TAMPERE UNIVERSITY OF TECHNOLOGY

OLLI ETUAHO

IMAGE QUALITY METRICS FOR STOCHASTIC RASTERIZATION

Master of Science Thesis

Examiner: Professor Tapio Elomaa
Examiner and topic approved by the Faculty Council of the Faculty of Computing and Electrical Engineering on 4th of April 2012.

TIIVISTELMÄ

TAMPEREEN TEKNILLINEN YLIOPISTO
Tietotekniikan koulutusohjelma
ETUAHO, OLLI: Stokastisen rasteroinnin kuvanlaadun mittaus
Diplomityö, 70 sivua, 5 liitesivua
Toukokuu 2012
Pääaine: Ohjelmistotekniikka
Tarkastajat: Tapio Elomaa
Avainsanat: stokastiset menetelmät, rasterointi, kuvanlaatu

Kehitämme yksinkertaisen näköön pohjautuvan kuvanlaadun mittausmenetelmän stokastisella rasteroinnilla tuotetuille kuville. Uusi metriikka pohjautuu siihen, miten näköalueen hermosolut reagoivat eritaajuuksisiin signaaleihin, ja soveltaa olemassaolevien metriikoiden ideoita sekä näkötutkimuksen tuloksia. Peittoilmiö jätetään uudessa metriikassa huomiotta, koska sillä ei ole merkittävää vaikutusta tällä sovellusalueella. Uusi metriikka korreloi vahvasti HDR-VDP2:n kanssa, mutta on käsitteellisesti tätä yksinkertaisempi ja soveltuu pienempien laatuerojen mittaamiseen. HDR-VDP2:n lisäksi metriikan tuloksia verrataan MS-SSIM-metriikan tuloksiin.
Uutta metriikkaa sovelletaan erilaisilla näytteenottomenetelmillä tuotettujen kuvien vertailuun. Mittauksissa käytetään niitä varten rakennettuja kolmiulotteisia testinäkymiä muutaman laajalti käytetyn luonnollisen näkymän lisäksi. Tällä saadaan kvantitatiivista tietoa näytteenottomenetelmien tuottamasta kuvanlaadusta, vahvuuksista ja heikkouksista. Näytepistejoukon tähtidiskrepanssin ja kuvanlaadun väliltä löytyy korrelaatio, joskaan diskrepanssi ei ole yksin riittävä menetelmä arvioimaan kuvanlaatua. Laitteistoystävällinen matalan diskrepanssin näytteenottomenetelmä menestyy hyvin, mutta laatuero pikselikohtaisesti stratifioituun näytteenottoon pienenee, kun näytteiden määrä nousee.
Laatuanalyysin taustamateriaalina esitetään perusteellinen matemaattinen malli kuvasarjan tuottamiseksi dynaamisen kolmiulotteisen näkymän pohjalta.

ABSTRACT

TAMPERE UNIVERSITY OF TECHNOLOGY
Master's Degree Programme in Information Technology
ETUAHO, OLLI: Image Quality Metrics for Stochastic Rasterization
Master of Science Thesis, 70 pages, 5 Appendix pages
May 2012
Major: Software Systems
Examiner: Tapio Elomaa
Keywords: stochastic sampling, rasterization, image quality

We develop a simple perceptual image quality metric for images resulting from stochastic rasterization. The new metric is based on the frequency selectivity of cortical cells, using ideas derived from existing perceptual metrics and research of the human visual system. Masking is not taken into account in the metric, since it does not have a significant effect in this specific application. The new metric achieves high correlation with results from HDR-VDP2 while being conceptually simple and accurately reflecting smaller quality differences than the existing metrics. In addition to HDR-VDP2, measurement results are compared against MS-SSIM results.

The new metric is applied to a set of images produced with different sampling schemes to provide quantitative information about the relative quality, strengths, and weaknesses of the different sampling schemes. Several purpose-built three-dimensional test scenes are used for this quality analysis in addition to a few widely used natural scenes. The star discrepancy of sampling patterns is found to be correlated to the average perceptual quality, even though discrepancy can not be recommended as the sole method for estimating perceptual quality. A hardware-friendly low-discrepancy sampling scheme achieves generally good results, but the quality difference to simpler per-pixel stratified sampling decreases as the sample count increases.

A comprehensive mathematical model of rendering discrete frames from dynamic 3D scenes is provided as background to the quality analysis.

PREFACE

The research topic, the natural test scenes, and the core of the stochastic rasterizer software were provided by NVIDIA's research team in Helsinki. Supervisors Jaakko Lehtinen and Samuli Laine provided the outline for the research project, pointed to some reference material for the background research and provided additional guidance when needed. The low-discrepancy sampling schemes based on Sobol sequences in the stochastic rasterizer software were implemented by Tero Karras.

Thanks go also to Peter Hedman for providing valuable feedback.

CONTENTS

ABBREVIATIONS, TERMS, AND SYMBOLS

BPMSE	Band-pass Pyramid MSE introduced in Section 5.3.
Chrominance	The hue and saturation components of color or how the light is composed of different wavelengths, as opposed to how bright the light is.
CSF	Contrast sensitivity function introduced in Section 4.6.
FFT	Fast Fourier transform. An algorithm for efficiently calculating the Fourier transform of a discrete signal.
fMRI	Functional magnetic resonance imaging. A medical imaging technique which dynamically measures blood flow to brain cells. The blood flow is correlated to brain activity.
Foveal vision	Vision of things that a person is directly looking at. Opposite of peripheral vision.
GPU	Graphics processing unit, a collection of dedicated hardware for graphics processing. Usually GPUs are geared towards rasterization, though they are getting increasingly used as generic massively parallel processing units.
HDR	High dynamic range, meaning color values having a high range of brightness. Usually such color values are represented as floating point numbers as opposed to the common integer representation.
HVS	Human visual system, encompassing the eyes and the parts of the nervous system dedicated to vision.
HWLDS	Hardware-friendly low-discrepancy sampling scheme used in our measurements in Chapter 6.
Luminance	Density of total light emitted in the visible area of the spectrum, which determines the human perceived power of the light.
MOS	Mean opinion score. Mean of normalized subjective image quality scores, usually in a scale from 1 to 5 or 0 to 100.
MSE	Mean square error.
MSSIM	Mean structural similarity index, mean value of a difference map produced by the SSIM metric.
MS-SSIM	Multi-scale structural similarity index, an image quality metric based on SSIM.

Nyquist limit	Frequency limit given by the Nyquist-Shannon sampling theorem. If some content frequency of the sampled signal is above the Nyquist limit of the sample density, the sampled signal becomes aliased.
Offline rendering	Rendering computer generated images without time constraints, for example for the purposes of an architectural visualization or an animated movie.
PSD	Power spectral density. A function demonstrating how signal power is distributed among frequencies.
Psychophysics	Study of relationships between physical stimuli and the perceptions that they cause.
Receptive field	The collection of neurons that feeds signals to a single neuron.
RGB	Red, green and blue. Red, green and blue triplets are used in computer graphics to represent colors perceptible to humans.
RMSE	Root mean square error.
SPP	Samples per pixel.
SSIM	Structural similarity index, an image quality metric originally developed by Wang, introduced in Section 5.2.3.
VDP	Visible Differences Predictor, an image quality metric originally developed by S. Daly, introduced in Section 5.2.2.
Visual cortex	The part of the brain's cerebral cortex dedicated to processing visual information. Also known as the striate cortex.
z-score	Normalized value of a subjective image quality score.

1. INTRODUCTION

Taking a photograph with a camera involves light passing through a two-dimensional aperture during a non-zero shutter interval. This results in a two-dimensional (2D) image with defocus blur and motion blur, effects that are also desirable in computer generated graphics. They are important for achieving realistic appearance [42, Chapter 6], and sometimes exaggerated motion blur is used to emphasize the feeling of motion [1, Chapter 6.8]. If the rendering process is interpreted as trying to approximate a time-dependent continuous image function, motion blur is also necessary to achieve temporal anti-aliasing.

So far, real-time rendering has relied on coarse approximations of these effects, but in a couple of GPU generations *stochastic rasterization* might become a viable method to render high-quality defocus blur and motion blur in real time [48] [33] [45] [37]. Many of the principles concerning stochastic rasterization also apply to distribution raytracing, which is a similar algorithm used widely in offline rendering.

The sampling pattern used in stochastic rasterization can be regular or completely randomized and it can vary between pixels. Either way, the characteristics of the sampling pattern greatly affect perceived image quality [42, Chapter 7.2]. Time complexity of the rendering is dependent on the total number of samples taken [42, Chapter 7.2], so having an efficient sampling pattern is especially important in real-time applications.

Attempts to improve sampling approaches have been based on simple mathematical characteristics of sampling patterns and subjectively perceived image quality, but an objective perceptual image quality metric enables more accurate assessment of the quality of the rendering [48]. Ideally, stochastic rasterization sampling patterns could also be optimized automatically using an objective image quality metric. There are already precedents in optimizing 2D sampling patterns [25] and multidimensional low-discrepancy sequences [8] using other kinds of metrics.

We begin by discussing the underlying algorithms of stochastic rasterization: emulating a camera in three-dimensional (3D) space in Chapter 2 and sampling and reconstruction in Chapter 3. From there on the focus switches on to the human visual system or HVS in Chapter 4 and image quality metrics in Chapter 5. Finally, image quality metrics are applied to stochastic rasterization in Chapter 6. Results and opportunities for further research are analyzed in Chapter 7.

2. CAMERA IN 3D SPACE

3D rendering takes a 3D scene as an input and produces a two-dimensional image of it as an output [42, Chapters 1, 6]. If the aim is to create the perception of actual 3D space, perspective projection is used. Ideally, the image resulting from this perspective projection shows what the observer would see if he looked at the 3D scene through his screen. The screen can be thought of as a window to the 3D scene from some vantage point in the real world. The color at a position on the screen corresponds to the light ray passing through that position from the scene towards the aperture of the observer's eye. This basic geometry of perspective projection of a 3D scene is illustrated in Figure 2.1a.

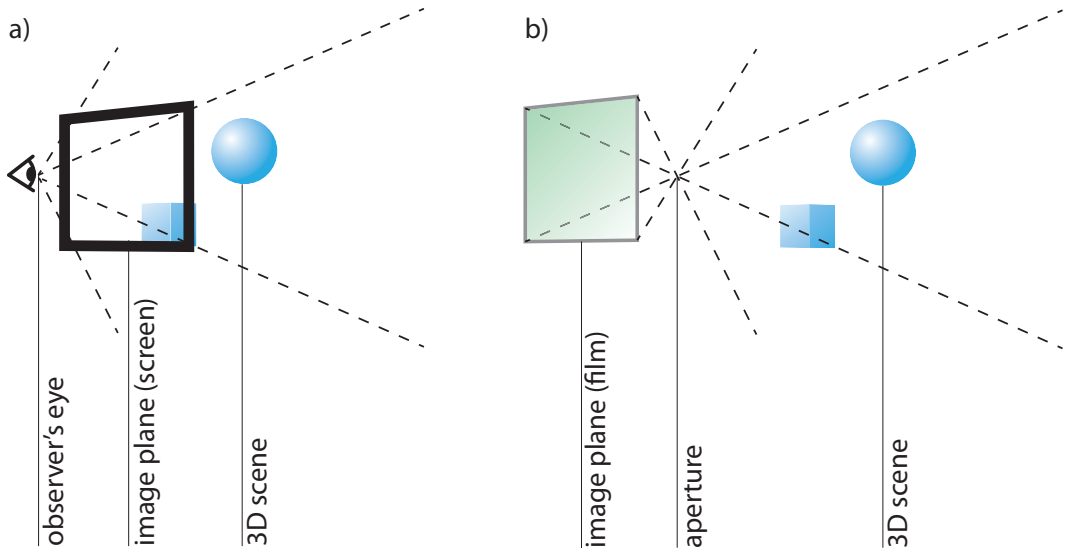


Figure 2.1: a) The eye model of projecting a 3D scene. The eye and screen are given coordinates in the 3D scene and the screen's coordinates define the image plane. b) The camera model of projecting a 3D scene onto an image plane through the camera's aperture. Note that the image on the image plane is mirrored with respect to the center of the plane.

In computer graphics terminology, the word camera is usually used instead of eye. This is despite the fact that in a real camera, the film defining the image plane is always located behind the aperture, instead of in front of it like in our ideal model of viewing a 3D scene in Figure 2.1a. This camera model of perspective projection is presented in Figure 2.1b. However, the correspondence between image plane positions and light rays is essentially equivalent in these two models. The

camera model is favorable because geometry presented in Figure 2.1b is often easier to work with, particularly in the case of depth of field focus discussed in Section 2.3.

The use of the term camera is even easier to justify if the aim is not to present a window to a 3D scene, but to actually emulate the appearance of real-world photography. This is a very common stylistic choice. The difference might not seem significant, but multiple phenomena including lens distortion, lens flare, non-circular aperture shape, and sensor noise or grain are only present in photography. The screen is then just a 2D display device for the resulting image.

The camera model is developed in the following sections. We start from a simple pinhole camera in Section 2.2 and then introduce non-point-like aperture in Section 2.3. We will also discuss shutter speed in Section 2.4. The following Section 2.1 goes over the mathematical notation used throughout this chapter.

2.1 Notation

For this purposes of this chapter, an image is thought of as an *image function*, a continuous function that returns a color value c for a given point in time t and a given x, y position on the image plane [42, Chapter 7.1.5]. x, y, and t are all defined as continuous variables.

$$f(x, y, t) \rightarrow c. \tag{2.1}$$

Everything in this chapter is explained using a camera-relative coordinate space. The camera aperture is located at the origin, facing towards the positive z axis, and the image plane is perpendicular to the z axis. The notation z_p is used for the z coordinate of the image plane, which is always negative. 3D scenes are essentially thought of as collections of static objects that have a clearly defined surface — specifically, we define function $scene(t)$ that gives a static snapshot of an arbitrary dynamic 3D scene at time t.

Function $intersect(r, s)$ intersects the ray defined by r with the scene s. The function returns the intersection with the closest surface, defined as the intersection with the minimum positive z value. If there is no such intersection, the function returns a null value.

Function $shade(i)$ is an arbitrary shading function that returns a color value based on an intersection. For the purposes of this chapter, the exact representation of color values is not important. For actually simulating light, the color values should model the whole wavelength distribution across the electromagnetic spectrum. Practical rendering algorithms usually use shading functions yielding red, green, and blue triplets that cover a subset of the shades that are perceptible to humans.

2.2 Pinhole Aperture

The simplest possible perspective rendering of a 3D scene treats the camera's aperture as a pinhole, essentially just a single point in 3D space. Each visible point in the scene is projected into one point on the image plane. This results in an entirely sharp image of the 3D scene.

Let us consider a single image plane position x, y, z_p. The line that intersects both this point and the pinhole aperture located at the origin is given by

$$r_{pinhole}(x, y) = \alpha \cdot (x, y, z_p), \tag{2.2}$$

where $\alpha \in \mathbb{R}$ is an arbitrary coefficient. If we restrict α to be negative, we get all the positions along the ray that extends from the aperture towards the positive z direction — in other words, the desired light ray corresponding to the specified image plane position.

Using this, we define the pinhole image function $f_{pinhole}$, which returns the color at the image plane position x, y and time t. Its value is determined by calculating the shading at the surface of the nearest object that intersects with $r_{pinhole}(x, y)$:

$$f_{pinhole}(x, y, t) = shade(intersect(r_{pinhole}(x, y), scene(t))). \tag{2.3}$$

2.3 Aperture and Lens

In reality, the aperture of a camera or an eye is not point-like, but rather has some finite two-dimensional shape of non-zero area [42, Chapter 6.2.3]. Additionally, a lens system located around the aperture diffracts rays from the scene depending on which part of the aperture they hit. As a result, only objects at a certain distance from the aperture are projected in perfect focus. This distance from the aperture defines the so-called *focal plane*. Points that are not at the focal plane are projected onto an area called *circle of confusion*. In effect, the image is partially blurred, and we use the term *defocus blur*. This effect is demonstrated in Figure 2.2.

The size of the circle of confusion depends on how far the projected point is from the focal plane. The circle of confusion might not always be a circle, but it takes the same shape as the aperture. In photography, the term *bokeh* is used for the shapes created by defocus blur especially if the aperture is not circle-shaped.

