rift and the HTC Vive, which is 1080 by 1200 for each eye [Dig16] or 1296 thousand pixels per eye, combined with the theoretical 30 percent of the samples needed as reported by Guenter et al. [GFD+12] and Patney et al. [PSK+16] (and by extension the 5 percent reported by Harada [AMD14]), means that the required 388.8 thousand samples per eye could theoretically be generated per frame.

Another method, however, would be to precompute a distribution or a number of distributions and reusing those in their original and transformed forms, such as flipped horizontally and vertically. This should seem random enough to the user since no distributions are repeated in subsequent frames. A consequence of this method is that the calculated distributions would need to be 4 times the screen size, assuming the centre of interest is in the middle of the distribution, in order to be able to position the distribution on the view regardless of the user's gaze direction. The increased size of the distribution is irrelevant for computational purposes, however, since in this case the distribution is precomputed.

3.4 Scattered interpolation

Many different 2D interpolation algorithms exist, but many assume an ordered, complete set of initial data, often laid out in a grid. To reconstruct a full image from the samples taken with a randomly generated distribution as described in Section 3.3, however, a *scattered interpolation* algorithm is needed. An added requirement to the algorithm is that it can be run efficiently in order to use it in a real-time application. Although it does not directly reveal feasibility in a real-time application, time complexity is a good indicator of the runtime speed of the algorithm and can at least be used as a guideline. Anjyo et al. provide an overview of scattered interpolation algorithms for computer graphics and their performance [ALP14]. The

algorithms addressed in this overview are claimed to have a runtime that is linear in the number of samples, sometimes with setup precomputation.

Different algorithms have different properties, such as containing visible peaks in either the points themselves or the borders between the points, depending on the basis for the interpolation. Such properties will have an effect on the resulting image, but to what extent that will be beneficial or harmful may vary. The complexity of interpolation algorithms also varies greatly and a higher complexity generally coincides with a higher quality.

A fairly naïve interpolation algorithm, *Nearest Neighbour*, offers minimal interpolation by making each pixel the colour of the closest known sample. An extension to it, *Natural Nearest Neighbour*, applies weighted interpolation based on volume. These algorithms both run in *O(N)* time, while Natural Nearest Neighbour offers limited smoothness in its interpolation and Nearest Neighbour is a class C^{-1}, discontinuous, method [BU06]. Figure 9 shows the results of these methods.

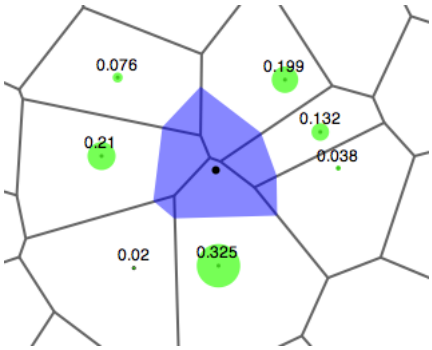

Figure 9: *The resulting cells from Nearest Neighbour in black and an example cell from Natural Nearest Neighbour in purple. [Wik10].*

Another weighted interpolation algorithm, which has often been reinvented in different forms and lies at the basis of many of the scattered interpolation algorithms, is *Shepard's interpolation*. This algorithm performs weighted interpolation based on distance to the known samples. It is far from an ideal interpolator in its original form, with a derivative of zero at the known data points and scaling quite poorly to large data sets and, as such, spawned many improvements. One such modification, *Modified Shepard's interpolation*, only considers samples that are close by for efficiency on large datasets and to reduce the local impact of distant samples. This method is continuously differentiable, or class C^1 smooth [MM05].

Another fairly simple interpolation is *linear interpolation* between points over, for example, triangles. This method requires constructing a triangula-

tion of the samples and applies interpolation within each triangle. It results in a less smooth interpolation of C^0 differentiability class.

A more complicated method, *Wiener interpolation*, known in machine learning as *Gaussian processes* and in geostatics as *Kriging*, is quite flexible in the smoothness of the result and the shape of the resulting function and, for discrete data, boils down to solving a matrix equation [ALP14]. It can use infinitely differentiable, class C^∞ functions to provide smooth interpolation [SCL13].

The most versatile and most commonly used scattered interpolation techniques are *Radial Basis Functions*. Within radial basis functions, each known data point has a set influence in a sphere around that point and the sums of these influences construct the values at the interpolated points. The *radial basis function kernel*, the function determining the influence of a point, can be freely substituted for a different effect, like increased smoothness, falling back down to zero after the last point or approximating through regularisation [ALP14]. The computation required involves solving a linear system of functions with a vector of weights as unknowns. These solutions generally have an $O(N^3)$ time complexity, but more efficient approaches to solve radial basis functions have been developed, which allow for precomputation of the most expensive calculations [ALP14]. Several of the popular Radial Basis Function kernels are class C^∞ smooth [FF05].

4 Method

The following section describes the proposed method as it has been developed based on the previous work discussed in Section 3, starting with the efficient generation of a distribution of points based on human perception. Next, the sampling of the scene based on these random points is discussed, after which a description of the chosen scattered interpolation methods is provided.

4.1 Point distribution

The *contrast sensitivity function* describes the perceptive ability of the human eye which rapidly decreases with increased eccentricity. To aptly capture the varying perception in sampling and rendering, we propose the use of a point distribution, where the density of the points at each location is based on the contrast sensitivity function and the user's gaze direction. This proposed method utilises precomputed *Poisson-disc sampling* sets. The point set is distributed over a canvas with a size of $2 * \text{screen width}$ by $2 * \text{screen height}$ with the supposed centre of interest in the middle. This is done so that the resulting point distribution can be shifted across the screen according to the user's gaze direction at execution, while maintaining coverage across the entire screen, as portrayed in Figure 10.

Since the point set is precomputed to save time during execution, there is no need for a real-time algorithm to create the point set. Instead, dart throwing is used for the Poisson-disc sampling, initially throwing darts at twice the screen size around the centre, followed by a series of darts thrown at 0.8 times the screen size around the centre to ensure sufficient coverage over the denser central area.

The *Poisson-disc* dart throwing is based on the *contrast sensitivity function* proposed by Wang et al. [WBLK01]. With the eventual user's centre of interest set in the middle of the canvas at the point *(screen width, screen height)*, the darts thrown have their threshold values set dependent on the dart's distance from the middle of the canvas. In other words, the value depends on the contrast sensitivity for the eccentricity at the location of the dart. As laid out in the paper by Wang et al., the eccentricity can be computed as:

$$e(v, x) = \tan^{-1}\left(\frac{d(x)}{Nv}\right) \tag{1}$$

where e is the eccentricity, $d(x)$ is the distance in pixels between the middle of the canvas and the dart's position with $d(x) = \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2}$, the parameter N is the width in pixels of the display the image is perceived on and v is the user's viewing distance measured in display widths. In order to obtain

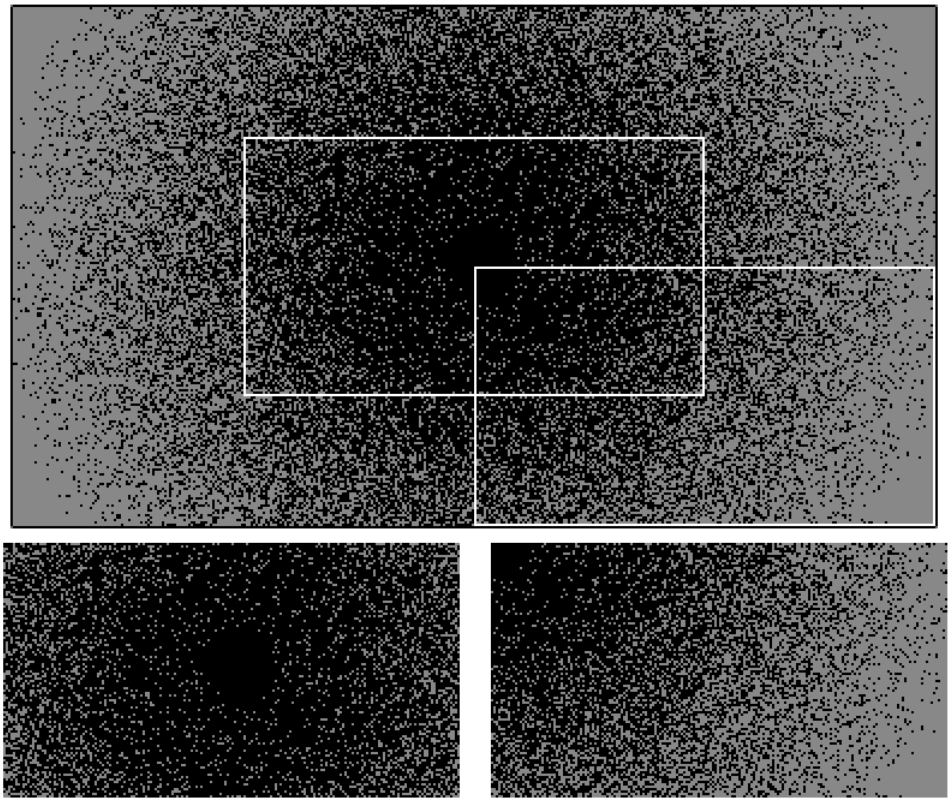

Figure 10: *An illustration of the point set as it shifts across the screen. The bottom-left and bottom-right images portray the used part of the point set when the user looks at the centre of the screen and the top-left of the screen respectively.*

differently spaced point sets containing fewer points, this method scales the resulting eccentricity by a set scaling value. The resulting eccentricity value can then be used in the following function to obtain the cut-off frequency f_c, the maximum perceivable frequency at the given eccentricity in cycles per degree:

$$f_c = \frac{e_2 \ln(1/CT_0)}{(e + e_2)\alpha} \tag{2}$$

The parameters CT_0, α and e_2 are model parameters and are set by Wang et al. based on the best fit reported in another paper by Geisler et al. [GP98] to $CT_0 = 1/64$, $\alpha = 0.106$ and $e_2 = 2.3$, which are also the values used in this method. The cut-off frequency represents how much human vision can perceive at certain eccentricities, but on the other hand, the perceivable frequency is also limited by the display the image is presented on. To calcu-

late the maximum display frequency, the highest frequency the display can show without aliasing, which is the display's Nyquist frequency or half of the display resolution in pixels per degree, resulting in the following function:

$$f_d = \frac{\pi N v}{360} \text{ (cycles/degree)} \tag{3}$$

In practice, this means that the cut-off frequency in the area direcly around the user's centre of interest is capped at the maximum display frequency and starts to decrease beyond that point, as shown in Figure 11. Since the point set that will be used to sample and reconstruct the final image is pre-computed, it has to contain enough information to be able to represent a worst-case scenario. In other words, the density of the points will be dictated by the cut-off frequency at each point. The highest values for the cut-off frequency are in the centre, where the distance between points is supposed to be lowest. To fix this, the following function is used:

$$p = \frac{f_d}{f_{c0}} - (1 - a) \tag{4}$$

where a is the area of one pixel, calculated with $\sqrt{1/\pi}$ and f_{c0} is the result of $min(f_d, f_c)$. By using p as the distance value of the points in the Poisson-disc sampling, this function makes sure the point density around the centre

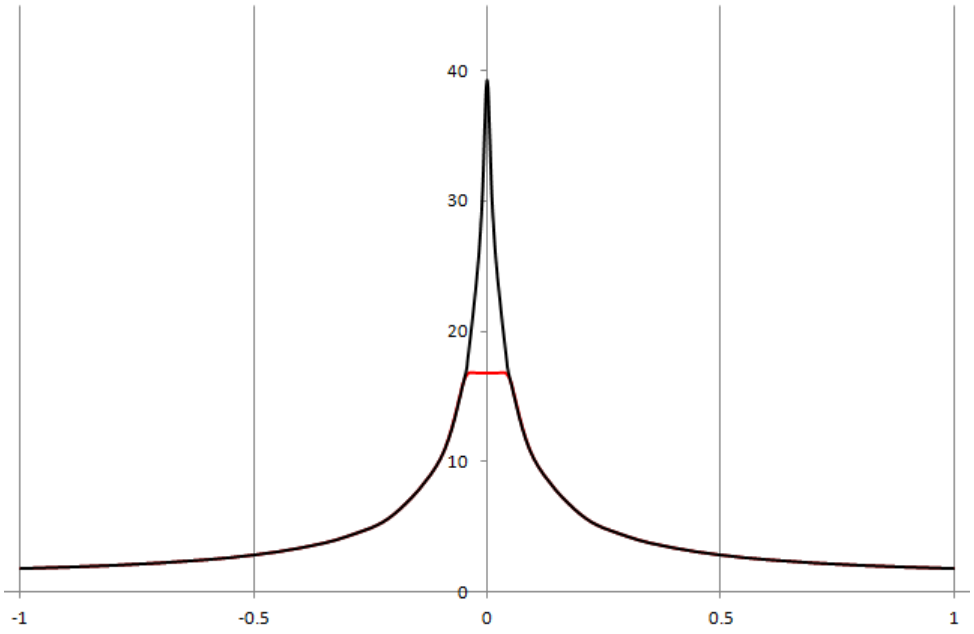

Figure 11: *Graph of the resulting cut-off frequency f_d with the eccentricity on the x-axis, where the maximum display frequency f_c is shown in red.*

of attention is not higher than one point per pixel and increases the spacing of points farther away from the centre.

During the dart throwing procedure, the darts are stored in a grid system to speed up the process and are finally stored ordered by *Morton index* for better memory access in the final application. In addition to this, another array is set up to store the indices in the aforementioned array of the k nearest neighbours for every pixel within the quadruple screen size plane to save the cost of looking up the nearest neighbours per pixel during execution. The application used for this method calculates and stores the 20 nearest neighbours, but varies in the number of used k neighbours during execution.