The z-coordinate of the focal plane is determined by how the lens system and the image plane are physically laid out. Given the z-coordinate of the focal plane z_f, and knowing that the aperture is located at the origin, we can calculate where rays intersecting at given x, y on the image plane intersect on the focal plane. This so called *focal point* $p_f(x, y)$ is given by

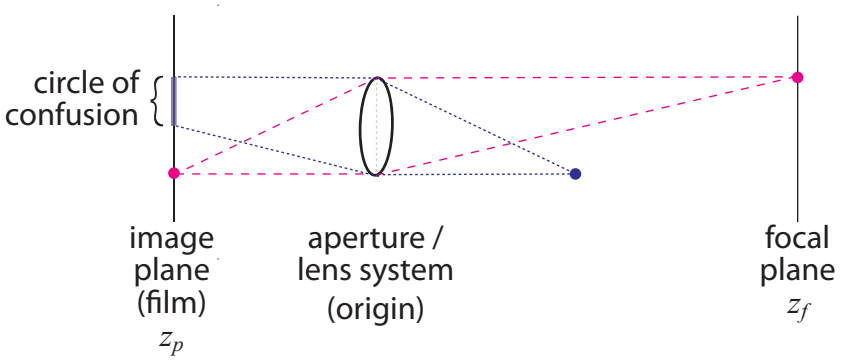


Figure 2.2: Side view of the camera demonstrating defocus blur and the circle of confusion. The lens system refracts light rays so that rays intersecting at the focal plane also intersect at the image plane. Observe the sharp projection of a point on the focal plane, and the blurred projection of another point not on the focal plane. Figure modeled after [42, Chapter 6.2.3].

$$p_f(x, y) = (x, y, z_p)(z_f/z_p). \tag{2.4}$$

Now the line equation through an arbitrary point u, v on the aperture towards the focal point corresponding to x, y on the image plane is given by

$$r(x, y, u, v) = (u, v, 0) + \alpha \cdot (p_f(x, y) - (u, v, 0)), \tag{2.5}$$

where $\alpha \in \mathbb{R}$ is an arbitrary coefficient. This time, we must restrict α to be positive to get the positions along the ray in the positive z direction.

This formulation of the ray equation is simplified in the sense that it assumes the lens system perfectly follows the focal plane model. With actual physical lens systems, this is not always entirely true [42, Chapter 6.2.3]. To model an actual physical lens system, additional distortion could be added, but this simple model is sufficient for our purposes.

In addition, we define $L(u, v)$ as follows:

$$L(u, v) = \begin{cases} 1 & \text{if } u, v \in A \\ 0 & \text{if } u, v \notin A \end{cases}, \tag{2.6}$$

where A is the set of points inside the aperture area. Using this, we obtain a formula for the color given a certain aperture position u, v, and the other coordinates x, y, and t:

$$f_{uv}(x, y, t, u, v) = shade(intersect(r(x, y, u, v), scene(t))) \cdot L(u, v). \tag{2.7}$$

To calculate the total light reaching a given point on the image plane, we have to integrate this function over the aperture dimensions u and v. Since we have defined the aperture as finite, we can define extents U and V for the aperture so that $L(u, v)$ can only be 1 when the absolute aperture coordinates are inside them, or more formally $(|u| > U) \vee (|v| > V) \Rightarrow L(u, v) = 0$. Using these, the full definition of the image function is as follows:

$$f(x, y, t) = \int_{-V}^{V} \int_{-U}^{U} f_{uv}(x, y, t, u, v) \mathrm{d}u\mathrm{d}v. \tag{2.8}$$

2.4 Shutter Speed and Time

In the previous section, the projection was done at a single instant. If our aim is to emulate photography, we have a non-zero shutter interval T during which light passes through the aperture. For the sake of simplicity, we assume that the shutter opens instantaneously and closes instantaneously. We arrive at the cinematographic image function $f_c(x, y, t_0)$, where we integrate the point color value over the frame interval:

$$f_c(x, y, t_0) = \int_{t_0}^{t_0+T} f(x, y, t) \mathrm{d}t, \tag{2.9}$$

where t_0 is the time at which the shutter opens. Note that the longer the shutter interval T is, the more light rays pass through the aperture. To achieve uniform luminance across frames with different T, the output can be scaled by $\frac{1}{T}$.

When used to render video, the resulting *motion blur* gives us a similar perception of fluid motion as with cinematography. The literature in the area often assumes that the cinematographic image function is used [34] [33].

3. SAMPLING AND RECONSTRUCTION

3.1 Reconstructing Frames from the Image Function

So far, we have treated the image resulting from the 3D rendering as a continuous function of x, y coordinates on the image plane and time t. In practice we are able to render and display only a finite number of bitmap frames of the projection. This equates to sampling the continuous image function with a regular grid in x, y, and t. In effect, the display device *reconstructs* the original continuous image function using these discrete samples, similarly to how a speaker system reconstructs an one-dimensional continuous audio signal from samples in the t dimension.

According to the Nyquist-Shannon sampling theorem, taking discrete samples of a high-frequency function will result in aliasing artifacts [50, Chapter 3]. To avoid aliasing, we need to filter out the frequencies above the *Nyquist limits* given by the density of the pixel grid and the framerate before sampling. In the pinhole aperture case with only x, y, and t dimensions, this gives us the low-pass filtered continuous image function

$$i_{pinhole}(x, y, t) = l(x, y, t) * f_{pinhole}(x, y, t), \tag{3.1}$$

where $l(x, y, t)$ is a suitable low-pass filter and the $*$ operator denotes convolution. If we assume that the low-pass filter has perfect frequency response, sampling this function gives the best possible approximation of $f_{pinhole}$ we can achieve within the limits of the density of the pixel grid and the framerate. The shape of the low-pass filter in t determines the amount of motion blur, which is actually just temporal anti-aliasing.

However, if the aim is to emulate cinematography, time is treated differently from the image plane dimensions x, y, and we use the cinematographic image function given in Section 2.4. If the cinematographic image function is chosen to be used, the low-pass filter does not take t into account, we explicitly control the amount of motion blur with the shutter interval T and the formula becomes:

$$i_{cpinhole}(x, y, t_0) = l(x, y) * \int_{t_0}^{t_0+T} f_{pinhole}(x, y, t)\mathrm{d}t. \tag{3.2}$$

This is actually equivalent to using Equation (3.1) with $l(x, y, t)$ that is separable

to a spatial filter and a box filter in t [17]. This means that the cinematographic image function is just a special case of the more generic formulation of generating frames from a dynamic 3D scene. Thinking in terms of the generic formulation gives a more complete understanding of the reconstruction problem, but thinking in terms of the cinematographic image function can be appropriate when the aim is to simulate a camera.

The implications of choosing one approach or the other may not seem significant at first, but it is easy to construct situations where content of the image function is above the Nyquist limit of the framerate, and using the cinematographic image function results in temporal aliasing [17]. One common example of temporal aliasing is rapidly spinning wheels appearing to spin backwards.

For the non-zero aperture case, we need to alter Equation (3.1) to include integrating over the u and v dimensions. The low-pass filtered image function with defocus blur is:

$$i(x,y,t) = l(x,y,t) * f(x,y,t) = l(x,y,t) * \int_{-V}^{V}\int_{-U}^{U}(f_{uv}(x,y,t,u,v))\mathrm{d}u\mathrm{d}v. \quad (3.3)$$

We now have a perfect formulation for the low-pass filtered image. This function can be sampled at the pixel locations and frame times t_i to get the final color values for each displayed pixel.

The remaining problem is that solving the result of this function analytically is not practically possible, so we will have to approximate it numerically. We will do this by sampling the color function $f_{uv}(x,y,t,u,v)$ given in Equation (2.7) across the $x, y, t, u,$ and v dimensions, and reconstructing the continuous image function $f(x,y,t)$ from these samples. To summarize, the whole process of generating bitmap frames from a dynamic 3D scene consists of the following steps [42, Chapter 7.7]:

1. Sampling the color function across the x, y, u, v, t dimensions.
2. Reconstructing the continuous image function $f(x,y,t)$ from the samples as accurately as possible.
3. Filtering out frequencies above the Nyquist limits given by the density of our pixel grid and the framerate.
4. Sampling the image function at pixel locations and the frame time t_0 to get the final color values.

In practice, steps 2 and 3 are combined into a single reconstruction filter.

To express the whole process as mathematical formula, the sampling pattern can be represented as a distribution of Dirac delta functions $s(x,y,t,u,v)$ [34]:

$$s(x, y, t, u, v) = \sum_{i=1}^{N} \delta((x - x_i) \cdot (y - y_i) \cdot (t - t_i) \cdot (u - u_i) \cdot (v - v_i)), \tag{3.4}$$

where N is the total number of samples and $(x_i, y_i, t_i, u_i, v_i)$ gives the position of the ith sample. Using this, the approximation done by sampling and reconstructing can be expressed as:

$$i(x, y, t) = r(x, y, t) * \int_{-V}^{V} \int_{-U}^{U} (f_{uv}(x, y, t, u, v) \cdot s(x, y, t, u, v)) \mathrm{d}u\mathrm{d}v, \tag{3.5}$$

where $r(x, y, t)$ is the combined low-pass and reconstruction filter.

The important thing to note here is that the different dimensions of the sampled function have different roles in the approximation of the image function. The aperture dimensions u and v are folded away completely by integrating over them, whereas the other dimensions are not. Big changes in the value of f_{uv} are also usually dependent on one or two specific dimensions at a time — it is said that the function has *low effective dimension* [20]. This means that the characteristics of lower dimensional projections of s can tell a lot about its effectiveness in capturing the shape of the function.

No matter which image function is being used, two factors determine the quality of the reconstruction. One is the sampling pattern $s(x, y, t, u, v)$, but the combined low-pass and reconstruction filter $r(x, y)$ or $r(x, y, t)$ also plays an important role. Signal processing theory gives us the encouraging result that the sinc filter can be used to perfectly reconstruct a sampled signal. However, perfect reconstruction is possible only when the original signal is band-limited in terms of frequency.

In 3D graphics, object edges unfortunately introduce infinite-frequency components to the image function, so perfect reconstruction is impossible, and using the sinc filter results in undesirable ringing artifacts [42, Chapter 7.1.2] [35]. The theory behind perfect reconstruction also relies on uniformly spaced samples. In addition, the sinc filter has infinite extents, so any practical application of it would be only an approximation. However, several other high-quality combined low-pass and reconstruction filters have been developed and are being used widely in offline rendering [42, Chapter 7.7.1]. Many good filters are found from the family of cubic filters, for example the Mitchell-Netravali filter [35].

Until recently, real-time rendering has not benefited from these advanced combined low-pass and reconstruction filters, and has been relying on the simple box filter instead [1, Chapter 4.4]. Increased quality was sought only by increasing the number of samples. However, this approach is no longer optimal due to increased

shading function complexity, which can be only partially alleviated by sharing shading results between neighboring samples of the same object. Because of this, current research in spatial anti-aliasing for real-time applications has been focused on implementing better filtering as a post-processing step [19].

Many recent real-time rendering engines use only one or two samples per pixel combined with sophisticated contrast dependent filters. This approach typically results in quality that is comparable to using four or eight samples per pixel with a box filter, which effectively demonstrates the importance of high-quality filtering. Some of these filtering implementations reuse samples across the t dimension for spatial anti-aliasing by reprojecting them [18].

3.2 Implementation: Distribution Raytracing

The most straightforward way to implement the sampling process is using *distribution raytracing*, earlier known as *distributed raytracing*. In distribution raytracing, Equation (2.7) is explicitly computed for each sample [5]. Each sampled light ray is traced back from the aperture to the nearest intersection by testing the ray equation against surface equations of objects in the scene. The shading function is typically computed by evaluating additional rays from the intersection [42, Chapter 1]. These rays can be refracted, reflected or cast towards a light source to evaluate shadowing. The naive algorithm for distribution raytracing is presented as pseudocode in Algorithm 1.

Algorithm 1 Computing a bitmap frame from a scene using the naive implementation of distribution raytracing.

```
function FIND INTERSECTION(ray, scene)
    closest ← intersection(ray, scene.background)
    for each object in scene do
        if ray intersects object and intersection(ray, object).z < closest.z then
            closest ← intersection(ray, object)
        end if
    end for
    return closest
end function

for all sample positions (x, y, t, u, v) do
    framebuffer[x,y] ← shade(find intersection(ray(x, y, u, v), scene(t)))
end for
frame ← downsample(filter(framebuffer))
```

If s is the number of samples or rays and n is the number of objects in the scene, the naive algorithm to implement raytracing has $O(ns)$ time complexity. This can be improved for common scenes by introducing spatial subdivision data structures to

efficiently narrow down which objects can potentially intersect each ray [42, Chapter 4]. Hierarchical spatial subdivision moves the time complexity of rendering closer to $O(s \log n)$, though good hierarchical structures can also be expensive to construct.

Optimized implementations also take advantage of instruction-level parallelism in modern CPUs and GPUs to evaluate multiple rays at once [4]. Still, the flexibility and simplicity of raytracing come at a cost of efficiency. Even today, raytracing is not typically used in real-time applications, even though there have been several proof-of-concept implementations [4] [28].

3.3 Implementation: Stochastic Rasterization

Since raytracing is so computationally intensive, more efficient *rasterization* algorithms have been used for 3D rendering especially in real-time applications. While modern GPUs are increasingly becoming generic parallel processing units, they are geared towards implementing an efficient rasterization based graphics pipeline. While raytracing supports multiple kinds of surface equations, in rasterization, all objects are constructed out of polygons.

We will first cover traditional rasterization, which implements pinhole rendering discussed in Section 2.2. First, vertices defining the polygons in the scene at frame time t_0 go through a perspective transformation that projects them into the unit cube [1, Chapter 2.3]. After the transformation, the x and y coordinates of the vertices have a linear relationship to their image plane coordinates, and the z coordinate in the [-1, 1] range represents depth.

The rasterization pipeline then computes which x, y sample positions each polygon covers, or the *image plane bounds* for the polygon. The pipeline iterates over the samples within the image plane bounds of each polygon and evaluates the shading function for each sample, storing the results in the *framebuffer* [1, Chapter 2.4]. Visibility of polygons that overlap in the x and y dimensions is determined by using the *z-buffer* algorithm, comparing the calculated z value of each sample to the value stored in the z-buffer at the sample's x, y position. Using the z-buffer algorithm, the polygons in the scene can be processed in an arbitrary order.

After all the polygons have been rendered into the framebuffer, the final bitmap is reconstructed and downsampled from the framebuffer samples. The rasterization algorithm is presented as pseudocode in Algorithm 2.

Many effects that are trivial to implement in a physically correct way using raytracing are usually approximated with complex algorithms in rasterization based 3D rendering. Such effects include shadows, reflections, and to some extent also motion blur and depth of field. In real-time applications, motion blur and depth of field are typically approximated using post-processing filters on pinhole rendered images. This approach is unavoidably prone to undesirable artifacts, which typically appear

Algorithm 2 Computing a bitmap frame from a scene at time t_0 using traditional rasterization and the z-buffer algorithm.

```
for all sample positions (x, y) do
    zbuffer[x, y] ← 1
end for
for each polygon in scene(t_0) do
    for each sample position (x, y) in polygon do
        if z(polygon, x, y) < zbuffer[x, y] then
            zbuffer[x, y] ← z(polygon, x, y)
            framebuffer[x, y] ← shade(polygon, x, y)
        end if
    end for
end for
frame ← downsample(filter(framebuffer))
```

especially around silhouette edges [2, Chapter 10] [33].

The *accumulation buffering* technique is another option for implementing motion blur or depth of field in real-time applications. In accumulation buffering, frames rendered with different t, u, v parameters are averaged together [2, Chapter 10]. This converges to the correct result when the amount of accumulations is increased, but coupling the dimensions together for every sample is clearly suboptimal compared to distributing the samples individually.

Stochastic rasterization generalizes the rasterization approach to separately vary the sample locations in all dimensions, so that better sampling patterns and subsequently better estimation of motion blur and depth of field become possible [3]. In traditional rasterization, the image plane bounds of a polygon are defined by the polygon edges at time t_0. In stochastic rasterization, the image plane bounds for a single polygon are expanded to encapsulate all image plane positions of the polygon in the sampled t, u, v intervals [3]. Motion can be approximated as linear to make the implementation more efficient.