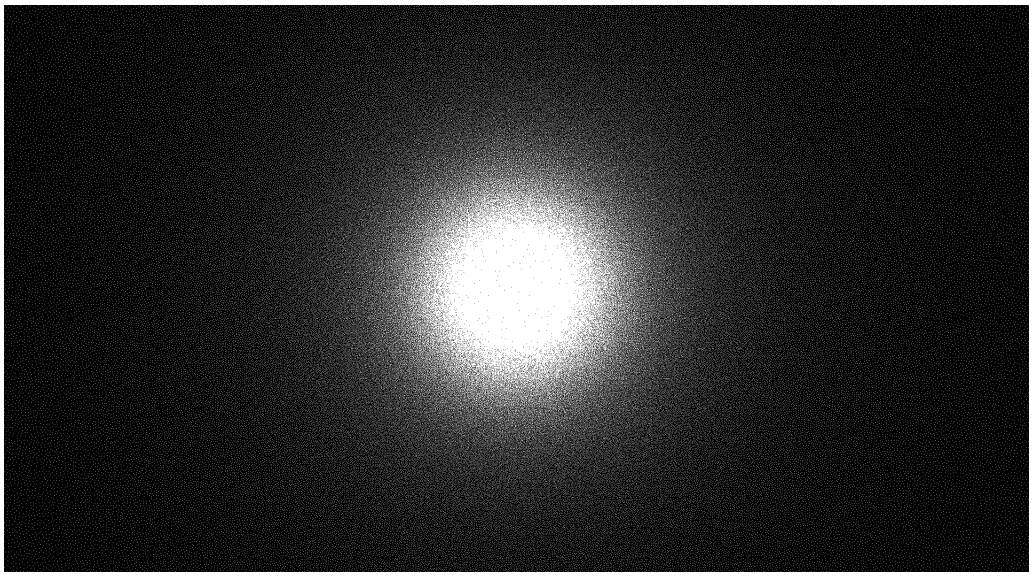

Figure 12: *A render of each point in the resulting distribution for a resolution of 1920 by 1080 and no scaling.*

4.2 Scattered sampling

Sparse sampling is trivially possible with ray tracing since rays can be independently cast at different locations in different directions. For the purposes of this method, these rays will be cast for each point generated in Section 4.1. The point distribution has to be offset by the gaze tracking location obtained by the gaze tracking hardware to centre the distribution around that location. Then each point within the boundaries of the screen can be sampled for. Points that fall within 10 pixels outside the boundaries of the screen are also included to prevent the interpolation algorithms using neighbouring points that have not been sampled in edge cases.

Each point stores its Poisson-disc distance and since the Poisson-disc distance is derived from the contrast sensitivity at that point, it signifies an aliasing zone, where the contents of the circle cannot be distinguished. Within this circle, whose radius is set to be the Poisson-disc distance of the point, several rays are cast through random locations. These random locations the rays are cast through are used for anti-aliasing purposes across the area of each point. The random numbers are generated with the same seed each frame to ensure the same locations are sampled between frames to prevent temporal artifacts. The results are of all rays are accumulated per point and stored to later reconstruct a final image from.

4.3 Scattered interpolation

In order to construct a full image from the scattered samples, an interpolation algorithm needs to be employed to fill the blank spaces. For the sake of this method and to get a comparative overview of scattered interpolation for the purposes of foveated rendering, three methods are chosen to represent different ends of the complexity spectrum. This spectrum consists of a trade-off between quality and computational cost. The first method is *Nearest Neighbour* interpolation, a very simple, efficient interpolation method with low cost, but also low quality interpolation, being a discontinuous function. To represent the other end of the spectrum, with high cost but also high quality interpolation, we include a class C^{∞} interpolation method, namely the *Radial Basis Functions* method. As a middle ground, the last included interpolation algorithm is the class C^{1} smooth *Modified Shepard's Method*. An example of the resulting images can be found in Figure 13.

Figure 13: *The resulting interpolation at high eccentricity for Nearest Neighbour (NN), the Modified Shepard's Method (MSM) and Radial Basis Functions (RBF).*

The first algorithm, Nearest Neighbour, does not scale at all with k since it only looks at one neighbour. As with all the other interpolation algorithms, the GPU kernel for this algorithm uses the arrays computed in the generation of the point distribution as described in Section 4.1 to look up the k nearest neighbours per pixel. In the case of the Nearest Neighbour algorithm, it really only looks at the first neighbour in the list and takes the point's accumulated colour obtained in Section 4.2, divides it by the amount of samples taken to accumulate that colour and applies it directly to the current pixel.

The second, middle ground algorithm, the Modified Shepard's Method as proposed by Renka [Ren88], scales linearly with k. It takes the k neighbours and uses their accumulated colours from sampling to calculate the resulting colour value for the current pixel through the formula:

$$F(x,y) = \frac{\sum_{i=1}^{k} w_i f_i}{\sum_{i=1}^{k} w_i} \tag{5}$$

where w_i is the weight for point i and f_i is the colour value for point i. The weight value is derived directly from the distance of the given point to the pixel being processed, as:

$$w_i = \frac{R - h_i}{R h_i} \tag{6}$$

where R is the distance from the current pixel to the farthest point of the k neighbours and h_i is the distance from the point i to the current pixel's location.

The third and final algorithm, the Radial Basis Function, generally has an $O(N^3)$ complexity, although there are options for precomputation. The general form to find interpolant $s(x)$ with radial basis functions, as described by Du Toit [DT08], is:

$$s(x) = \sum_{i=1}^{N} \lambda_i \phi(r) \tag{7}$$

where r is the distance function from data point i to the current pixel, $\phi(r)$ is the RBF kernel and λ_i is the weight for data point i. Since the resulting $s(x)$ at each point is known, the accumulated colour value f_i, it is possible

to set up a system of linear equations to solve for λ:

$$\begin{bmatrix} \phi(\|x_1 - x_1\|) & \phi(\|x_2 - x_1\|) & \cdots & \phi(\|x_n - x_1\|) \\ \phi(\|x_1 - x_2\|) & \phi(\|x_2 - x_2\|) & \cdots & \phi(\|x_n - x_2\|) \\ \vdots & \vdots & \ddots & \vdots \\ \phi(\|x_1 - x_n\|) & \phi(\|x_2 - x_n\|) & \cdots & \phi(\|x_n - x_n\|) \end{bmatrix} = \begin{bmatrix} \lambda_1 \\ \lambda_2 \\ \vdots \\ \lambda_n \end{bmatrix} = \begin{bmatrix} f_1 \\ f_2 \\ \vdots \\ f_n \end{bmatrix} \tag{8}$$

This matrix computation is the biggest cost and scales rapidly as the $k \times k$ size of the matrix increases, so keeping k small is beneficial.