The polygon visibility is then tested at each x, y, t, u, v sample position inside these expanded image plane bounds. Optimized implementations use various tricks to reduce the required number of visibility tests [26]. If the polygon is visible at a given sample position and the z-buffer test passes, the shading function is then evaluated for this sample and the result is stored in the framebuffer. Filtering and downsampling are performed similarly to traditional rasterization, though specialized post-processing algorithms have also been developed to improve the quality of the resulting image [48]. The stochastic rasterization algorithm is presented as pseudocode in Algorithm 3.

The efficiency advantages of choosing stochastic rasterization over distribution raytracing might not be immediately apparent from the pseudocode. In the worst

Algorithm 3 Computing a bitmap frame from a scene using stochastic rasterization and the z-buffer algorithm.

```
for all sample positions (x, y) do
    zbuffer[x, y] ← 1
end for
for each polygon in scene do
    for each sample position (x, y, t, u, v) in expandedbounds(polygon) do
        z_p ← z(polygon, x, y, t, u, v)
        if visible(polygon, x, y, t, u, v) and z_p < zbuffer(x, y) then
            zbuffer[x, y] ← z_p
            framebuffer[x, y] ← shade(polygon, x, y, t, u, v)
        end if
    end for
end for
frame ← downsample(filter(framebuffer))
```

case, the expanded bounds for each polygon cover the whole screen, and the algorithm has $O(ns)$ time complexity. However, the amount of samples included in the average polygon's image plane bounds can usually be made much smaller than the total number of samples s. Furthermore, iterating over the samples can be implemented more efficiently than iterating over the rays in raytracing, since many computations can be performed per vertex instead of per sample.

3.3.1 Stochastic Transparency

Transparent polygons are problematic for rasterization, since they can not simply rely on the depth-testing provided by the z-buffer algorithm [27]. In offline rendering, the *a-buffer* algorithm is sometimes used to collect a depth-sorted list of overlapping polygons for each sample. Using the a-buffer algorithm is not feasible for real-time rendering, since the amount of space needed by each sample is practically unbounded. In real-time rasterization, transparent polygons are usually separated from the rest of the scene, depth-sorted, and rendered in back-to-front order after the rest of the scene has already been rendered into the framebuffer [1, Chapter 4.5].

The need to handle transparent polygons separately can be eliminated by using stochastic transparency [27]. With stochastic transparency, the polygon is rendered as if it were either completely opaque or completely transparent at each sample, but the opaque option is chosen only when a random variable is less than the polygon's opacity value. Using this algorithm, transparent polygons can be processed in arbitrary order and the process can be completed in a fixed amount of space. This approach is somewhat similar to the earlier fixed pattern transparency techniques [1, Chapter 4.5], but can generate much better results coupled with higher sampling rates and high-quality filtering.

3.4 Sampling Schemes

We already discussed the importance of the low-pass and reconstruction filter to image quality. In stochastic rasterization, the design of the sampling pattern is equally important [42, Chapter 7.2]. The naive approach is to sample the image function at evenly spaced grid points — this is called the *uniform grid* sampling pattern. As the density of the grid is increased, the effective Nyquist frequency gets pushed higher, and we get an increasingly accurate reconstruction of the sampled function. We say that our reconstruction converges towards the correct result when the sample density is increased. Sample density is usually measured as samples per pixel (SPP).

The uniform grid pattern can work relatively well with a limited amount of dimensions, but as the amount of dimensions increases, the amount of evenly spaced samples required to accurately capture the sampled function becomes unmanageable. Sampling patterns with certain type of variability can capture high-dimensional functions much more effectively [8]. We already mentioned that lower-dimensional projections of the sampling pattern are often more important than uniformly covering the whole sampling space.

How the error resulting from inadequate sample density manifests is also important. In general, the aim of sampling scheme design is to shape the error into high-frequency random noise, which will largely be eliminated by the low-pass filter step. There are some sampling pattern analysis methods that can be used to quantify how well sampling patterns achieve this goal. We will look into these analysis methods in the following two sections. Section 3.5 explains what is meant by sampling pattern discrepancy, and Section 3.6 discusses how frequency analysis can be applied to sampling patterns.

The sections after that discuss specific sampling schemes. We first discuss sampling schemes that have been developed for anti-aliasing in the x, y dimensions in Section 3.7. After that, we give an overview of issues that are specific to stochastic rasterization sampling patterns in Section 3.8, especially achieving the desired aperture distribution, and explain specific sampling schemes that are suitable for high dimensional sampling in Sections 3.9 to 3.13.

3.5 Sampling Pattern Discrepancy

The *discrepancy* of a sampling pattern is defined as the maximum difference between a shape's volume and the relative number of sample points in that volume [42, Chapter 7.4] [13, Chapter 10.5.18]. We assume that we are sampling the unit d-dimensional volume, which has one corner located at the origin. Given a sequence of points x_i inside the unit volume and a set of shapes S, discrepancy D_N is defined

as

$$D_N(x_1...x_N, S) = sup_{s \in S} \left| \frac{\#\{x_i \in s\}}{N} - \lambda(s) \right|, \tag{3.6}$$

where s is a shape in the set of shapes S, $\#\{x_i \in s\}$ gives the number of points inside s and $\lambda(s)$ is the d-dimensional volume of s.

All sets of shapes S do not yield meaningful discrepancy values, of course. Now, S can be for example the set of all axis-aligned boxes inside the unit volume that have one corner located at the origin. The term *star discrepancy* is used for the discrepancy measure using such a set of boxes. The formal definition of star discrepancy D_N^* is

$$D_N^*(x_1...x_N) = D_N(x_1...x_N, B) = sup_{b \in B} \left| \frac{\#\{x_i \in b\}}{N} - \prod_{j=1}^{d} c_j \right|, \tag{3.7}$$

where B is a set of axis-aligned boxes b that have one corner located at the origin and another corner located at point c inside the unit d-dimensional volume. Other sets of shapes S that have been used for analyzing discrepancy in graphics research have included arbitrary boxes and spheres inside the unit volume and arbitrary linear division of the unit volume [13, Chapter 10.5.18]. However, star discrepancy is the most commonly used discrepancy measure.

Sometimes discrepancy measures can be solved analytically, but for more complex shapes and long sequences of points the analytical solution can be very hard to derive, computationally very expensive or both. For such cases, discrepancy can be estimated numerically by selecting some representative subset of shapes in S and calculating the maximum discrepancy from them.

Sampling pattern discrepancy indicates how well the set of points covers the space that is being sampled. Low discrepancy ensures that low-frequency components of the sampled function are sampled sufficiently to accurately reconstruct them. However, discrepancy is not a sufficient sampling pattern quality measure by itself [13, Chapter 10.5.18]. Particularly, low discrepancy does not ensure that aliasing error manifests as visually less distracting noise [42, Chapter 7.4]. This is easily apparent in the case of sampling in one dimension, where an evenly spaced sequence of points is optimal in terms of discrepancy, even though it does not help with aliasing error.

Experimental results from two-dimensional sampling also speak against using discrepancy as the sole measure of sampling pattern quality. Jittered sampling detailed in Section 3.10 results in lower discrepancy than Poisson disk sampling detailed in Section 3.12, even though it was found to produce more grainy images in practice [13, Chapter 10.5.18]. Discrepancy also does not take the reconstruction filter into

account, which can have just as large effect on image quality as the sampling pattern [25].

As the amount of dimensions increases, low-discrepancy sampling patterns become less uniform, and usually work relatively well as sampling patterns in practice [42, Chapter 7.4], so using discrepancy as a quality measure is more justified in the case of high dimensional sampling. Nevertheless, even high dimensional low-discrepancy sequences can be more prone to structured aliasing than true randomly distributed sequences. We will see how star discrepancy measurements of 5D sampling patterns compare to image quality measurements in Section 6.10.

3.6 Frequency Analysis of Sampling Patterns

Sampling pattern quality can be analyzed by looking at its *power spectral density* or PSD, which tells how the power of a signal is distributed in terms of frequency [32] [13, Chapter 9.2.2]. Power spectral density $R_f(\omega)$ of a signal $f(\bar{x})$ is defined as the Fourier transform of its autocorrelation function $r_f(\bar{y})$:

$$R_f(\bar{\omega}) = \mathcal{F}[r_f(\bar{y})] = \mathcal{F}[f(\bar{x}) * f(-\bar{x})]. \tag{3.8}$$

For a real-valued signal, this is equal to the square of the magnitude of the Fourier transform of $f(\bar{x})$:

$$R_f(\bar{\omega}) = |\mathcal{F}[f(\bar{x})]|^2. \tag{3.9}$$

Back in Section 3.1, we showed that a sampling pattern can be represented as a distribution of Dirac delta functions. The Fourier transform of a whole class of sampling patterns can be solved analytically in some simple cases, but in cases where the sampling pattern has a more complex definition, the overall PSD of a pattern must usually be approximated with the average discretized PSD from several instances of the pattern [32].

At least in the cases where all the sampled dimensions have an equal role in the final result, it is reasonable to assume that radially symmetric responses are appropriate [32]. Using this assumption, the information in a multidimensional PSD can be condensed into two graphs representing power and anisotropy relative to the radius from the origin.

To do this radial analysis, the frequency domain is split into a set of circular annuli that are centered on the origin [32] [24]. Radial power P_i and the variance σ_i^2 are computed within each annulus. In the two-dimensional case, the radial power is given by

$$P_i = 1/A_i \int_0^{2\pi} \int_{f_i}^{f_i+1} R_f(fcos\theta, fsin\theta) f \mathrm{d}f \mathrm{d}\theta \tag{3.10}$$

where A_i is the area of an annulus defined by radii f_i and $f_i + 1$. The series of radii is spaced evenly so that annuli further from the origin have more area than ones near the origin.

The variance σ_i^2 within an annulus used as the anisotropy measure is given by

$$\sigma_i^2 = 1/A_i \int_0^{2\pi} \int_{f_i}^{f_i+1} (R_f(fcos\theta, fsin\theta) - P_i)^2 f \mathrm{d}f \mathrm{d}\theta \tag{3.11}$$

These formulae are approximated on the discrete grid by assigning each sample to an annulus based on its location, and calculating the area of an annulus as the number of samples falling into the annulus.

In the case of the x, y dimensions, the desirable properties of the frequency spectrum are relatively well understood. We would like the power spectrum to demonstrate *blue noise* characteristics, with the DC spike at the origin and the noise concentrated at the high frequencies [24]. No significant spikes signaling structure should be visible. The anisotropy spectrum should be relatively flat, representing equal characteristics across orientations.

However, applying this kind of frequency analysis to higher dimensional sampling in the context of rendering is more complex and not as well understood [34]. This is because the sampled dimensions beyond x, y, and t do not exist anymore in our projection into a series of bitmap frames. Thus we can expect at least the u and v dimensions to play a qualitatively different role in the sampling pattern than the spatial dimensions, and it is hard to determine what exact frequency characteristics would be desirable in these dimensions.

Whether the t dimension has a qualitatively different role from the spatial dimensions is an interesting question. Looking only at the reconstruction formula, it is exactly the same as the spatial dimensions, but on the other hand time surely is different in the context of human vision. Image functions common in rendering also likely behave differently with respect to t than with respect to x or y.

Calculating the Fourier transform of a high dimensional signal is also computationally intensive — a resolution of 64 grid points per dimension already yields a grid with about 1 billion points in the case of 5 dimensions. We could exploit the fact that lower-dimensional projections of sampling patterns often matter more than the distribution in the whole high-dimensional space [20]. This means that we could get useful results from applying Fourier analysis only to the lower-dimensional projections. Still, we might miss some characteristics that are only visible from the higher dimensional data.

Summarizing the blue-noise characteristics of the power and anisotropy spectra with a single number is also non-trivial. Because of these issues, frequency analysis is a much more impractical tool for analyzing high-dimensional sampling patterns than using discrepancy measures.

3.7 Sampling Schemes for Spatial Anti-aliasing

Sampling patterns and filters for the image plane x and y dimensions is a well-researched problem [25]. These dimensions get special treatment, since sampling them is relevant for all graphics applications, not just rendering 3D scenes with motion blur and defocus blur. Many of the sampling schemes developed for spatial anti-aliasing are closely coupled with a specific filter, since they have been designed for a real-time hardware-based implementation. There is evidence that quality analysis of the sampling patterns also needs to take the filter into account [25].

Thinking of pixels as rectangular regions of the image is an incorrect mental model of the reconstruction problem [42], but we can illustrate the proximity of samples to each pixel location by making a Voronoi diagram of the pixels, which yields a grid of rectangular regions centered on the pixel locations. A reasonable requirement for any sampling pattern is that at least one sample hits any such rectangle.

In the case of just the x and y dimensions, uniform sampling can be preferable to nonuniform sampling at very low sample counts, since it prevents polygon edges appearing jittery when a box filter is used. For four uniform samples per pixel, a rotated grid scheme has been found to be optimal for accurately representing object edges [25] and is widely used. The sample locations and the filter for this scheme are illustrated in Figure 3.1a.

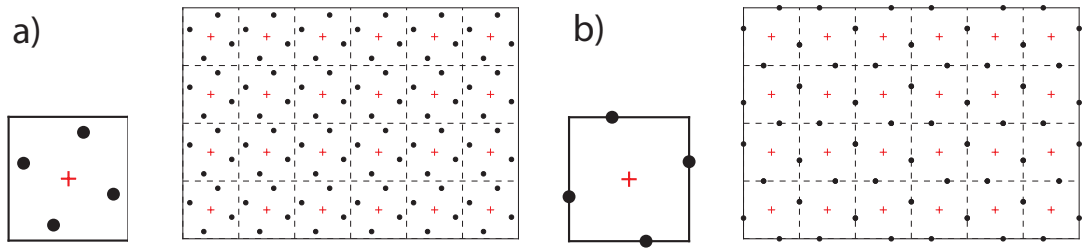


Figure 3.1: Samples on the image plane are illustrated as disks with the radius scaled according to the sample weight in the filter. In these patterns all the samples have equal weights. The grid lines indicate the extents of the square neighborhoods of each pixel center. a) The rotated grid sampling pattern. b) The Flipquad sampling pattern. The figures are modeled after figures in [25].

With an uniform sampling scheme, closely packed object edges can cause disturbing repeating artifacts, but using an interleaved pattern that repeats across just two

pixels can make them mostly imperceptible [22]. The Flipquad scheme combines this kind of interleaved sampling with sharing samples across pixels to facilitate a sampling rate of just two samples per pixel, while achieving similar results as the rotated grid scheme. The sample locations and the filter of the Flipquad sampling scheme are illustrated in Figure 3.1b.

3.8 5D Stochastic Sampling Schemes

Simple sampling patterns developed for the x and y dimensions are not directly applicable when three more dimensions are added. While simple uniform sampling still performs relatively well in the case of just two dimensions, structured aliasing is much more easily visible when sampling also in the t, u, and v dimensions. See Figure 3.2 for an example. In the following sections we will investigate better sample generation approaches, such as random sampling featured in Figure 3.2c.

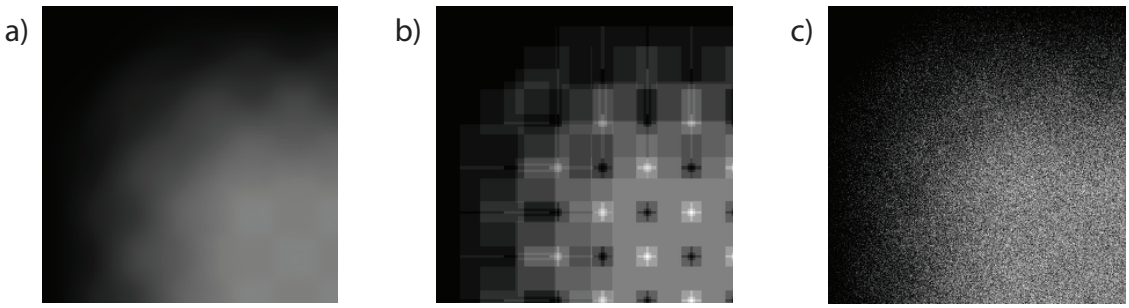


Figure 3.2: a) The reference rendering of a checkerboarded billboard with heavy defocus blur. b) Using uniform grid sampling with 16 discrete grid points on the aperture results in significant structured aliasing. c) Random sampling with 16 samples per pixel trades the structured aliasing to random noise.