The result of radial basis functions can vary greatly due to the kernel functions they use. For this method, two popular RBF kernels, the *Gaussian* and *multiquadric* kernels, were considered and tested, of which the multiquadric kernel provided the best results. The Gaussian kernel has the form:

$$\phi(r) = e^{-(cr)^2} \tag{9}$$

while the multiquadric kernel has the form:

$$\phi(r) = \sqrt{r^2 + c^2} \tag{10}$$

where r is still the distance function and c is an arbitrary scale parameter to change the shape of the kernel function in order to customise it to the current needs. The sensitivity to the c factor varies between kernels and obtaining the right setting to get a good result requires some experimentation.

5 Performance

The benefit gained in performance using our proposed method depends on two primary factors: the scaling factor of the generated point set and the interpolation method used. The performance of the interpolation method can also vary depending on the chosen k.

The first factor, the scaling factor in the generation of the point set, changes the density of the resulting set, lowering the final number of points stored and used for sampling the scene. This leads to an immediate benefit with a higher scaling value, although not all sets are viable to use if the user can notice a lack of resolution. Should the interpolation still restore enough quality for the user to not notice any change, that could provide a huge benefit. Note, however, that not all of the generated points in the set are within the confines of the screen at any given time which can vary performance slightly, as there will be fewer points when looking at a corner than when the user's gaze is centred on the screen and the complete denser area is used, as shown in Table 1.

Total	Centred	Corner
576,775	420,248	153,857
100%	72.86%	26.68%

Table 1: *The change in number of points on screen depending on user's gaze direction for a non-scaled point set, showing the total number of points, the number of points on screen with a centred gaze and with a gaze directed at the bottom-right corner, including the percentage of the total.*

Although based on these results, this could potentially change performance quite drastically, users will generally focus on the area around the centre of the screen. Additionally, it is fairly difficult to direct eye-tracked gaze to the very corner or edge of the screen. Regardless, the values shown in Table 1 merely portray a factor of the total number of points, which makes a more significant, constant impact on performance. Figure 14 shows the tested scaling values plotted against the corresponding number of points in resulting set, which clearly shows diminishing returns at higher scaling values, but a significant point reduction in the lower values.

The second factor is the interpolation method that is used to restore the complete image. The interpolation runs over all pixels on the screen and uses the k nearest neighour points to estimate the colour at that pixel. This means that the interpolation performance is completely independent from the scaling factor of the point set and is solely affected by the chosen k value

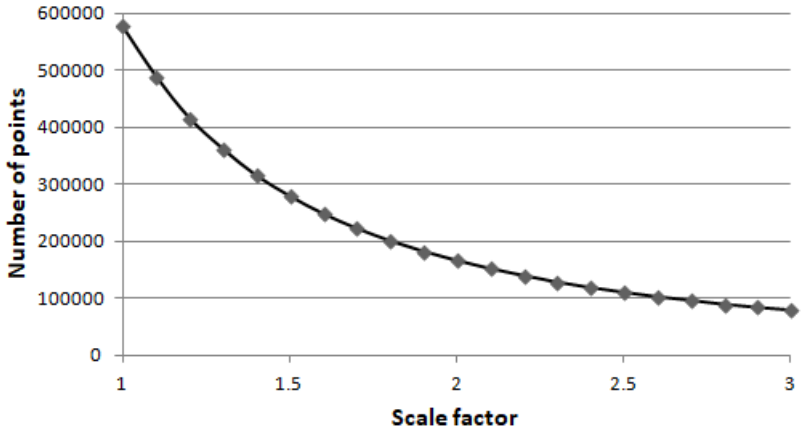

Figure 14: *The scaling factor used in the point set generation plotted against its resulting number of points in the set.*

and the resolution that is rendered to. Assuming the rendering resolution is the full resolution that the HMD uses, the only remaining parameter that can be changed in order to affect the method's performance is the k value. Although the k value does not affect the Nearest Neighbour method, there could be merit in using a higher k to increase the qualitative output from the Modified Shepard's Method or the Radial Basis Function. If a higher k increases the runtime of the interpolation, but allows for a higher scaling value that reduces the number of sampled points which saves more time than it costs, it could be worth using the higher k value. On the other hand, since colour interpolation is a local problem, the benefit of an increased k value suffers from diminishing returns, as seen in Figure 15, meaning that a higher value is not always a significant improvement. In the same vein as balancing k against the required number of points, it could prove worthwhile to choose a

Figure 15: *High eccentricity areas from the images resulting from the RBF method with different k values displaying the diminishing returns of a higher value.*

more costly Radial Basis Function over the Nearest Neighbour interpolation if the result is significantly better.

For the purposes of performing measurements and experiments, a custom ray tracing framework was extended with the proposed foveation method. To test the performance of each interpolation method, four different k values were tested: 5, 10, 15 and 20. This way the methods (with the exception of Nearest Neighbour) use between 5 and 20 nearest points in their calculations, which shows how each method scales quite clearly. Table 2 shows the results of these measurements.

	$k = 5$	$k = 10$	$k = 15$	$k = 20$
NN	2.3 ms	2.3 ms	2.3 ms	2.3 ms
MSM	4.8 ms	7.2 ms	10.2 ms	13.4 ms
RBF	4.6 ms	15.3 ms	112.0 ms	372.3 ms

Table 2: *The measured time cost to run each of the evaluated three interpolation methods: Nearest Neighbour (NN), the Modified Shepard's Method (MSM) and the Radial Basis Function (RBF). The measurements were performed on an Nvidia GTX960M GPU at the FOVE HMD resolution for one eye of 1280 by 1440.*

From these results we can tell quite clearly that the Nearest Neighbour method is indeed not affected by k, while the Modified Shepard's Method scales linearly and the Radial Basis Function method scales rapidly. The current implementation of the Radial Basis Function method clearly becomes entirely impractical at k values beyond 10. Altogether, this eliminates 5 settings for the interpolation. Although these numbers might not all appear feasible for a real-time application running at 90 frames per second where each frame has to be calculated within little more than 11 milliseconds, the measurements were performed on an Nvidia GTX960M GPU, which is about 5 times slower than the recent flagship Nvidia GPU, the GeForce GTX 1080 [Use16]. Taking this into account, the Nearest Neighbour method, all Modified Shepard's Method settings and the Radial Basis Function settings up to a k of 10 can be utilised.

6 Experiment

This section addresses the qualitative experiment performed to establish which settings are perceptually adequate. To start, the design of the experiment is laid out, explaining which concepts were considered for the test and which ones were or were not used and why. Next, the experiments setup is formulated based on the previously explained design choices. This is followed by a report on the results of the performed experiment and the section is concluded by an analysis of the results and further outcomes of the experiment.

6.1 Design

To see which methods are in fact useful and to identify the best interpolation method to use in our proposed method, it is necessary to perform a qualitative experiment comparing the different interpolation options. The selection of methods can be based on the performance found in Section 5 and those results can be combined with the outcome of the qualitative experiment to establish the most ideal interpolation method to use combined with the highest scaling factor that it allows to obtain the maximum benefit from applying our foveation method.