In this section, we concentrate on the specific problem of how to achieve u, v distributions that follow the desired aperture shape. We need to achieve this goal while simultaneously maintaining the qualities of a well-formed point distribution.

The simplest way to achieve an arbitrary sample distribution in the u, v dimensions is using the *rejection method* [42, Chapter 13.3.2]. As long as our sampling pattern guarantees that there is no correlation between the aperture coordinates and other dimensions, simply discarding sample locations that fall out of the aperture distribution $L(u, v)$ does not change the desirable qualities of the sampling pattern.

Unfortunately, using the rejection method will result in a lower sample density, so the sample generator should be able to generate more sample locations to be filtered to achieve a desired constant sample density. This is not a trivial requirement in the case of all sample generation methods, since the generated samples might be well distributed only with certain specific total numbers of samples.

In most cases, circular aperture shape is desirable. To achieve this, the u, v coordinates need to be uniformly distributed across the unit disk. For this, we do not necessarily need to use the rejection method, but can transform u and v coordinates from a square distribution to a disk-shaped distribution.

To do this, we need a square-to-disk mapping that has three key qualities: preserving fractional area of shapes, preserving continuity, and low distortion of shapes [49]. Preserving fractional area of a shape R is defined as

$$\frac{\lambda(R)}{\lambda(S)} = \frac{\lambda(m(R))}{\lambda(m(S))}, \tag{3.12}$$

where λ is a function determining the area of a shape, S is the unit square, m is the mapping function, and R is an arbitrary shape inside the unit square. Continuity is defined as usual, and it should be preserved also in the inverse disk-to-square mapping.

A formal definition for the low distortion of shapes is more complex and outside the scope of this thesis. For the purposes of sampling, preserving the relative distances of neighboring points after the mapping to some small degree of error is basically enough.

A simple mapping of concentric squares into concentric circles, from here on *concentric-square mapping*, satisfies all three criteria [49] [42, Chapter 13.6.2]. The mapping is illustrated in Figure 3.3. We assume that our u and v coordinates are separately uniformly distributed across the $[-1, 1]$ range. The concentric-square mapping for a point inside the unit square can be constructed from this mapping covering one fourth of the unit square:

$$m_p([u_S, v_S]) = \left[u_S \cos\left(\frac{\pi v_S}{4u_S}\right), u_S \sin\left(\frac{\pi v_S}{4u_S}\right) \right], \tag{3.13}$$

where u_S and v_S are the coordinates on the unit square and m_p is the partial mapping [49]. The fourth covered by this mapping is the right side of the both diagonals of the square where $u_S > |v_S|$. This fourth is highlighted in Figure 3.3. The mappings for the other three parts of the unit square are symmetric to this one.

3.9 Random Sampling

In random sampling, sample locations are simply taken from an uniform random distribution in all dimensions [42, Chapter 7.3]. As long as the random numbers have sufficiently high quality, this trades all structured aliasing artifacts to less distracting random noise [13, Chapter 9.1.2], which makes it superior compared to uniform grid sampling. Similarly to uniform grid sampling, reconstruction from random sampling is guaranteed to converge towards the correct result as the sample

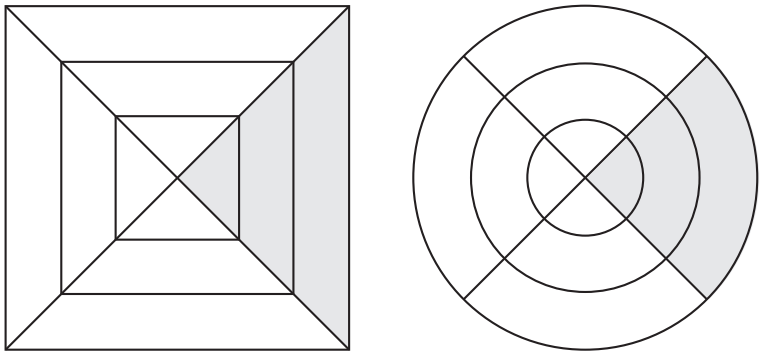


Figure 3.3: Mapping concentric squares to concentric circles. The line segments on the left are mapped to the curves on the right. The fourth given in Equation (3.13) is highlighted.

count is increased [13, Chapter 9.2.2].

The downside of random sampling is that the samples are not evenly distributed in space, and thus do not cover it effectively: there can be large volumes inside the sampling space that do not have a single sample, and also dense concentrations of samples [13, Chapter 9.4]. In other words, random sampling patterns have generally high discrepancy. Lower-dimensional projections of the pattern share these same characteristics. As a result, distracting medium-frequency artifacts may appear in the rendered images, though such artifacts will not have any regular structure.

3.10 Stratified Sampling

Random sampling can be improved upon by *stratifying* the sampling space: instead of generating random sample locations inside the whole sampling space, we first divide the sampling space into equally sized strata, each of which will contain an equal number of samples [42, Chapter 7.3]. In principle, dividing the samples equally among pixels is already a kind of stratification of the sampling space, though we will use the term random sampling in this case.

In practice, we split each dimension of the sampling space into equally sized non-overlapping intervals, so that the whole d-dimensional sampling space is split into equally sized axis-aligned boxes. Then we assign one sample to each box, or stratum. This can be understood as randomly jittering the sample locations of a uniform grid pattern. Because of this, the term *jittered sampling* is sometimes used for this sampling scheme.

The number of strata needed for this division is the product of the numbers of strata in each dimension, and thus quickly rises as a function of the number of dimensions. The number of strata needed to have n strata in each of d dimensions is n^d. For two strata per dimension, a total of 32 strata would already be needed to sample 5 dimensions. For four strata per dimension, 1024 strata would be needed.

This is an instance of an effect known as the *curse of dimensionality* [42, Chapter 7.3].

The curse of dimensionality can be mitigated by using so-called *padded stratification* [42, Chapter 7.3], which exploits the low effective dimensionality of the sampled function. In a padded stratified sampling scheme, dimensions are stratified separately and the resulting sets of coordinates are then randomly associated with each other. An effective approach is to stratify image plane coordinates in two dimensions, time in one dimension and aperture coordinates in two dimensions, and then randomly associate these three sets of coordinates together for each pixel or a small neighborhood of pixels.

If the sample locations inside the strata are generated randomly, the stratified sampling schemes have the same guarantees as random sampling: error will manifest as random noise, and the reconstruction will converge to the correct result as the number of samples is increased. However, since the stratified patterns cover the sampling space more evenly than uniform random sampling, they have lower discrepancy. This results in the error being pushed to higher frequencies. The padded stratified sampling scheme is relatively easy to understand and implement, but can still produce good images with a reasonable number of samples.

3.11 Latin Hypercube Sampling

Latin hypercube sampling, or *n-rooks sampling* as it is sometimes called, is a special case of padded stratification [42, Chapter 7.3]. In Latin hypercube sampling, all of the dimensions are stratified separately, and the coordinates from different dimensions are then randomly associated with each other. This process is typically applied separately for each pixel.

Latin hypercube sampling has the advantage that any number of samples can be used — the number of samples does not need to be a product of the numbers of strata in different dimensions. This makes it more flexible than stratified sampling, though it has less guarantees considering sampling pattern discrepancy and the two-dimensional projections, and can thus result in lower image quality. On the other hand, its superior one-dimensional projections can help it attain better image quality in some cases.

3.12 Poisson Disk Sampling

One problem with stratified sampling is that samples may still be clumped together at the edges of neighbouring strata [42, Chapter 7.5] [13, Chapter 9.4]. Sampling patterns satisfying the *Poisson disk* property solve this problem: all sample points have a minimum distance to all other points in the pattern. The distance measure

needs to wrap around the edges of each sampled interval to avoid sample points clumping to the edges of the sampled intervals. Patterns satisfying the Poisson disk property have low discrepancy and good frequency characteristics [24] [21].

Generating patterns that have the Poisson disk property can be done by using the *dart throwing algorithm* [24] [42, Chapter 7.5]: on each iteration, a random sample point is added to the pattern only if it does not break the Poisson disk property. This is computationally a very expensive process, since a large number of candidate points may need to be tested before a suitable one is found. Furthermore, the resulting total number of points is hard to determine in advance [13, Chapter 10.5.12].

If we are willing to use a pattern that has only roughly the same characteristics as a true Poisson disk distribution, we may use more efficient sample generation methods. The *best-candidate method* consists of generating a set number of candidate points on each iteration, and choosing the one that has maximum distance to the points already in the pattern [42, Chapter 7.5]. Any number of points can easily be generated with this method.

There is some empirical evidence that generating a Poisson disk distribution or a best-candidate pattern directly in five dimensions is not the best approach [56], likely due to the low effective dimensionality of the sampled function. Rather Poisson disk distributions should be generated separately for the time, aperture, and spatial domains and then randomly associated with each other, similarly to the recommended approach to padded stratification. Still, we will concentrate on five-dimensional best-candidate patterns in our measurements to explore their characteristics.

More sophisticated methods for creating or approximating Poisson disk patterns also exist, though research on the area has mostly focused on generating two-dimensional patterns [24] [21] [13, Chapter 10.5.15]. Of course, they can still be applied to higher-dimensional sampling by combining multiple lower-dimensional distributions generated with such methods. A method based on Voronoi diagrams should directly extend to higher dimensions [21], though it is significantly more complicated to implement than the best-candidate and dart-throwing methods.

Other distance measures than Euclidean distance can also be applied to generate a best-candidate pattern in five dimensions. Since we prefer good lower-dimensional projections rather than an optimal distribution in the whole sampling space, a distance measure that more effectively penalizes proximity in any single dimension can be used instead of separately generating the distributions in the different dimensions. We will test one such improved distance measure, which is the sum of the square roots of the absolute distances in all dimensions:

$$d = \sqrt{|x_a - x_b|} + \sqrt{|y_a - y_b|} + \sqrt{|t_a - t_b|} + \sqrt{|u_a - u_b|} + \sqrt{|v_a - v_b|}. \quad (3.14)$$

To the best of our knowledge, this is a novel variation of generating a best-candidate sampling pattern for rendering.

3.13 Low-Discrepancy Sampling

There are some pseudo-random sampling schemes that aim specifically for low discrepancy. These methods can produce patterns that have some improved characteristics even over Poisson disk sampling, and some of them are also remarkably efficient to implement in software or hardware [42, Chapter 7.4.2]. Only a few lines of code are required.

The first two methods we will discuss are based on the *radical inverse* function Φ_b, which converts a nonnegative integer to a decimal value in $[0, 1)$. This conversion is done by reflecting the digits d_i in base b around the decimal point:

$$\Phi_b(n) = 0.d_1 d_2 ... d_m. \quad (3.15)$$

The d-dimensional Halton point sequence is defined using the radical inverse function with a different base in each dimension. The bases must be relatively prime to each other, so we will use the first n prime numbers $p_1 ... p_n$ [42, Chapter 7.4.2]. A point in the Halton sequence is given by

$$x_i = (\Phi_2(i), \Phi_3(i), \Phi_5(i), ..., \Phi_{p_n}(i)). \quad (3.16)$$

Any subsequence of the Halton sequence is well-distributed. If the number of points N is known in advance, we can use the Hammersley sequence, which gives slightly lower discrepancy:

$$x_i = \left(\frac{i}{N}, \Phi_2(i), \Phi_3(i), ..., \Phi_{p_n}(i) \right). \quad (3.17)$$

Again, other relatively prime bases for the radical inverse function could also be used. Both the Halton and the Hammersley sequences have asymptotically optimal star discrepancy [42, Chapter 7.4.2].

However, Halton and Hammersley sequences can still exhibit regular patterns as the base of the radical inverse increases [42, Chapter 7.4.2]. Permuting the digits fed to the radical inverse function can be used to mitigate this problem, so that these methods can be applied to higher-dimensional sampling.

We will not cover the construction of the much more complex Sobol sequences

here in detail, but will cover some of the theory behind them to understand how they behave in terms of discrepancy. The approach for constructing Sobol sequences that was used in our measurements was taken from [20].

So-called (t, m, d)-*nets* are a theoretical construct which can be used to explain why some low-discrepancy sequences are able to attain low star discrepancy [20]. Let $d \geq 1$ be our amount of dimensions, $b \geq 2$ and $0 \leq t \leq m$ be integers. A point set of b^m points is a (t, m, d)-net, if each box in a division of the sampling space into b^{m-t} identical rectangular boxes contains exactly b^t points. In the Sobol sequence, $b = 2$. An example of a $(0, 4, 2)$-net is given in Figure 3.4.

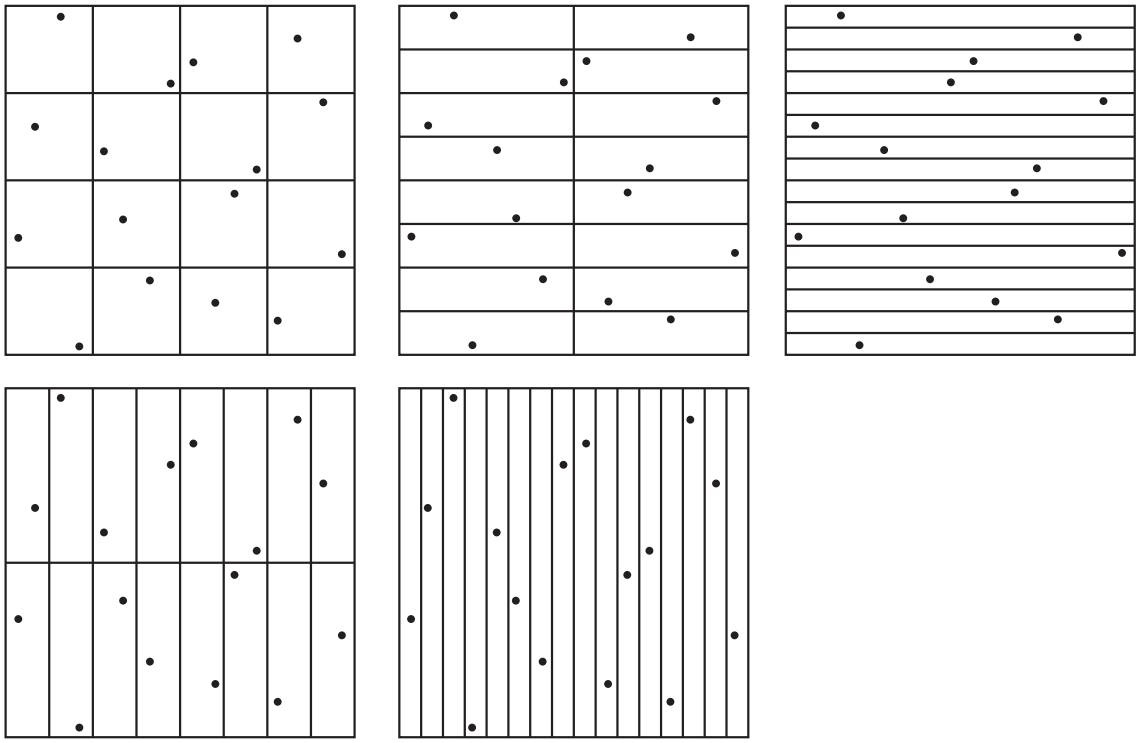


Figure 3.4: Example of a $(0, 4, 2)$-net. There are $2^{4-0} = 16$ boxes in each of the five possible unique divisions. Each box contains $2^0 = 1$ point. Figure modeled after [42, Chapter 7.4.3]

A point set that is a (t, m, d)-net has an upper bound on its star discrepancy depending on $1/b^{m-t}$ [20]. Thus, the smaller the t value is, the lower the discrepancy. The t-value of a Sobol sequence is given by

$$t = \sum_{j=1}^{d} (s_j - 1), \tag{3.18}$$

where s_j is the degree of the primitive polynomial in dimension j, used to generate the Sobol sequence [20]. In practice this means that the t value increases as more dimensions are added. The t value of lower dimensional projections can be optimized by choosing the parameters for generating the sequence carefully [20].

To have different low-discrepancy point sets for each pixel, it is possible to scramble (t, m, d)-nets while preserving their desirable properties [42, Chapter 7.4.2]. The implementation used in our measurements also achieves co-operation between pixels by intelligently distributing partially scrambled indices of the Sobol sequence between pixels.