The experiment was to be performed using the FOVE HMD, which runs at an eye tracking rate of 120 Hz and a refresh rate of 70 Hz, and an Nvidia GTX960M GPU, so in order to approach real-time performance for the sake of the experiment, this experiment is performed using a sandbox cubemap sampling application instead of fully ray tracing a scene. This speeds up the sampling part of the application significantly to allow some leeway for the performance of the actual foveation for testing purposes. In this setup, a cubemap image is directly sampled to provide the colour values for the points, which are then interpolated according to different schemes.

The basis of the experiment is the comparison of the different interpolation methods. To limit the scope of the experiment, the fastest versions of the Nearest Neighbour, Modified Shepard's Method and Radial Basis Function interpolations were included — each with a k value of 5. This allows for comparison between the methods without relying on significant qualitative improvements at higher k values that could allow for fewer samples overall, since those higher k values suffer from diminishing returns anyway, as seen in Figure 15 in Section 5. This results in a total of three test settings.

Different numbers of samples per point were tested for temporal stability and artifact reduction. Lower values tend to allow for bigger changes from frame to frame, while a higher number of samples diminishes the effect of

one changing sample, leading to increased stability at the cost of requiring more samples. The final value that provided a significant improvement, 32 samples per pixel, was chosen as the sampling basis of our method.

A simple version of temporal anti-aliasing over the whole frame, using just the information of the previous and the current frame, was implemented and tested based on findings in previous work in order to limit or eliminate temporal artifacts, such as flickering. Since this did indeed lead to a significantly improved experience, it was decided to use this temporal anti-aliasing in all test settings.

Figure 16: *The scene of Newberry Market in San Francisco by Steel Blue used in the experiment.*

The scene to be tested with was decided to be a scene of a shopping mall in San Francisco [Ste16], as seen in Figure 16. This scene was primarily chosen for its varying contents, as it contains varying lighting, man-made, straight structurs as well as some natural elements and many busy human characters. Since a lot goes on in the scene this allows for testing different circumstances as well as keeping test subjects engaged during the test.

Lastly, these concepts need to be tested across different point sets to be able to compare their results. For this experiment, we had a range of 1.0 to 3.0 available, in steps of 0.1. The goal is to find the threshold values where test subjects start to notice the employment of foveation. This is not an exact threshold, however, since it stems from human senses and perception, which can differ between different people. To account for this, the *method of limits* from the field of *psychophysics*, as explained in Section 2.6, can be employed. The resulting averages can then be compared between the methods.

6.2 Setup

The final experiment was performed on 17 subjects of different ethnicities, ages and genders, where each subject tested each setting across a *staircase method* series. For efficiency purposes, the initial step size was set to 0.2 in an ascending series starting from the scaling value of 1.0 and at the first

reversal that was not included in the results, the step size was decreased to 0.1. From there the regular staircase method was performed until each subject reported four threshold values per setting.

The method should work and be unnoticeable regardless of whether the user knows about it, so to replicate such a situation, each subject was told about the method and what they will be presented with before the tests. Testing the detection in the subjects' peripheral sight also proved to be quite difficult if they did not know what was going on, so the full explanation also helped correctly detect the threshold values.

Prior to the experiment, each subject got a short explanation of the experiment and the FOVE HMD needed to be calibrated. The subjects then tested each method with a staircase series with breaks between each method. Before starting a series, each subject was shown the most extreme case of foveation available, with a scaling factor of 3.0, and was asked what they noticed about the image so that it would be possible to talk to them about what they see in their own words throughout the rest of the series. All in all, the experiment took around one hour per test subject.

The aim in the test setup was to simulate a real experience with foveated rendering in an interactive environment and to investigate whether users would notice anything unnatural within their vision. To simulate such an experience the subject was asked each step in the series to shift their attention to the next location or object in the scene and was asked to report anything they saw while looking for or at that location that they found strange or unnatural. The order of locations and objects was set in advance and remained the same for each test subject. In the interest of the test subjects, each series was assigned its own theme in order to maintain interest in the experiment. The first round focused on locations and interior design, the second focused on the people in the scene and the last series was based around notable occurrences such as duplicate people repeated throughout the scene. The subjects were asked at each step whether they noticed anything off with the scene and were sometimes purposefully thrown off by showing a non-foveated image to keep them attentive and critical. This was done to test subjects against falling back on previous answers and limit the the *error of habituation* and the *error of expectation*. Should they note that something is off, they were also asked what it was that they saw or where they saw it, so that further conclusions could be reached about the vulnerabilities of this foveation method.

6.3 Results

The experiment provided four threshold values per person per method with a total of 17 test subjects. These values were then aggregated into their respective means, visible in Table 3. The SPSS *MIXED* procedure for linear mixed-effect models was performed on the natural logarithms of the means for a better model fit to compare them against each other, taking into account that the means for different methods came from the same test subject. The method was set as a fixed effect and was also set as the repeated variable, while the subject ID's were set as subject with a scaled identity repeated covariance type. The p-values reported in Table 4 were obtained through pairwise comparisons of the estimated marginal means and show no significant differences between the methods, with all values being greater than 0.05.

	Mean	Std. dev.
NN	1.751	0.350
MSM	1.675	0.248
RBF	1.779	0.362

Table 3: *The means and standard deviations per interpolation method of the per-person means of their respective test results. These values represent the scaling values for the point set density.*

The same result is also shown by the univariate test performed on the estimated marginal means, resulting in $F(2, 48) = 0.403$, $p = 0.670$, which is also greater than 0.05, again showing there is no significant difference between any of the methods.

The test subjects were also questioned on what it was that they saw or noticed during the experiment. Their findings very often pointed to the same locations being noticeably irregular. These were generally locations

	NN	*MSM*	*RBF*
NN	-	0.792	0.544
MSM	0.792	-	0.386
RBF	0.544	0.386	-

Table 4: *The p values for the pairwise comparisons between the different interpolation methods. A p value below 0.05 would indicate a significant difference between the two compared methods.*

with high contrast, like a dark pillar against a light background, the lights on the ceiling of a shop or shadows cast on a lit floor, where the edges would appear to be trembling. Another commonly noticed location was the tree in the scene where it seemed as though its leaves were flickering, which is most likely also a result of high contrast through the irregular lighting as the light falls onto the individual leaves.

6.4 Analysis

The values shown in Table 4 show that no one method appears to perform better than another in reconstructing a sufficiently detailed image for the user's perception, leaving the question of which interpolation method to use in this method up to circumstance and preference. One way to make this choice is to look at the performance as reported in Section 5 and picking the method with the smallest effect on overall performance, which would be the Nearest Neighbour method. Another advantage is the sheer simplicity of the method, making it very easy to implement and work with. It is worth noting that our qualitative experiment excluded higher k values and extra additions that could lead to a better result.

Before the qualitative experiment was performed, the Modified Shepard's Method was expected to perform worse than the others, because it introduced a much more notable tunnel vision effect. The final results show that this was not the case, although the responses from different test subjects were very divergent. One test subject was quite vocal about his dislike for the Modified Shepard's Method's tunnel vision effect compared to the other two, while another test subject was bothered by that interpolation the least of all.

The Radial Basis Function, with a k value of 5, displayed some darker values in the periphery at higher density scales, which may have affected its results. This was caused by the balancing of the arbitrary scale parameter c in Function 10. Its effect had been significantly reduced, but could not be fully eliminated, so it was still present in higher scales. This problem is not present with a k of 10. Regardless, the test subjects were questioned each time they reported they noticed something strange on what it was that they saw and the areas they reported did always align with earlier expectations and areas that had been reported before across all the methods. No strange results were noticed during the tests because of this, although it might have an effect on the sensitivity to trembling edges or flickering due to high contrast.

7 Discussion

Given the results found through the performance review in Section 5 and the qualitative experiment in Section 6, the questions posed in Section 1.1 can be answered. The primary research question posed is as follows:

> What is the achievable speedup for ray tracing when utilising gaze-tracking and foveation without users being able to notice any differences with a regular image?

The answer to this follows from its individual elements, the first of which is performance. It became clear from Section 5 that the aim for the first part of the method, the sampling, is to use a scaling factor that is as high as possible, since the primary computative benefit comes from a reduction of taken samples, as was also noted by Harada [AMD14]. For the second part of the method, the interpolation, it shows that the Nearest Neighbour interpolation is the fastest method, being twice as fast as the other tested methods.

The second element of the research question is the question of which setting allows for the biggest decrease in sampling resolution, or in other words, which settings allow for using the highest scaling factor value, without the users being disturbed by the performed foveation. Section 6.3 appears to show that no one method comes out on top and all methods perform similarly. It is worth noting here that the results across all methods show similarities in which locations in the scene proved to be problematic, suggesting that the primary challenge with the proposed method appears to lie not in the individual interpolation methods, but in the general method's sensitivity to high contrast. At least while that is not improved upon, the interpolation methods show similar qualitative performance. An improvement is not trivial in the current setup, though, since the sampling is based on a precomputed set and does not take into account which areas might perceptually be more problematic. This directly leads to a potential solution to the problem, which would be to use a dynamic distribution generated per frame that uses scene data to establish high-risk areas. One such implementation by Stengel et al. [SGEM16] was shared with us in a discussion about our method at the TU Delft, which takes different factors into account such as visual acuity, eye motion, adaptation and contrast to establish their sampling pattern. As the other potential factor for the allowed resolution reduction is the interpolation method used and no method currently qualitatively performs better than any other, the choice for which method to use can rely solely on its performance, as there is no motivation to use a more costly method. This means that the

best method to use, based on these results, is the Nearest Neighbour method, thanks to its relative simplicity and the efficiency resulting from that.

Together, these results should provide some insight into the answer of the overall research question. Yet there is one issue with calculating a set achievable speedup, as the resulting means reported in Table 3 do not represent safely usable scaling factors. Instead they show the 50 percent threshold described in Section 2.6. Needless to say, the desired outcome is a foveation that is noticed in 0 percent of the cases, which was the assumption from the outset and is the reason why the point set distribution is generated based on the worst-case scenario, assuming maximum contrast everywhere in the image. As such, it makes sense that issues can arise when scaling the distribution beyond this worst-case scenario. Still, it should be possible to scale or change the distribution, since maximum contrast is a rare occurence and if the scene even contains such areas, it is often a very local issue. Since the method does not currently accomodate for local differences, it cannot be guaranteed that scaling the point set further works in all cases.

Nonetheless, even without scaling the point set further, there is a significant speed-up compared to traditional ray-traced rendering. Only a maximum of 420,248 points (see Table 1) or when rendering for each eye individually, as will generally be the case, its double, 840,496 points, need to be sampled, instead of the full resolution of $1280 * 1440 * 2 = 3{,}686{,}400$ pixels, again accounting for both eyes. This means that with a 1.0 scaling factor in the point set generation, this method already requires only 22.80% of the samples for about a 4.3 times speedup. This speedup may be diminished by the required number of samples taken to represent the sampled area, since the points—at least in the periphery—represent a bigger area than the pixels in traditional ray tracing, but does still allow for further improvements.

With further technical developments, HMD's keep increasing their display resolutions and field of view in order to provide the best immersion and most realistic experience. The current performance improvement will only become greater as displays increase in size and resolution as it will be possible to simply interpolate the values for more and more pixels instead of fully sampling and calculating them.

8 Conclusion

This thesis designed a method to perform foveated rendering within the ray tracing algorithm and tested the performance benefit and it's qualitative soundness. Foveated rendering alleviates the burden put on the existing rendering algorithms with ever increasing screen sizes and resolutions, especially in head-mounted displays, but also gives the ray-tracing algorithm the opportunity to move to the forefront in the rendering field by mitigating its primary disadvantage—its high cost. The method proposed in this thesis is a step towards a future with real-time, interactive ray-traced rendering with all the practicality and quality it delivers.

This concluding section provides a summary of the thesis with its proposed method and the resulting findings and wraps up with suggestions for potential future research topics.

8.1 Summary

The research question this thesis was built around is as shown below:

> What is the achievable speedup for ray tracing when utilising gaze-tracking and foveation without users being able to notice any differences with a regular image?

This thesis proposes a method that samples the scene's colour according to a adaptive Poisson-disc distribution of points constructed based on the contrast sensitivity function, which describes the human visual systems ability to distinguish objects across the visual field. The colours are sampled per point in an area with a radius of the Poisson-disc weight value and then stored per point to represent the sampled area. Next, a scattered interpolation method is used to provide the individual pixels of the screen with colours depending on the surrounding points and the according colours.

The primary performance gain of foveated rendering in ray tracing comes from a reduction of the rays cast, so there is a benefit as long as the number of points in the distribution multiplied by the number of samples taken to adequately depict its area is smaller than the number of pixels multiplied by the number of anti-aliasing samples taken. By scaling the point distribution the number of points it contains can be reduced, but since the distribution initially accounts for the human visual system's ability to discern contrast, this reduction leads to possible detection of the used foveation, as was found in the performed experiment. This experiment was meant to identify the interpolation method to use out of three potential candidates:

Nearest Neighbour, Modified Shepard's Method and Radial Basis Functions. As the primary issue found across all interpolation methods was that test subjects were sensitive to big changes in contrast in the periphery, no one interpolation method appeared to perform better than any other. Based on this, the proposed interpolation method to use is the least complex and most efficient—the Nearest Neighbour method. With these findings, the answer to the posed research question amounts to about a 4.3 times speedup without changing the number of samples taken per pixel, with potential extensions and improvements, like the real-time construction of a scene-aware distribution to allow for distribution scaling while accounting for high contrast areas and varying the amount of samples to take based on distance from the gaze location to reduce the number of rays cast further.