4. THE HUMAN VISUAL SYSTEM

4.1 Overview

The human visual system encompasses the eyes and the parts of the brain that are specialized in processing visual input [12, Chapters 1, 6] [15]. An overview of the visual system is presented in Figure 4.1. Light first passes through the cornea, the pupil, and the lens of the eye. It then gets converted to nervous signals in the retina, which are passed on through the optic nerve fibers and the *lateral geniculate nuclei* to the visual cortex in the rearmost part of the cerebral cortex. There are also some other brain regions that react to visual input, but the visual cortex is thought to be the most important.

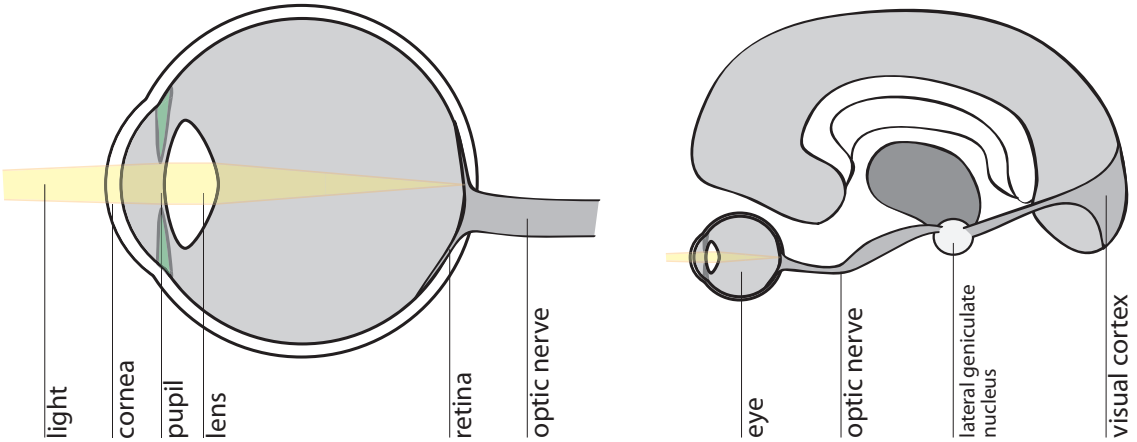


Figure 4.1: A simplified diagram of a cross section of the HVS. The human eye is presented on the left, and the whole HVS in the context of the human brain on the right. The figures are modeled after figures in [12] [15] [14, Chapter 2]. This side view omits details related to stereo vision: there are two eyes and two lateral geniculate nuclei, left and right, and roughly half of the optic nerves from each eye cross over to the lateral geniculate nucleus on the other side of the head [12]. The visual cortex is likewise divided to the left and right halves. The lateral geniculate nucleus on the left is responsible for processing the signals from the left half of both retinas, and the lateral geniculate nucleus on the right is responsible for processing the signals from the right half of both retinas [12, Chapter 1].

The optics of the human eye are relatively simple. The aperture provided by the pupil limits the amount of light entering the eye, and the curved cornea and the lens inside the eye refract the incoming light to focus on objects at a certain distance. Microelectrode experiments and closely examining dissected eyes have also given us

a relatively good idea of how the retina operates [12, Chapters 1, 6] [15]. The retina is discussed in more detail in Section 4.2.

The operation of the lateral geniculate nuclei and the visual cortex are much more complex, but psychophysical experiments and fMRI measurements of the brain have given us at least a rough overview on how the early stages of the visual cortex operate. fMRI only sees blood flow, not neurons actually firing, which makes its temporal resolution inherently limited [38]. However, its spatial resolution is relatively good, and there have been some studies of human visual cortex activity in response to static images. Recently, researchers at Berkeley also correlated human brain activity to characteristics of video samples, and subsequently used this data to reconstruct videos of the visual experience from fMRI measurements [38]. The visual cortex is discussed in more detail in Section 4.5.

4.2 Retina and Visual Acuity

The eyes, and especially the retina have a couple of interesting properties concerning the study of image quality. There are two kinds of photoreceptor cells in the retina: *rods* and *cones*. Rods are simple cells that respond to a relatively wide array of wavelengths and are sensitive to very low amounts of light, as little as single photons. This low-light vision provided by the rods is referred to as scotopic vision.

Whereas there is only one type of rods, there are three different types of cones. Each type of cone reacts to a different distribution of wavelengths of visible light, thus providing the basis of color vision [15, Chapter 8]. The cones are larger than rods and mostly concentrated on the fovea, where they are tightly packed to provide a small area of accurate color vision [15, Chapter 3] [12, Chapter 6]. The high-luminance color vision provided by cones is referred to as photopic vision.

The density of cells in the fovea provides an upper bound on the *visual acuity*, the ability to resolve small details, of the human vision. It is most useful to measure visual acuity as an angular quantity — how big angle does one phase of a regular high-contrast grating have to cover in the observer's field of view to become visible. In the fovea, the visual acuity is at most about 100 cycles/degree [15, Chapter 3]. The visual acuity of the fovea gives us an upper bound on the useful spatial resolution of computer generated imagery.

The chemical reactions that result from light hitting either type of photoreceptor are not instantaneous, but take place over several milliseconds [13, Chapter 1.3]. This, in effect, performs low-pass temporal filtering on the received signal, eliminating flickering above roughly 60 Hz. This can be understood as an upper limit for the temporal acuity of the HVS: as a general rule, a video signal displayed at 60 Hz is indistinguishable from a time-continuous image. If the discrete video signal does not have significant temporal aliasing, an even lower framerate can be practically

indistinguishable from 60 Hz.

The photoreceptor cells connect to the optic nerve via two layers of nerve cells inside the retina, *bipolar cells* and *ganglion cells*. There are about 100 receptors for each optic nerve fiber, so these first layers of nerve cells already perform some processing on the visual input [15, Chapter 3] [12, Chapters 1, 6]. In the fovea, almost every receptor has its own ganglion cell and optic nerve fiber, so this does not affect the peak visual acuity, but a much larger array of cells feeds a single ganglion cell in the peripheral areas [15, Chapter 3].

The receptive fields of different ganglion cells also overlap each other, and there is evidence the ganglion cells perform a sort of a convolution function over the output of the receptors [12, Chapters 5-6]. The shape of this convolution function is roughly the difference of two Gaussian functions. It is hypothesized that this provides the earliest processing stage of an edge detection method used by the HVS [12, Chapter 5].

4.3 Color Vision

The HVS perceives color from the outputs of the three different types of cone receptors. It is important to note that the three types of cones do not directly correlate to the monochromatic red, green, and blue color channels of display devices, but they react to overlapping wavelength distributions of light.

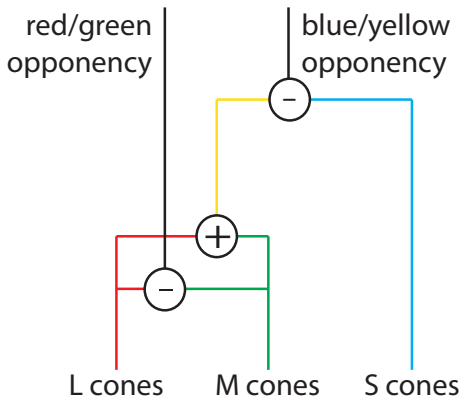


Figure 4.2: The red/green and blue/yellow opponency pairs in the HVS.

These color signals from the cones are then processed into two opponent pairs in the HVS [12, Chapter 17] [55]. First the ratio of red and green cone signals determines the output of the red/green opponency path. If the red and green signals are in balance, they cancel each other out. The total of the red and green cone signals form yellow, which determines most of our perception of luminance, and is in turn the opponent of the blue cone signal on the blue/yellow opponency path. The opponent pairs are illustrated in Figure 4.2.

4.4 Luminance Adaptation

The photoreceptors adapt to different levels of luminance, which means that the HVS's sensitivity to luminance changes depends on the ambient level of luminance [14, Chapter 2] [6]. One model for this nonlinearity in the luminance response is given by:

$$\frac{R}{R_{\max}} = \frac{L^n}{L^n + \alpha^n}, \tag{4.1}$$

where $R/R_{\max}$ is the normalized luminance response of the photoreceptor, L is the input luminance and α is the semisaturation constant [6]. The exponent parameter n for the model should be chosen from between 0.7 and 1.0. The semisaturation constant grows linearly with respect to the light level to which the part of the retina is adapted.

4.5 The Structure of the Visual Cortex

The axons of the ganglion cells of the retina terminate in the two lateral geniculate nuclei, which perform the second stage of filtering to the visual signal [12, Chapters 1, 9]. What exactly happens in the lateral geniculate nucleus is unknown, but the mapping from the retina to the lateral geniculate nucleus is still quite straightforward. The connections from the retina project to 6 bilayers in each lateral geniculate nucleus — in effect, each nucleus holds 12 copies of the left or right half of the retinal images. The area dedicated to foveal vision is relatively large compared to peripheral vision.

The lateral geniculate nuclei also have lots of feedback connections from the visual cortex. One hypothesis is that the lateral geniculate nuclei are a part of an incremental object recognition system [12, Chapters 1, 9]. According to that, the initial impression from the feedforward connections determines rough categorization, and the feedback connections from the visual cortex help in choosing details of the image signal for further recognition processes.

The visual cortex itself is organized in hypercolumns, which are hypothesized to perform different kinds of low-level visual analysis to parts of the image signal [12, Chapter 9]. However, the complexity involved means that their function has not been deduced from neurophysical experiments [12, Chapter 4]. Instead, the knowledge of the visual cortex builds largely on psychophysical evidence, and neurophysical experiments have only been able to confirm the psychophysical findings in some cases.

4.6 Contrast Sensitivity Function

One of the defining characteristics of the low-level behavior of the visual cortex is its sensitivity to contrasting patterns, which is defined by the *contrast sensitivity function* or *CSF* [12, Chapter 4] [13, Chapter 1]. The value of the CSF is determined as the amount of contrast needed to make a given pattern perceptible. The parameters and the shape of the CSF have been determined from testing the perceptibility of different contrast gratings. The most important parameter is the spatial frequency of the pattern, which gets its upper bound from the acuity of foveal vision. The CSF for a vertical grating of changing frequency is demonstrated in Figure 4.3.

Contrast grating orientation and distance from the center of the gaze also affect perceptibility. Horizontal and vertical gratings are slightly more perceptible than diagonal ones. Scotopic vision has a different CSF from photopic vision, and generally lower contrast sensitivity [13, Chapter 1.4].

According to Mannos' model, the frequency-relative CSF for a contrast grating the has approximate form of

$$c(f/f_0)e^{-f/f_0}, \tag{4.2}$$

where f is the frequency of the contrast grating and f_0 is the frequency with the highest perceptibility. Estimates for f_0 range from 3 to 5 cycles/degree [29].

The most telling indication of the operation of the HVS are *adaptation aftereffects* that affect the CSF [12, Chapter 4] [13, Chapter 1]. Adaptation to a constant stimulus causes nerve cells to inhibit further signals, which can decrease the perceptibility of a certain kind of grating. This results in a notch in the CSF, which shows that there are neural pathways tuned to detecting certain spatial frequencies and orientations of contrast gratings. In effect, the collection of neural pathways dedicated to a given spatial frequency can be thought of as a band-pass filter on the image signal.

There is not yet much exact quantitative data about these frequency-tuned channels of the HVS, but experiments suggest that they split the frequency spectrum into 1 to 3 octave wide sections [51] [55]. In one comprehensive study of the macaque visual cortex, the width of the frequency tuning varied greatly among cells, but the most narrowly tuned cells had about 1 octave worth of frequency bandwidth and 26 degrees of orientation bandwidth [51].

Some mathematical results explain why this kind of frequency and orientation selectivity might have evolved, and add further support for the findings [11]. In typical natural images, the energy is divided quite evenly among the octaves on the frequency spectrum. This results in that the information in natural images can be effectively coded by the kind of orientation and frequency selective channels present

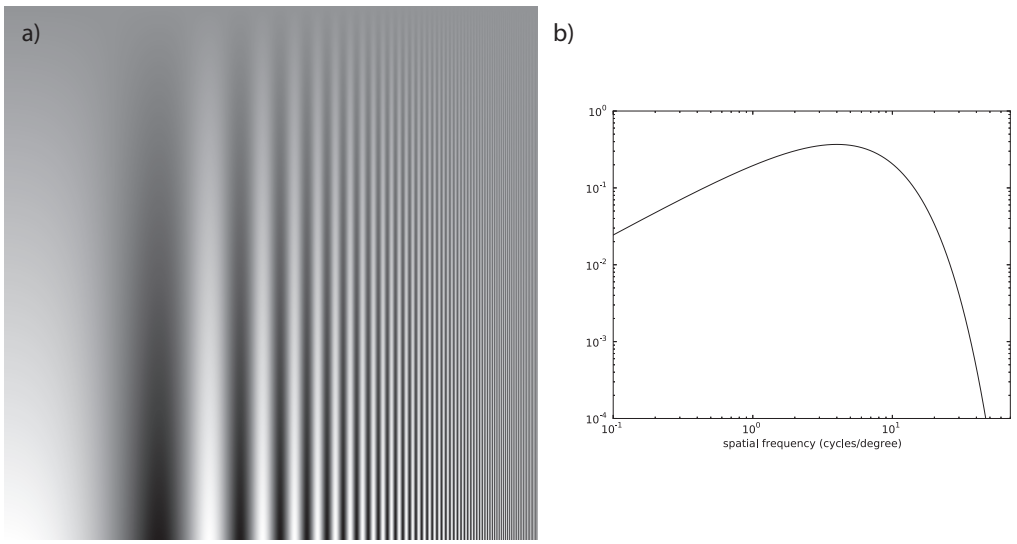


Figure 4.3: a) A demonstration of the approximate shape of the frequency-relative CSF for a horizontal sinusoidal grating. Contrast decreases linearly towards the top of the image, and frequency increases exponentially towards the right. If viewed from a certain distance, the frequencies on the left and on the right need more contrast than the frequencies in the middle to become visible. The grating was rendered so that viewing it from a distance of 10 times its width results in a smooth frequency scale from 0.1 cycles per degree to 70 cycles per degree. The figure is modeled after [12, Chapter 4]. b) The corresponding CSF plot using the model in Equation (4.2) and $f_0 = 4$ cycles/degree.

in the visual cortex.

The CSF model has been criticized on the basis that contrast sensitivity has only been defined with regards to relatively simple signals, most typically sinusoidal gratings [54]. It is unclear how well the perception of more complex signals can be understood in terms of the CSF. Still, CSF is perhaps the most useful quantitative model of the HVS that is based on its currently measurable functionality.

4.7 Masking

Different parts of the signal content can affect each other's perceptibility in the HVS. A certain signal might not be visible in the presence of another, or might only become perceptible in the presence of another. These phenomena are known as masking [55]. We will call the signal that hides another signal from perception the *masking signal*, and the signal that is being hidden the *masked signal*.

Masking is a gradual phenomenon depending on the strength of the masking signal. This behavior has been modeled using a threshold elevation model, where the visibility threshold of the masked signal begins rising after the masking signal reaches a set contrast level [6]. Below this level, there is no significant masking effect.

Masking effects have been determined for both sinusoidal gratings and uniform noise fields [6]. Most often, masking has been modeled inside a single frequency-tuned channel, though there is some recent evidence that inter-channel masking also occurs [55]. Nevertheless, masking is strongest when the masking and masked signals contain similar spatial frequencies.

In the case of stochastic rasterization, high-frequency error manifests only in the presence of low-frequency blur in the reference signal, so there is usually no significant chance of masking effects coming into play. This makes masking somewhat uninteresting in the context of stochastic rasterization. This is in contrast with pinhole rendering, where sharp textures in the reference signal would commonly mask any approximation errors that manifest as noise. As the amount of blurriness in an image rendered with stochastic rasterization increases, the artifacts not only get worse, but also stand out more.

Sometimes different degrees of blurriness are overlapping in the reference signal, for example in the presence of motion blurred objects against a static background. However, in this kind of cases masking effects can only hide or distort the approximation error. If our aim is to measure the worst-case quality resulting from using a certain sampling pattern, taking masking effects into account can actually be counterproductive.