8.2 Future work

The method proposed in this thesis allows for extensions and improvements and work can still be done in areas beyond the scope of this project. Such areas include, among others, reusing and reprojecting samples from the view for one eye to the other, for example like proposed by Fujita et al. [FH14] and the related presentation by Harada [AMD14].

Another possible extension could be to update the foveal region with a higher update rate than the peripheral regions to save on some sampling time as proposed by Guenter et al. [GFD+12]. The risk this brings, however, is that asynchronicity of the different regions lead to further temporal artifacts, which are especially harmful in head-mounted displays.

Including a customised temporal anti-aliasing that is content-aware in both distance and contrast difference may be worthwhile in further alleviating temporal artifacts and troublesome high-contrast areas.

An area that could be further researched is the inclusion of contrast enhancements as described by Patney et al. [PSK+16]. They argue that thanks to the contrast enhancements they were able to further scale down their high-resolution area, while maintaining enough contrast in the periphery to keep users from experiencing tunnel vision effects.

A more efficient implementation of the Radial Basis Function could be included in the method for further research into the effect of that method with higher k values, although so far its performance cost did not seem worth the investment based on its results in the qualitative experiment.

To further reduce the number of rays cast without changing the sampling distribution, an option could be to vary the number of samples cast per point based on the distance from the gaze location. Since the foveal area directly around the gaze location essentially has regular pixel density, less samples

would need to be taken there for anti-aliasing purposes than for a point in the periphery that represents an entire area of pixels.

Lastly, the most promising extension to our proposed method would be the inclusion of a real-time, scene-aware generation of a point distribution. This way, the distribution can allow for scaling, while maintaining enough samples in high-risk, high-contrast areas to provide a temporally stable result. A possible implementation of such a system is described by Stengel et al. [SGEM16].

References

[ALP14] Ken Anjyo, John P. Lewis, and Frédéric Pighin. Scattered data interpolation for computer graphics. In *ACM SIGGRAPH 2014 Courses*, page 27. ACM, 2014.

[AMD14] AMD, Takahiro Harada. Foveated Ray Tracing for VR on Multiple GPUs. `http://www.slideshare.net/takahiroharada/foveated-ray-tracing-for-vr-on-multiple-gpus`, December 2014. [Online; accessed 1-1-2018].

[Bik12] J. Bikker. *Ray tracing in real-time games*. PhD thesis, Delft University of Technology, Delft, The Netherlands, 2012.

[Bra99] Karlheinz Brandenburg. MP3 and AAC Explained. In *Audio Engineering Society Conference: 17th International Conference: High-Quality Audio Coding*. Audio Engineering Society, September 1999.

[Bri07] Robert Bridson. Fast Poisson disk sampling in arbitrary dimensions. In *SIGGRAPH sketches*, page 22, 2007.

[BU06] Tom Bobach and Georg Umlauf. Natural neighbor interpolation and order of continuity. *GI Lecture Notes in Informatics, Visualization of Large and Unstructured Data Sets*, pages 68–86, 2006.

[Bus16] Business Insider, Jason D. Rowley. Google buys Eyefluence, which builds eye-tracking technology for virtual reality. `http://www.businessinsider.com/google-buys-eyefluence-vr-ar-eye-tracking-startup-2016-10`, October 2016. [Online; accessed 1-1-2018].

[DH06] Daniel Dunbar and Greg Humphreys. A spatial data structure for fast Poisson-disk sample generation. *ACM Transactions on Graphics (TOG)*, 25(3):503–508, 2006.

[Dig16] Digital Trends, Digital Trends Staff. Oculus Rift vs. HTC Vive — Spec Comparison — Digital Trends. `http://www.digitaltrends.com/virtual-reality/oculus-rift-vs-htc-vive/`, October 2016. [Online; accessed 1-1-2018].

[DT08] Wilna Du Toit. *Radial basis function interpolation*. PhD thesis, Stellenbosch: Stellenbosch University, 2008.

[FF05] Bengt Fornberg and Natasha Flyer. Accuracy of radial basis function interpolation and derivative approximations on 1-d infinite grids. *Advances in Computational Mathematics*, 23(1):5–20, 2005.

[FH14] Masahiro Fujita and Takahiro Harada. Foveated real-time ray tracing for virtual reality headset. Technical report, Tech. rep., Light Transport Entertainment Research, 2014.

[FOV15] FOVE. FOVE Eye Tracking Virtual Reality Headset. `https://www.getfove.com/`, May 2015. [Online; accessed 1-1-2018].

[Ges13] George A Gescheider. *Psychophysics: the fundamentals*. Psychology Press, 2013.

[GFD+12] Brian Guenter, Mark Finch, Steven Drucker, Desney Tan, and John Snyder. Foveated 3D graphics. *ACM Transactions on Graphics (TOG)*, 31(6):164, 2012.

[GP98] Wilson S. Geisler and Jeffrey S. Perry. Real-time foveated multiresolution system for low-bandwidth video communication. In *Human vision and electronic imaging*, volume 3299, pages 294–305, 1998.

[Kar14a] B. Karis. High Quality Temporal Supersampling. `https://de45xmedrsdbp.cloudfront.net/Resources/files/TemporalAA_small-59732822.pdf`, 2014. [Online; accessed 1-1-2018].

[Kar14b] B. Karis. High-quality temporal supersampling. *Advances in Real-Time Rendering in Games, SIGGRAPH Courses*, 1, 2014.

[Kic15] Kickstarter, FOVE. FOVE: The World's First Eye Tracking Virtual Reality Headset by FOVE — Kickstarter.

https://www.kickstarter.com/projects/fove/fove-the-worlds-first-eye-tracking-virtual-reality, May 2015. [Online; accessed 1-1-2018].

[Lin16] Tim Lindeberg. Concealing rendering simplifications using gaze-contingent depth of field. Master's thesis, KTH Royal Institute of Technology, 2016.

[MM05] A.V. Masjukov and V.V. Masjukov. Multiscale modification of shepard's method for multivariate interpolation of scattered data. *Mathematical Modelling and Analysis*, 10:467–472, 2005.

[PSK+16] Anjul Patney, Marco Salvi, Joohwan Kim, Anton Kaplanyan, Chris Wyman, Nir Benty, David Luebke, and Aaron Lefohn. Towards foveated rendering for gaze-tracked virtual reality. *ACM Transactions on Graphics (TOG)*, 35(6):179, 2016.

[Ren88] Robert J. Renka. Multivariate interpolation of large sets of scattered data. *ACM Transactions on Mathematical Software (TOMS)*, 14(2):139–148, 1988.

[SAS16] SKYbrary Aviation Safety. Vision (Operator's Guide to Human Factors in Aviation). http://www.skybrary.aero/index.php/Vision_(OGHFA_BN), October 2016. [Online; accessed 1-1-2018].

[SCL13] Hyeongjin Song, KK Choi, and David Lamb. A study on improving the accuracy of kriging models by using correlation model/mean structure selection and penalized log-likelihood function. In *10th world congress on structural and multidisciplinary optimization. Florida, Orlando*, 2013.