5. IMAGE QUALITY METRICS

5.1 Mean Square Error and Peak Signal-to-Noise Ratio

Most practical approaches to measure image quality are based on comparing degraded images to some perfect reference image [40] [10] [55]. The degradation might be caused by imperfections in analog systems, lossy image compression, or in our case by approximation errors made when rendering 3D scenes. Let the reference signal be A, and let the degraded signal be B. The mean square error or MSE between these two discrete signals is defined as

$$\mathrm{MSE}(A, B) = \sum_{i=0}^{N} (A[i] - B[i])^2, \tag{5.1}$$

where N is the sample count of the signals being compared, and $A[i]$ and $B[i]$ are sample values of signals A and B at index i [40]. Signals with more than one dimension can be compared by using any bijection from values of i to the coordinate space. Note that A and B are interchangeable, so MSE is symmetric. The square root of MSE called root mean square error or RMSE is also sometimes used [14, Chapter 8.1]. RMSE is defined as

$$\mathrm{RMSE}(A, B) = \sqrt{\mathrm{MSE}(A, B)} = \sqrt{\sum_{i=0}^{N} (A[i] - B[i])^2}. \tag{5.2}$$

Due to its simplicity, MSE is widely used as a signal quality metric [40] [53]. It is sometimes presented in the form of signal-to-noise ratio SNR or peak signal-to-noise ratio PSNR. SNR determines the amount of noise relative to the actual signal, and PSNR relative to the peak signal, which is in our case the maximum possible color luminance value. PSNR is usually given on a logarithmic scale, computed using

$$\mathrm{PSNR}(A, B) = 10 \log_{10} \frac{max^2}{\mathrm{MSE}(A, B)}, \tag{5.3}$$

where max is the maximum value of a single sample. If the samples were 8-bit luminance values, max would be 255 [16].

Like one could infer from the previous chapter, MSE, RMSE, and PSNR are somewhat arbitrary when used as a quality measure. MSE can be interpreted as

the energy of the error signal, but there is no perceptual basis for it as an error metric [10]. Obviously, MSE or PSNR do not take masking or contrast sensitivity into account, so they correlate with subjective image quality measurements only in a very limited fashion [16] [55] [40].

The PSNR resulting from lossy image or video compression has been studied extensively [10] [55] [16]. While PSNR usually changes monotonically with respect to subjectively perceived quality when just adjusting the parameters of a compression method for a single type of content, it cannot really be considered as an accurate quality metric when either the compression method or the content varies [16]. While the artifacts typically appearing in stochastic rasterization are not as varied as artifacts resulting from image compression, this still makes PSNR less than ideal for our purposes.

The problems of PSNR are easy to demonstrate by using an artificial example. Consider a defocus blurred white rectangle rendered using one sample per pixel and a box filter. The maximum PSNR for this example would be achieved by having a white rectangle with rounded corners in the image, even though an image with the density of white pixels varying smoothly along the blur radius would clearly represent the blurring more accurately.

5.2 Perceptual Metrics

There are many existing image quality metrics that try to take the characteristics of HVS into account. Most such perceptual metrics take a bottom-up approach to the HVS, commonly trying to simulate some combination of non-linear luminance response of the retina, variable contrast sensitivity, and different kinds of masking. Some other perceptual metrics start from some intuitive notion of the end goals of the HVS, such as object recognition, and construct the metric from there in a top-down fashion.

Conversation on objective perceptual metrics dates back to the seventies. At that time, the term *distortion measure* was also used [29]. Nearly all of the metrics have been designed to detect artifacts appearing in image or video transmission or storage — whether resulting from imperfections in analog systems or lossy digital compression.

The terminology used by different perceptual metric authors is very varied [40]. The area of study lies between psychology, signal processing, and information theory, and terminology has been adapted from the traditions of each of these disciplines.

Some of the first perceptual metrics were based on weighting the frequency domain representation of the images with the CSF [40] [36]. This can outperform simple PSNR, but the connection to the HVS is not exactly clear, since the HVS does not actually perform Fourier analysis.

Some metrics improve on this simple application of Fourier analysis by operating on a series of band-pass filtered images [40]. Many of these are based on Peli's definition of contrast perception [41]. According to this definition, the contrast at a location on a certain frequency band is the band-pass image pixel divided by local luminance. The local luminance is defined as the value at that location in a low-pass filtered image.

Damera-Venkata's [7] *Noise Quality Measure* or *NQM* is one of the metrics based on Peli's definition of contrast. NQM uses a bank of cosine-log band-pass filters in the frequency domain to split the image into a pyramid of band-pass images. NQM also enhances Peli's contrast model by introducing per-pixel variable thresholding based on frequency masking. NQM fared quite well when evaluated against a limited set of subjective test results [7], but it has not been proven by a more comprehensive evaluation.

Interestingly, NQM's pyramid of band-pass images is defined beginning from the lowest frequencies and the high-frequency residual is completely omitted from consideration. Because the amount of steps in the pyramid is limited, the high-frequency residual could actually contain a lot of the interesting frequency content in the case of large, fullscreen images. For this reason, the metric would probably require readjustment before it could be applied to typical graphics rendering cases.

Nowadays, *Visible Differences Predictor* or *VDP* [6] is one of the most widely used perceptual metrics [10]. Further development since the original publication has led to the existence of several different variations of VDP. Other notable contemporary metrics include *Structural Similarity Index* or *SSIM* [53] and its variants. In Section 5.2.2 and Section 5.2.3 we will look into these two metric families in more detail. In the next section, we will look into how perceptual metrics have been evaluated against subjective measurements.

5.2.1 Evaluating Metrics Against Subjective Measurements

Perceptual metrics can be evaluated by comparing the metric results to subjective measurements. The extent of the resulting correlation is often called the *performance* of the metric [40].

The results of subjective measurements are usually given in the form of *mean opinion score* or *MOS*. Mean opinion score is calculated by taking the mean of image quality scores on a linear scale given by test subjects [46]. Before taking the mean, the image quality scores given by subjects need to be normalized. One such normalization approach is to convert them to so called *z-scores* [46]. The z-score z_{ij} for image j and subject i is given by

$$z_{ij} = \frac{d_{ij} - \bar{d}_i}{\sigma_i}, \tag{5.4}$$

where $\bar{d}_i$ and σ_i are the mean score and the standard deviation of scores given by subject i, respectively. This ensures that different scoring patterns of individual test subjects do not skew the results.

There are large databases of MOS measurements that have been used in the evaluation of several different metrics [40]. One widely used database is the LIVE database [46] [47], which contains images with 5 different types of distortions. The difference mean opinion score from more than 20 test subjects has been recorded for each of the nearly 800 image pairs.

The TID2008 database improves on LIVE by including a set of 17 different distortion types, many of them different kinds of noise [44] [43]. In total, TID2008 contains 1700 test images. TID2008 is also based on an order of magnitude more subjective measurements, though it has achieved this by compromising on the consistency of the test setups. However, TID2008 claims smaller normalized variance and thus higher accuracy in its measurements than LIVE. A summary of the distortion types found in LIVE and TID2008 is given in Table 5.1.

Distortion Type	LIVE	TID2008
Additive Gaussian noise	X	X
Additive noise with increased intensity in color components		X
Spatially correlated noise		X
Masked noise		X
High frequency noise		X
Impulse noise		X
Quantization noise		X
Gaussian blur	X	X
Image denoising		X
JPEG compression	X	X
JPEG2000 compression	X	X
JPEG transmission errors		X
JPEG2000 transmission errors	X	X
Non eccentricity pattern noise		X
Local block-wise distortions of different intensity		X
Mean shift		X
Contrast change		X

Table 5.1: Summary of distortion types in LIVE and TID2008.

One major weakness of the LIVE and TID2008 databases is that they use only photographic source images, with the only exception of one artificial test image in TID2008 [47]. In addition, none of the source images in the databases exhibit signifi-

cant motion blur, and only a few of them exhibit significant defocus blur. Paintings, drawings, and 3D renderings are not included in either database. The goal in 3D rendering is not always photorealism, even though stochastic rasterization is used to specifically recreate effects that are usually found in photography. As a redeeming factor, the subject matter of the photographs and their visual characteristics are quite varied in both of the databases.

Of the distortion types included in LIVE, the white noise distortion type is most similar to the approximation errors seen in stochastic rasterization. In LIVE, the white noise was added to the images in linear RGB color space, where the color values were scaled to the [0,1] range. The noise values were taken from a Gaussian distribution with standard deviations varying from 0.012 to 2.0. LIVE contains 174 sample images with white noise applied. The same standard deviation was used for all color channels, but noise was applied separately to each color channel. The resulting color values were again clamped to the [0,1] range after the noise was applied.

This type of white noise still differs from stochastic rasterization approximation errors in some respects. First of all, the approximation errors are not uniformly distributed across the image, but the amount of noise depends on the amount of depth of field blur and motion blur at each pixel. Second, with high-quality sampling patterns the approximation errors are concentrated on the high frequencies. Third, the approximation errors may result in structured artifacts depending on the sampling pattern. Fourth, the approximation errors are correlated across different color channels.

Only some of the additional noise types added in TID2008 are relevant for our purposes. Additive Gaussian noise with increased intensity in color components, spatially correlated noise, impulse noise and quantization noise are not similar to the approximation errors seen in stochastic rasterization. Masked noise and high-frequency noise are relevant, however.

Masked noise is distributed nonuniformly across the image, even though it is concentrated on the areas with sharp features instead of blurred areas, and high-frequency noise has a spectrum more similar to the approximation errors produced by high-quality sampling patterns. However, these noise types are still uncorrelated across color channels and produce less varied artifacts than stochastic rasterization. These differences are significant enough that constructing a database of subjective measurements from rasterized images would add value compared to using the TID2008 database.

All the same, LIVE and TID2008 measurements and metrics evaluated against them still provide an adequate point of comparison for our purposes. Especially results from TID2008 are interesting, since the database includes a more varied set

of noise distortions than LIVE. The white noise, high-frequency noise, and masked noise results from TID2008 could provide one basis for evaluation of a perceptual quality metric for stochastic rasterization.

5.2.2 Visible Differences Predictor

The VDP image quality metric is constructed in a bottom-up fashion. The metric takes the two image signals being compared, and performs transformations that aim to simulate the processing happening in the HVS. The simulation includes three major characteristics of the HVS: nonlinear luminance response, frequency- and orientation-tuned channels, and masking. The transformed signals are then compared at each sample location to determine whether the difference at that location is likely to be visible. The result of this process is a difference map of pixels that are likely to be perceptually different.

It is important to note that the first version of VDP does not provide a way to compare the quality of two different images, but it only shows where perceptible errors are located in each image [6]. Producing a difference map can be useful in adaptive lossy compression, but in most cases the results still need subjective analysis. The original version of VDP is interesting mostly due to the quality of its HVS simulation rather than its usefulness as a metric.

However, further publications on VDP have defined some aggregation functions that pool the pointwise error perceptibility predictions into a single number to be used for comparison. For *High Dynamic Range Visible Differences Predictor 2* or *HDR-VDP2*, such an aggregation function was defined by correlating the results of several possible aggregation functions with databases of subjective image quality measurements and choosing the aggregation function that produced the best correlation [31]. As a result, HDR-VDP2 achieved better correlation with some of such databases than any other known method.

Comparative studies of perceptual metrics have also ranked VDP as one of the most versatile methods available [10]. VDP is able to assess many different kinds of errors and disregard those that are not important to human perception. As a mature metric, its results have been confirmed many times over by psychophysical measurements.

One big weakness of the VDP family of metrics is that it does not provide a way to measure color differences. All images need to be converted to a monochrome color space before measurement. In the context of our application, taking masking effects into account can also be counterproductive, as discussed in Section 4.7. This should not be a significant factor in most of the test images we will use in our measurements, however.

5.2.3 Structural Similarity Index

SSIM variants are another well known metric family that has also achieved high correlation with databases of subjective image quality measurements [53]. Unlike VDP, SSIM follows a top-down design philosophy, starting from the assumption that the HVS's function is to extract structural information from natural images. The metric does not try to explicitly model the processes of the HVS, but splits the image spatially into windows and does statistical analysis on them to extract structural information.

The SSIM index for each window separately estimates differences in luminance, contrast, and structure using mean intensity, standard deviation, and covariance. These values are then combined to form the per-window SSIM index by multiplying them together. Luminance difference $l(x, y)$ between local window a in image A and the corresponding local window b in image B is calculated using

$$l(a,b) = \frac{2\mu_a\mu_b + C_1}{\mu_a^2 + \mu_b^2 + C_1}, \tag{5.5}$$

where μ_a is the mean intensity of window a and μ_b is the mean intensity of window b. This formula results in a value between 0 and 1, with 1 signifying zero difference in luminance. The C_1 constant is included to stabilize the formula when denominator is very close to zero.

The contrast difference $c(a, b)$ between two corresponding windows is calculated using a similar formula:

$$c(a,b) = \frac{2\sigma_a\sigma_b + C_2}{\sigma_a^2 + \sigma_b^2 + C_2}, \tag{5.6}$$

where σ_a is the standard deviation of window a, σ_b is the standard deviation of window b and C_2 is another stabilizing constant. The stabilizing constants C_n used in SSIM are calculated using:

$$C_n = (K_n L)^2, \tag{5.7}$$

where L is the maximum luminance value and K_n are user-supplied positive constants that should be significantly smaller than 1.

The structural similarity $s(a, b)$ of two corresponding windows is calculated as the stabilized Pearson correlation coefficient with the formula:

$$s(a,b) = \frac{\sigma_{ab} + \frac{C_2}{2}}{\sigma_a\sigma_b + \frac{C_2}{2}}, \tag{5.8}$$

where σ_{ab} is the covariance of the local windows a and b. Finally, the SSIM index of two corresponding local windows is calculated as a weighted product of $l(a, b)$,

$c(a,b)$, and $s(a,b)$:

$$\mathrm{SSIM}(a,b) = l(a,b)^{\alpha} c(a,b)^{\beta} s(a,b)^{\gamma}. \tag{5.9}$$

It is recommended that the weighting constants are chosen as $\alpha = \beta = \gamma = 1$, so that the formula becomes:

$$\mathrm{SSIM}(a,b) = \frac{(2\mu_a\mu_b + C_1)(2\sigma_{ab} + C_2)}{(\mu_a^2 + \mu_b^2 + C_1)(\sigma_a^2 + \sigma_b^2 + C_2)}. \tag{5.10}$$

The result of the per-window analysis is a difference map as in VDP, but the original SSIM has two characteristics that sets it apart from VDP [53]. First of all, SSIM aims to capture the extent of the differences rather than just determining whether the difference is visible in a binary fashion.

Second, SSIM also recommends *Mean SSIM* or *MSSIM* of the spatial windows as a pooling method. MSSIM has achieved moderate correlation with databases of subjective image quality measurements [31]. MSSIM is defined as

$$\mathrm{MSSIM}(A,B) = \frac{1}{n}\sum_{j=1}^{n} \mathrm{SSIM}(a_j, b_j), \tag{5.11}$$

where A and B are the images being compared, a_j and b_j are the image content at the jth local window, and n is the total number of local windows.

SSIM has been improved further in *Multi-Scale SSIM* or *MS-SSIM*, which computes SSIM of a pyramid of downsampled low-pass filtered images and pools the results together [54]. Each filtering pass applies a box low-pass filter and downsamples the image by a factor of 2. The original image is indexed as Scale 1. The formula for MS-SSIM is

$$\text{MS-SSIM}(a,b) = [l_M(a,b)]^{\alpha_M} \cdot \prod_{j=1}^{M} [c_j(a,b)]^{\beta_j} [s_j(a,b)]^{\gamma_j}, \tag{5.12}$$

where l_j, c_j, and s_j give the luminance, contrast and structure difference at scale j, respectively, and M is the maximum scale of low-pass filtering. From the formula we can see that the luminance difference is calculated only for the highest Scale M, whereas the contrast difference and the structure difference are calculated for every scale. It is suggested that $\alpha_j = \beta_j = \gamma_j$ and that weights would be normalized so that $\sum_{j=1}^{M} \gamma_j = 1$ to enable direct comparison of different weight vectors [54].

With optimized weights, MS-SSIM has achieved very good correlation with databases of subjective image quality measurements, including LIVE and TID2008. Its performance is comparable to or in some cases better than that what has been achieved with HDR-VDP2 [31]. However, it is notable that MS-SSIM did not achieve higher performance than PSNR in the case of white noise measurements in LIVE

[46]. This casts doubts on MS-SSIMs usefulness in measuring stochastic rasterization quality, and it should be used only as a secondary reference.