[SEB03] Hamid R. Sheikh, Brian L. Evans, and Alan C. Bovik. Real-time foveation techniques for low bit rate video coding. *Real-Time Imaging*, 9(1):27–40, 2003.

[Sen14] SensoMotoric Instruments. Eye Tracking HMD Upgrade Package for the Oculus Rift DK2. http://www.mindmetriks.com/uploads/4/4/6/0/44607631/smi_flyer_hmdpackage.pdf, 2014. [Online; accessed 8-1-2018].

[Sen16] SensoMotoric Instruments. SMI Mobile Eye Tracking HMD based on Samsung GearVR. https://www.smivision.com/wp-content/uploads/2016/10/

smi_prod_mobile_ET_HMD_SamsungGearVR.pdf, 2016. [Online; accessed 8-1-2018].

[SGEM16] Michael Stengel, Steve Grogorick, Martin Eisemann, and Marcus Magnor. Adaptive Image-Space Sampling for Gaze-Contingent Real-time Rendering. In *Computer Graphics Forum*, volume 35, pages 129–139. Wiley Online Library, 2016.

[SM14] Adam Siekawa and Supervised Radoslaw Mantiuk. Gaze-dependent ray tracing. In *Proceedings of Central European Seminar on Computer Graphics (non-peer-reviewed)*, 2014.

[Ste16] Steel Blue - Newberry Market scene. SBrusse.com - WIP. http://wip.sbrusse.com/SB_CubeMap/, 2016. [Online; accessed 1-1-2018].

[Tec16] TechCrunch, Josh Constine. The Eye Tribe Oculus Rift DK2 solution. https://techcrunch.com/2016/12/28/the-eye-tribe-oculus/, December 2016. [Online; accessed 8-1-2018].

[Tra17] Tobii Eye Tracking. Tobii Eye Tracking | Assassin's Creed® Origins. https://tobiigaming.com/games/assassins-creed-origins/, 2017. [Online; accessed 1-1-2018].

[Upl15] UploadVR, Will Mason. Oculus is working on eye tracking technology for the next generation of VR. http://uploadvr.com/oculus-is-working-on-eye-tracking-technology-for-next-generation-of-vr/, October 2015. [Online; accessed 1-1-2018].

[Upl16] UploadVR, Joe Durbin. SMI Releases Eye Tracking Developer Kit For The HTC Vive. https://uploadvr.com/smi-releases-eye-tracking-dev-kit-htc-vive/, July 2016. [Online; accessed 8-1-2018].

[Use16] UserBenchmark. UserBenchmark: Nvidia GTX 1080 vs 960M. http://gpu.userbenchmark.com/Compare/Nvidia-GTX-1080-vs-Nvidia-GTX-960M/3603vsm27242, 2016. [Online; accessed 1-1-2018].

[Val14] Valve, Steam Dev Days, Michael Abrash. What VR Could, Should, and Almost Certainly Will Be Within Two Years. http://media.steampowered.com/apps/abrashblog/Abrash%20Dev%20Days%202014.pdf, 2014. [Online; accessed 1-1-2018].

[WBLK01] Zhou Wang, Alan C Bovik, Ligang Lu, and Jack Kouloheris. Foveated wavelet image quality index. In *Proc. SPIE*, volume 4472, pages 42–52, 2001.

[Wei08] Li-Yi Wei. Parallel Poisson disk sampling. In *ACM Transactions on Graphics (TOG)*, volume 27, page 20. ACM, 2008.

[Wik07] Wikimedia Commons, Rhcastilhos. Schematic diagram of the human eye. `https://commons.wikimedia.org/wiki/File:Schematic_diagram_of_the_human_eye_en.svg`, January 2007. [Online; accessed 1-1-2018].

[Wik09] Wikimedia Commons, Vanessa Ezekowitz. Approximation of the acuity of the Human eye, horizontal cross section. `https://commons.wikimedia.org/wiki/File:AcuityHumanEye.svg`, July 2009. [Online; accessed 1-1-2018].

[Wik10] Wikimedia Commons, Markluffel. Natural neighbors coefficients example. `https://commons.wikimedia.org/wiki/File:Natural-neighbors-coefficients-example.png`, January 2010. [Online; accessed 1-1-2018].

A Experiment results

Subject	1	2	3	4	5	6	7	8	9
Gender	M	F	F	M	M	M	F	F	M
Age	57	49	31	23	23	63	45	31	26
Nearest Neighbour	1.6	1.7	1.3	2.5	1.9	1.9	1.4	1.4	2.1
	1.7	1.8	1.5	2.8	2.2	2.0	1.9	1.6	2.2
	1.6	1.7	1.3	2.4	2.1	1.9	1.7	1.4	2.0
	1.7	1.8	1.4	2.5	2.2	2.0	1.8	1.6	2.2
Modified Shepard's	1.6	1.5	1.9	1.7	1.7	2.2	1.3	1.3	2.7
Method	1.8	1.7	2.2	1.8	1.9	2.3	1.4	1.5	2.9
	1.6	1.6	2.0	1.5	1.8	2.2	1.3	1.4	2.7
	1.7	1.7	2.1	1.8	2.1	2.3	1.5	1.5	2.9
Radial Basis	1.6	1.3	1.9	1.8	1.9	1.7	1.2	1.5	1.7
Functions	1.8	1.7	2.0	1.9	2.1	2.0	2.1	1.6	1.8
	1.5	1.5	1.9	1.6	2.0	1.8	2.0	1.5	1.5
	1.8	1.7	2.0	1.7	2.3	1.9	2.1	1.6	1.7

Subject	10	11	12	13	14	15	16	17	
Gender	F	M	M	F	M	M	F	M	
Age	65	23	28	27	23	35	62	59	
Nearest Neighbour	2.2	1.5	1.4	1.9	1.3	1.1	1.4	1.5	
	2.3	1.6	1.6	2.0	1.4	1.5	1.6	1.6	
	2.2	1.5	1.5	1.9	1.2	1.3	1.5	1.5	
	2.3	1.7	1.8	2.0	1.3	1.5	1.6	1.6	
Modified Shepard's	1.8	1.4	1.7	1.8	1.4	1.5	1.8	1.7	
Method	1.9	1.5	1.8	2.2	1.5	1.6	2.0	1.8	
	1.8	1.2	1.6	2.0	1.4	1.4	1.6	1.6	
	1.9	1.3	1.9	2.1	1.6	1.5	1.8	1.8	
Radial Basis	1.9	1.2	1.3	2.0	1.2	1.3	1.3	1.3	
Functions	2.0	1.3	1.4	2.2	1.5	1.5	1.8	1.5	
	1.8	1.2	1.3	1.9	1.3	1.4	1.6	1.4	
	1.9	1.4	1.6	2.2	1.4	1.6	1.9	1.6	

Table 5: *The four scaling value results per subject per interpolation method.*