5.3 Band-pass Pyramid Mean Square Error

In Section 4.6, it was shown that the HVS effectively performs band-pass filtering to the image signal at different frequencies as one of the stages in its object recognition process. This has been a common premise in many existing image quality metrics. In this section, we develop a new objective image quality metric that concentrates on simulating this aspect of the HVS. This is a good approach for measuring stochastic rasterization quality, since other major characteristics of the HVS do not play a significant role when considering errors that are typical for stochastic rasterization. As explained in Section 4.7, taking masking effects into account could even be counterproductive.

By disregarding secondary characteristics, we are able to develop a metric that is conceptually and computationally simple. Of course, we will end up with a metric that is not as versatile as many existing perceptual quality metrics, but tuned only to this specific application.

That being said, the metric should still achieve high correlation with more versatile perceptual quality metrics when used to evaluate images produced by stochastic rasterization. This is an important secondary goal for the metric and one indicator of the metric's quality.

The band-pass filter we need for the metric is simplest to implement as a difference between two Gaussian low-pass filters. Using a Gaussian filter especially benefits computability, since it is the only circularly symmetric two-dimensional filter that is separable into two applications of one-dimensional filters in the x and y directions [50, Chapter 24]. If the dimensions of the image are N times N, and the dimensions of a two-dimensional filter are M times M, the time complexity of two-dimensional convolution is $O(N^2M^2)$, whereas time complexity of two one-dimensional convolutions is $O(N^2M)$.

Using the difference of Gaussians as our filter also has some basis in the physiology of the HVS, even though the frequency and orientation selective channels in the HVS are usually modeled by Gabor filters [7] [6]. The difference of Gaussians is a good model for the response of the retinal nerve cells with wide receptive fields as explained in Section 4.2. The difference of Gaussians is also a good approximation of the *Laplacian of Gaussian* filter, which is used as an edge detection filter in computer vision applications [9].

The one-dimensional Gaussian filter for standard deviation σ is given by

$$g_\sigma(x) = \frac{1}{2\pi\sigma^2} e^{\frac{w^2}{2\sigma^2}}, \tag{5.13}$$

where w is the distance from the filter center. Using this, we define $\mathrm{LP}(A, \sigma)$ as either the signal itself or the signal low-pass filtered with two one-dimensional Gaussian filters of a given standard deviation:

$$\mathrm{LP}(A, \sigma) = \begin{cases} A * g_{\sigma x} * g_{\sigma y} & \text{if } \sigma \neq 0 \\ A & \text{if } \sigma = 0 \end{cases}. \tag{5.14}$$

where x in subscript denotes convolution in the x direction and y in subscript denotes convolution in the y direction. Again, these two convolutions are equivalent to taking a two-dimensional convolution with a two-dimensional Gaussian filter. In practice, we will do the convolutions between a discrete image signal and a discrete Gaussian filter kernel. To do this efficiently, the Gaussian kernel is windowed to a conservative width of $\lceil \sigma \cdot 3 \rceil \cdot 2 + 1$.

This discretization of the Gaussian kernel has some implications for its accuracy. First, the sum of the resulting weights can be different from 1. To compensate for this, the weights are normalized by multiplying them with a constant to make the sum of the weights 1 again. In addition, the actual standard deviation computed from the discrete filter kernel can be different from the σ that was used to specify the kernel. We chose not to compensate for this, since it only slightly skews the ratio between the standard deviations of two consecutive filter kernels used when computing the difference of Gaussians.

Now, the band-pass filtered image is defined as the difference between two $\mathrm{LP}(A, \sigma)$ images:

$$\mathrm{BP}(A, k) = \mathrm{LP}(A, \bar{\sigma}[k+1]) - \mathrm{LP}(A, \bar{\sigma}[k]), \tag{5.15}$$

where $\bar{\sigma}$ is a series of standard deviations indexed by k. Using this, we define the band-pass filtered difference measure $\mathrm{FRMSE}(A, B, k)$ by taking the RMSE of two image signals A and B filtered with a difference of Gaussians filter:

$$\mathrm{FRMSE}(A, B, k) = \sqrt{\sum_{i=1}^{N} (\mathrm{BP}(A, k)[i] - \mathrm{BP}(B, k)[i])^2}, \tag{5.16}$$

where N is the total amount of samples in $A_{BP[k]}$. Our new metric is simply a weighted sum of these FRMSEs. For this, we use a series of σ values $\bar{\sigma}$ starting from $0, \frac{1}{2}, 1, 2$ and doubling from there on each iteration, so that each FRMSE we compute covers a range of one octave. The low-frequency residual beyond a chosen k is omitted from consideration.

It is appropriate to use these 1-octave sections based on what we know of the HVS and the spectral energy distribution found in natural images [41]. In natural images, the spectral energy is divided roughly evenly among this kind of sections,

and dividing the images this way produces an efficient image code. The width of 1 octave is a reasonable choice based on what is known about the bandwidth of cortical simple cells.

The resulting metric labeled *Band-pass Pyramid MSE* or *BPMSE*, is defined as

$$\mathrm{BPMSE}(A, B) = \sum_{k=1}^{M} W[k] \cdot \mathrm{FRMSE}(A, B, k), \tag{5.17}$$

where W is a vector with length M containing arbitrary weights for the series of FRMSE values. We choose a good set of weights for BPMSE in Section 6.7.

5.4 Application to Color Images

So far all our metrics have been directly applicable only to monochrome images. However, display devices typically have red, green, and blue color channels, which then get processed by the HVS to signals of luminance and opponency pairs of green-red and blue-yellow.

The simplest way to compare color images is to convert them to a monochrome color space. This method is used in many contemporary metrics, SSIM and VDP among them [6] [53]. However, conversion to a single channel can greatly diminish the perceptibility of errors as demonstrated in Figure 5.1, which is not desirable.

Ideally, we would use a perceptual color difference measure. Many such measures have been developed based on the CIE Lab color space, which aims to be perceptually uniform [40] [13, Chapter 1]. Simply taking the Euclidean distance between CIE Lab values is a relatively accurate measure, but several more advanced measures have also been developed, some reaching better correlation with psychophysical results [40].

However, our aim is to develop a metric for analyzing stochastic rasterization, which typically operates in linear RGB color space, and color does not play a significant role in the rasterization algorithm. It can be thought that each of the RGB color channels in the resulting image is a separate rendering with different monochromatic shading parameters.

We do want to take color differences into account on a rudimentary level to reflect the errors in each color channel, but a perceptual approach to color quality adds little value. Furthermore, to evaluate the perceptual approach to color we would need an altogether different sample set than the one we get with stochastic rasterization.

Based on this reasoning, we use the Euclidean distance of the RGB values as the basis of BPMSE for RGB images. This replaces the simple luminance difference that was used with monochrome images. Equation (5.18) defines the difference of two RGB color values as their Euclidean distance in RGB space.

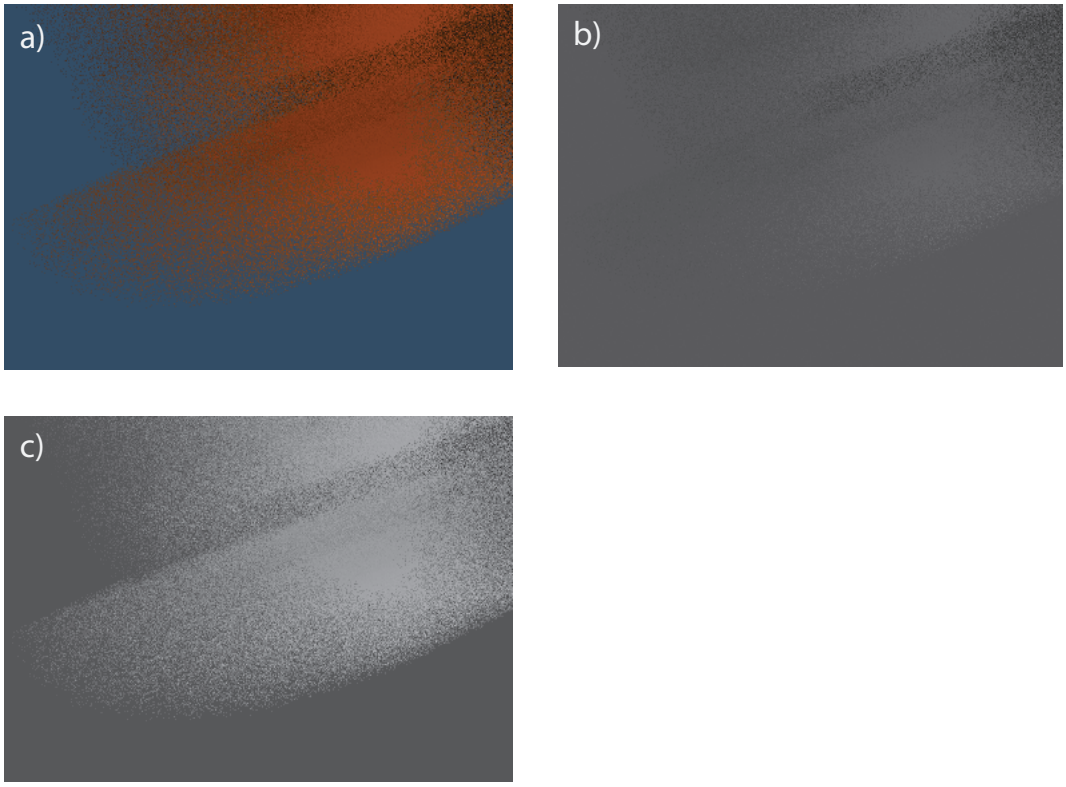


Figure 5.1: Notice how the color difference between the object and the background is much more perceptible in the color figure a) than in the monochrome conversion b), as is the error resulting from stochastic rasterization with a low sample count. In the red color channel represented in figure c), the difference is very perceptible.

$$A[i] - B[i] = \sqrt{(A_r[i] - B_r[i])^2 + (A_g[i] - B_g[i])^2 + (A_b[i] - B_b[i])^2}. \tag{5.18}$$

A and B are now RGB images, where the letter in subscript denotes the channel red, green or blue. BPMSE is calculated from this definition of per-pixel difference by using equations (5.14)-(5.17) as before, only with band-pass filtering applied separately to each color channel.

Unfortunately, this simple method of comparing RGB colors can not be directly applied to SSIM-based metrics, and developing a conceptually sound VDP variant based on it would also be challenging. One possible strategy would be to take SSIM or VDP measurements independently from each color channel and sum these together, but there is no existing research on this. So, BPMSE is the only metric included in our experiments which takes color differences into account. Of course, this does not affect measurements of images that are monochrome to begin with.

5.5 Quantization Error

To display images on a digital display device, the red, green or blue color values need to be quantized to n-bit integers. It is useful to know the PSNR resulting from such quantization. This PSNR can be used as a point of comparison for approximation errors, especially when the aim is to create high-quality images. When the approximation errors are smaller than the quantization error, they practically do not affect the displayed image.

Quantizing uniformly distributed real numbers to integers results in an uniformly distributed error between $-\frac{1}{2}$ and $\frac{1}{2}$. From this we can calculate the MSE for the quantization error by integrating over the range of possible error values. This theoretical value for quantization MSE or QMSE is given by

$$\text{QMSE} = \int_{-\frac{1}{2}}^{\frac{1}{2}} x^2 dx. \tag{5.19}$$

This gives us an absolute value of $\frac{1}{12}$, or a relative value of $\frac{1}{3\ 060}$ if we apply it to the common 8-bit quantization normalized to the [0, 1] range. From this, we can also calculate the peak signal-to-quantization-noise ratio or PSQNR, using the formula

$$\text{PSQNR} = 10 \log_{10} \frac{(2^b - 1)^2}{\text{QMSE}}, \tag{5.20}$$

where b is the quantization bit depth. For 8-bit quantization, this formula gives us the value of $10 \log_{10}(780\ 300)$, or approximately 58.9226 dB.

6. MEASURING IMAGE QUALITY OF STOCHASTIC RASTERIZATION

6.1 Test Scenes

We developed a set of 16 abstract test scenes to measure image quality resulting from different sampling schemes. The abstract scenes were designed to isolate specific characteristics of the sampling patterns and reveal some of their flaws, but they also remained reasonably close to real content so that they would not introduce irrelevant types of errors. The abstract scenes can be divided into three sets based on their contents.

We will call the first two abstract scenes the *gradient scenes*. They are meant to produce a linear gradient due to motion blur or the combination of motion blur and defocus blur. These scenes are intended to reveal especially how the sampling pattern is distributed in the t dimension.

We will call the second set of abstract scenes the *horizon scenes*. This set of abstract scenes contains a checkerboard perpendicular to the image plane and varying kinds of blur. These are meant to reveal subtle regularities in the sampling patterns and also determine their behavior with images that have varying edge densities and orientations. One of these test scenes is rendered entirely without blur to determine the spatial anti-aliasing quality of the sampling pattern.

We will call the third set of abstract scenes the *other scenes*. These contain defocus blurred and motion blurred quads facing the camera, some with a checkerboard texture. The ones with defocus blur are meant to reveal especially how the sampling pattern is distributed in the u and v dimensions. The higher-frequency checkerboard patterns are interesting, since they can reveal regularities in the sampling patterns.

In addition to the abstract scenes, four natural scenes were used. The scenes are not natural in the sense that they would be completely realistic, but they combine different shapes and textures so that images resulting from rendering them share the characteristics of natural images. Most importantly, they are similar to the photographic material that perceptual image metrics have traditionally been evaluated with. The natural scenes included Fairy Forest featuring defocus blur and three variations of a speeding Mustang sports car featuring different kinds of blur.

A complete illustrated list of the test scenes is found from Appendix A.

6.2 Rasterizer Software And Configuration

The stochastic rasterizer software we used was designed to simulate hardware-friendly sampling and reconstruction schemes. As such it implements uniform sample density across pixels and a simple box reconstruction filter in x, y, and t. Each sample is weighted equally, and no samples are shared between different pixels. These are reasonable limitations for a hardware renderer, and were found to be acceptable for our purposes.

We ran our main series of experiments with the following sampling schemes.

1. *Random sampling.* x, y, t, u, and v coordinates were randomized independent of each other.

2. *Padded stratified sampling.* x, y, t, u, and v were stratified efficiently for each pixel using padded stratification as described in Section 3.10.

3. *Latin hypercube sampling.* x, y, t, u, and v were stratified separately for each pixel, as described in Section 3.11.

4. *Best-candidate sampling.* A sampling pattern approximating a Poisson disk pattern was generated using an Euclidean distance measure in all of the 5 dimensions. The Euclidean distance measure was taken from a normalized sampling space where the t, u, and v values were in the [0, 1] interval, and pixel centers were 1 unit apart in the x and y dimensions. The algorithm generated an equal number of sample positions for each pixel neighborhood by restricting the candidates to a randomly selected pixel neighborhood on each iteration. 64 candidates were generated for each sample position.

5. *Best-candidate sampling (non-Euclidean).* A sampling pattern was generated using the distance measure given in Equation (3.14). Otherwise, the sampling scheme implementation was identical to the best-candidate sampling.

6. *Hardware-friendly low-discrepancy sampling*, or HWLDS from now on. t, u, and v coordinates were generated with Sobol matrices using techniques given in Section 3.13. Coordinates for x and y dimensions were equal to the sample coordinates used for anti-aliasing in the Nvidia Fermi GPU architecture when there were at most 16 samples per pixel, and generated using the Hammersley sequence when there were more. The x, y sequence was the same for all pixels.

All patterns used the concentric-square mapping square-to-disk method described in Section 3.8 to transform the square UV coordinate distribution into the unit disk. All patterns except the random sampling repeated in $N \times N$ pixel tiles so that they could be cached for efficiency. The width of the tile N was chosen according to the

sample count so that the tiling would not noticeably affect the perceptual quality. The implementations of all sampling patterns were verified manually by inspecting all of their 2D projections.

The Mersenne Twister algorithm was used as the source of pseudo-random numbers for the random, stratified, jittered, and best-candidate sampling schemes. The period of the algorithm is magnitudes larger than the total number of samples in any of the generated images. The same pseudo-random sequence was used to generate the required numbers for all of the samples.

The software implemented per-pixel transparency with stochastic transparency, which is visible as some additional error in the Fairy Forest scene. The threshold values for stochastic transparency were uniform random numbers for all patterns except for HWLDS, which generated them from a low-discrepancy sequence. The values were scrambled by bitwise XORing them with polygon IDs to avoid correlation between polygons.

All images were rendered as 32-bit floating point linear RGB bitmaps.

6.3 High-quality Reference Images

Perfect reference images are impossible to acquire in the general case, so reference images were rendered with the padded stratified sampling pattern introduced in Section 3.10. The simple box reconstruction filter was used also for the reference images. The box filter does not yield optimal image quality, but it ensures that reconstruction does not affect the comparisons between the sampling schemes.

$2^{14} = 16\,384$ samples per pixel were used to render the reference images. This gives 128 x, y, u, and v strata and 16 384 t strata for each pixel. PSNRs between images generated with two different random seeds with this sampling scheme were computed to give an indication of reference image quality. These PSNR values are given in Table 6.1, and were reasonably close or above the PSQNR of 8-bit quantization defined in Section 5.5 in almost all cases.

Since the error manifests as mostly white noise, this means that the differences between the two reference images would be virtually invisible on an ordinary computer monitor. Manual inspection reveals that reference images for scenes 2 and 11 still have a small amount of visible noise, but the level is low enough that it should not significantly skew the measurement results.

6.4 Artifact Classification

Roughly six types of visible artifacts could be found in the rendered images. Understanding the causes of these artifacts is important for assessing sampling pattern quality.

	Scene	PSNR (dB)
1	Blank billboard in motion	71.87
2	Blank billboard in motion and defocus	48.38
3	Blank billboard in 1x defocus	68.22
4	Blank billboard in 2x defocus	66.89
5	Blank billboard in 3x defocus	65.66
6	Blank billboard with camera zooming	70.22
7	8x8 checkerboarded billboard with camera zooming	61.54
8	8x8 checkerboarded billboard in defocus	60.46
9	16x16 checkerboarded billboard in defocus	56.97
10	32x32 checkerboarded billboard in defocus	53.94
11	64x64 checkerboarded billboard in defocus	51.06
12	Checkerboard in partial defocus	53.10
13	Checkerboard in partial defocus with texture filtering	59.03
14	Checkerboard with high amount of parallel motion	53.87
15	Checkerboard with low amount of parallel motion	54.31
16	Checkerboard in focus	69.67
17	Fairy Forest	66.76
18	Mustang in defocus and motion	63.50
19	Mustang in motion	71.05
20	Mustang in defocus	71.21

Table 6.1: PSNRs between two reference images generated with different random seeds. The average PSQNR from 8-bit quantization is approximately 58.92 dB.

High-frequency noise. This error type was the most expected. Most of the stochastic rasterization approximation errors manifested as high-frequency uniform noise with the appearance of random noise. This noise was not visually very distracting save for the lowest sample density levels. It can be effectively filtered out in reconstruction. A successful image quality metric should deemphasize this artifact compared to the other, more severe artifacts.

Structured medium-frequency artifacts. Lower-frequency noise with some regular structure appeared in some of the checkerboard scenes rendered with the pseudo-random sampling patterns. This was also to be expected, though the strength of the noise could be surprisingly high. This type of noise is visually very distracting, and it should be easily caught by all of the perceptual image quality metrics we surveyed.

Spatial aliasing. Spatial aliasing was easily apparent in the in-focus checkerboard scene with those low-density sampling patterns that had the same relative x and y sample coordinates for each pixel. Spatial aliasing manifested as jagged edges and *Moiré* patterns near the horizon.

Tiling. Most of the sample patterns we used repeated periodically in the x and

y dimensions. The tiling period was made sufficiently large for each sample density level so it would not noticeably affect the perceptual quality. Still, in some cases careful examination could reveal that repeating sampling patterns had been used. This artifact can not be detected by those metrics that operate on a small window of the image at a time, such as SSIM-based metrics. Image quality metrics based on Fourier analysis obviously have an advantage in detecting this.

Ideally, a production system would not use a repeating sampling pattern but would be able to generate as many unique sample positions as are needed on the fly. This effectively rules out slow sampling pattern generation methods. For real-time applications, fast sampling pattern generation could be done by implementing some low-discrepancy sequence in hardware.

Perceived change of brightness. This was especially apparent in the black and white scenes, where images rendered with fewer samples per pixel commonly appeared brighter. This can be due to luminance nonlinearity of the display device and to a smaller extent the luminance nonlinearity of the human vision. This effect was much greater than what we expected, but it could be mostly countered by applying gamma correction to the images in software. This highlights the importance of color correction for stochastic rasterization.

Perceived banding. Surprisingly strong banding artifacts could be seen in some of the images approximating smooth gradients. This was not caused by any kind of bias in the actual color values, but was found to be an effect of background-dependent contrast. Bright dots on a darker background are perceived to be brighter than they are. When there is a large number of bright dots on an uniform background, the edge of such a region of dots is perceived as structure in the image. This effect could not be fixed by gamma correction.

These artifacts appeared in the images of black and white scenes rendered with sampling patterns that had per-pixel stratification. In these cases, pixels can only get two distinct color values at any single point of a uniform gradient created by a moving edge in the stratified dimension. This creates large regions where there are a number of single-colored brighter dots on a single-colored background.

The banding is an interesting artifact, because it suggests that there can be a downside to perfect stratification in the sampling pattern. However, images that had the banding artifacts still had subjectively better perceptual quality than the images rendered with the random sampling pattern, and the banding artifacts ceased to be disturbing around the sample density of 16 samples per pixel. Better reconstruction filtering could also lessen the importance of this artifact.

6.5 Comparison With Dithering

Rendering an image of a black-and-white scene with a 1 sample reconstruction filter results in an image with only black (0) or white (1) pixels. If n samples are equally weighted to reconstruct a pixel, there are $n + 1$ possible distinct grey levels, since either 0, 1, ... or n samples can be evaluated as white. These $n + 1$ grey levels are evenly distributed in the range [0, 1].

Based on this information on the grey levels, additional references can be generated for the black-and-white scenes by applying dithering to the high-quality reference image. Since the dithered images have similar characteristics as the images that have been rasterized with low sample counts, they can be used to parametrize and evaluate image quality metrics for stochastic rasterization.

We used Ostromoukhov's high-quality error-diffusion dithering method to generate these additional reference images. The method operates with an optimized filter kernel for each 8-bit grey level [39]. The method was applied to floating point bitmaps so that each floating point color value was rounded to the closest 8-bit integer to choose the filter kernel.

Error-diffusion dithering can result in certain kinds of undesirable artifacts. The common Floyd-Steinberg method is prone to structural artifacts at specific grey levels [39] and top edges of horizontal gradients. See Figure 6.1a for an example of Floyd-Steinberg dithering.

Ostromoukhov's method was created specifically to produce good output at the problematic grey levels [39], but it is also prone to slight artifacts at the top edges of horizontal gradients. Images containing mostly horizontal gradients were transposed prior to dithering to avoid these artifacts. Pixels were processed in serpentine order. See Figure 6.1b for an example of Ostromoukhov dithering.

Dithering to more than 2 grey levels with the error diffusion methods was found to result in similar banding artifacts as stratified sampling. These artifacts were mitigated by adding random noise to the images prior to dithering. Noise with the amplitude of $\frac{1}{3}d$, where d is the difference between two consecutive grey levels, was experimentally determined to produce good output.

The idea to add random noise was derived from an improved version of Ostromoukhov's dithering method that used random threshold modulation to shape the spectrum of the dithered images [57]. Noise was not added to pixels that were entirely black or white in the reference image. See Figures 6.1c and 6.1d for an example of this.

It was found that the dithered images often had noticeably higher perceptual quality than the images rasterized with the best known sampling patterns. Sampling has the inherent disadvantage that it does not know the shape of the original image

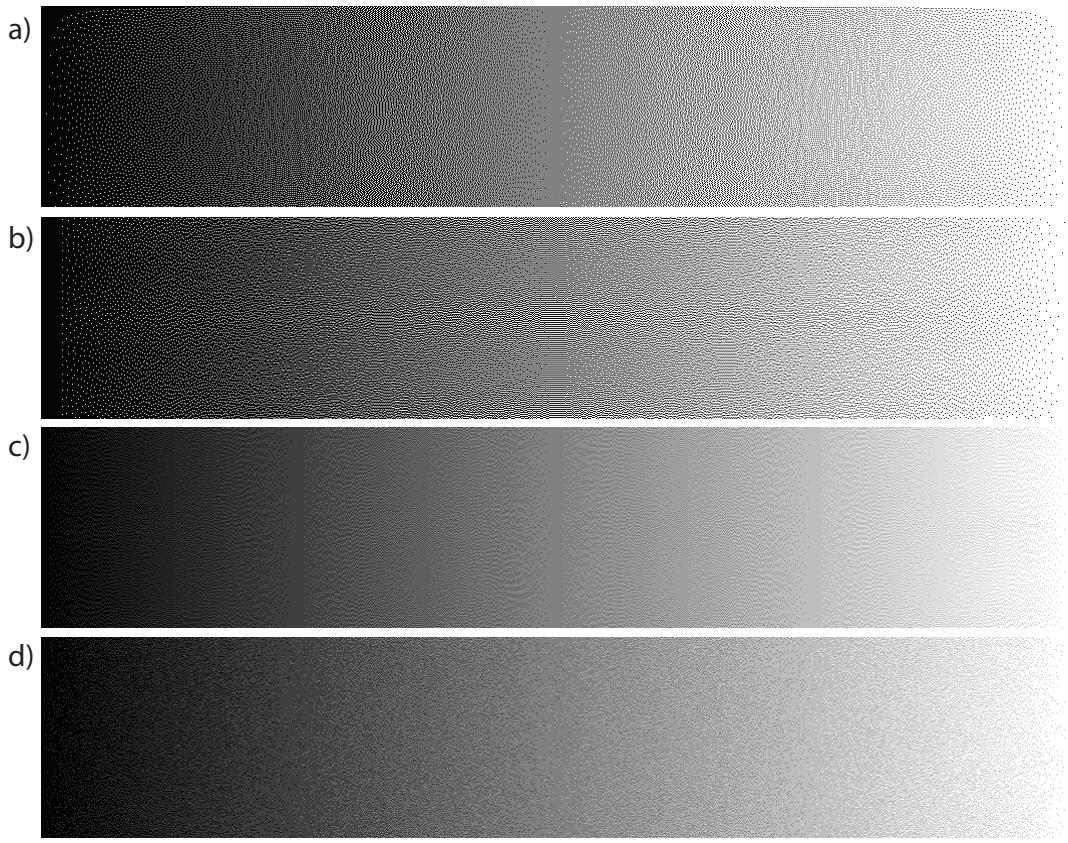


Figure 6.1: a) Floyd-Steinberg dithering of a horizontal gradient demonstrating the problematic grey levels and the artifacts near the corners of the top edge. b) Ostromoukhov dithering of the same gradient, transposed to avoid the edge artifacts. c) Ostromoukhov dithering to 5 grey levels. Notice the banding artifacts. d) Ostromoukhov dithering to 5 grey levels with added noise. The dithering is not as accurate as the one above, but the undesirable banding is mostly gone.

function in advance, so results from optimal dithering are bound to be better than results from optimal sampling. Thus the improvement potential of the sampling patterns cannot be judged only based on this, even though it is always possible to construct an image-specific sampling pattern that results in the exact same image as the dithering.

6.6 Parametrizing Existing Metrics

We compared the new BPMSE metric against Matlab implementations of two other perceptual image quality metrics: HDR-VDP2 [30] and MS-SSIM [23]. In addition, we did this comparison with the original MSSIM [52], though we expected the similar MS-SSIM to simply outperform it in all cases. All of these require some parameters to be chosen when they are used.

The HDR-VDP2 Matlab implementation was used with the default recommended viewing distance setting of 30 pixels per degree, which corresponds to a typical view-

ing distance for a 72 dpi computer monitor. The *luminance* color encoding setting was used with all images, and color images were converted to monochrome luminance maps prior to measurement. The RGB color encoding settings that the implementation offers exist only for convenience, and the metric still internally operates only on monochrome images.

The original MSSIM suggests taking viewing distance into account by downsampling the images prior to measurement. It is suggested that all images are downsampled to approximately 256 times 256 pixels [52], but this stems from the assumption that the viewing distance is a few times higher than the physical width of the image. With fullscreen viewing on a computer monitor the viewing distance is usually shorter. Thus our test images were only downsampled to the width of 512 pixels, or half of their original width prior to measurement. This has a similar effect as increasing the sampling rate to 4 times its original value.

The MS-SSIM metric was used with default parametrization. This means 5 different levels of low-pass filtering weighted with the weight vector given in Table 6.2. The default stabilizing constants $K_1 = 0.01$ and $K_2 = 0.03$ and the default Gaussian window filter were used for both MSSIM and MS-SSIM.

MS-SSIM scale	1	2	3	4	5
Weight	0.0448	0.2856	0.3001	0.2363	0.1333

Table 6.2: The default weights for MS-SSIM.

6.7 BPMSE Weights

There does not exist a database of subjective quality measurements for stochastic rasterization, and creation of such a database is a very time-consuming effort. For this reason, we derived the BPMSE weight vector W by maximizing correlation with leading contemporary metrics, MS-SSIM and HDR-VDP2. The sum of Spearman rank correlation coefficients with MS-SSIM and HDR-VDP2 was used as the fitness measure. The correlation with HDR-VDP2 was given five times the weight of correlation with MS-SSIM, since MS-SSIM was determined not to perform as well on noise artifacts.

We used all of our stochastic rasterization quality measurements for calculating the correlation coefficients. Since correlation between RGB measurements using BPMSE and monochrome measurements using MS-SSIM and HDR-VDP2 is not meaningful, we used monochrome conversions of the RGB scenes for computing the BPMSE values for these correlations.

Our first approach was to derive the weight vector from Mannos' CSF model

detailed in Section 4.6. This approach has the drawback that the CSF is not completely appropriate for assessing contrast perceptibility when the contrast is sufficiently above the visibility threshold [54], but it turned out to yield good results in practice. We used the same 30 pixels per degree viewing distance parameter as we used with HDR-VDP2, and chose the CSF peak frequency parameter for the model between 3.0 and 5.0 degrees by testing peak frequencies with increments of 0.01 and choosing the one which resulted in the highest fitness measure.

We included six FRMSE values in each BPMSE measurement, so that the largest filter had a standard deviation of 16 pixels. The resulting peak frequency for the CSF model was 3.38 degrees, and the absolute Spearman rank correlation coefficients with MS-SSIM and HDR-VDP2 were 0.9447 and 0.9870, respectively. These can be considered good results.

The BPMSE values are plotted against MS-SSIM and HDR-VDP2 values in Figure 6.2. It can be seen that the correlation with HDR-VDP2 is good overall. The overall correlation with MS-SSIM is not as good, which was to be expected. However, when the MS-SSIM results are examined scene-by-scene, they show much better correlation with BPMSE and HDR-VDP2. It looks like MS-SSIM does have a relationship to perceptual quality even in the case of stochastic rasterization noise, but the MS-SSIM values can not be compared across different scenes.

In addition to this approach, we used a simple hill-climbing evolutionary algorithm to seek better correlation with weights that were different from the CSF model. The algorithm added random variation to all the weights simultaneously and tested if the changes would improve the fitness measure. To avoid only hitting the local maximum, the amount of random variation added was varied between iterations.

The evolutionary method sometimes yielded erratically varying weights, and even the very best set of weights we found put suspiciously low weight on the second lowest band-pass frequency. We suppose that this is because of the uneven distribution of rendering artifacts among frequency levels in our set of samples. For this reason, the universal applicability of weights derived using this evolutionary approach is questionable.

FRMSE level k	1	2	3	4	5	6
CSF-derived weights	0.00104	0.0441	0.203	0.308	0.268	0.177
Evolved weights	0.000223	0.0552	0.0636	0.154	0.00884	0.718

Table 6.3: Weights for BPMSE. We chose to use the CSF-derived weights.

The best absolute Spearman rank correlation coefficients with MS-SSIM and HDR-VDP2 found using the evolutionary approach were approximately 0.9451 and 0.9872, respectively. Since these are just marginally better than the results for the

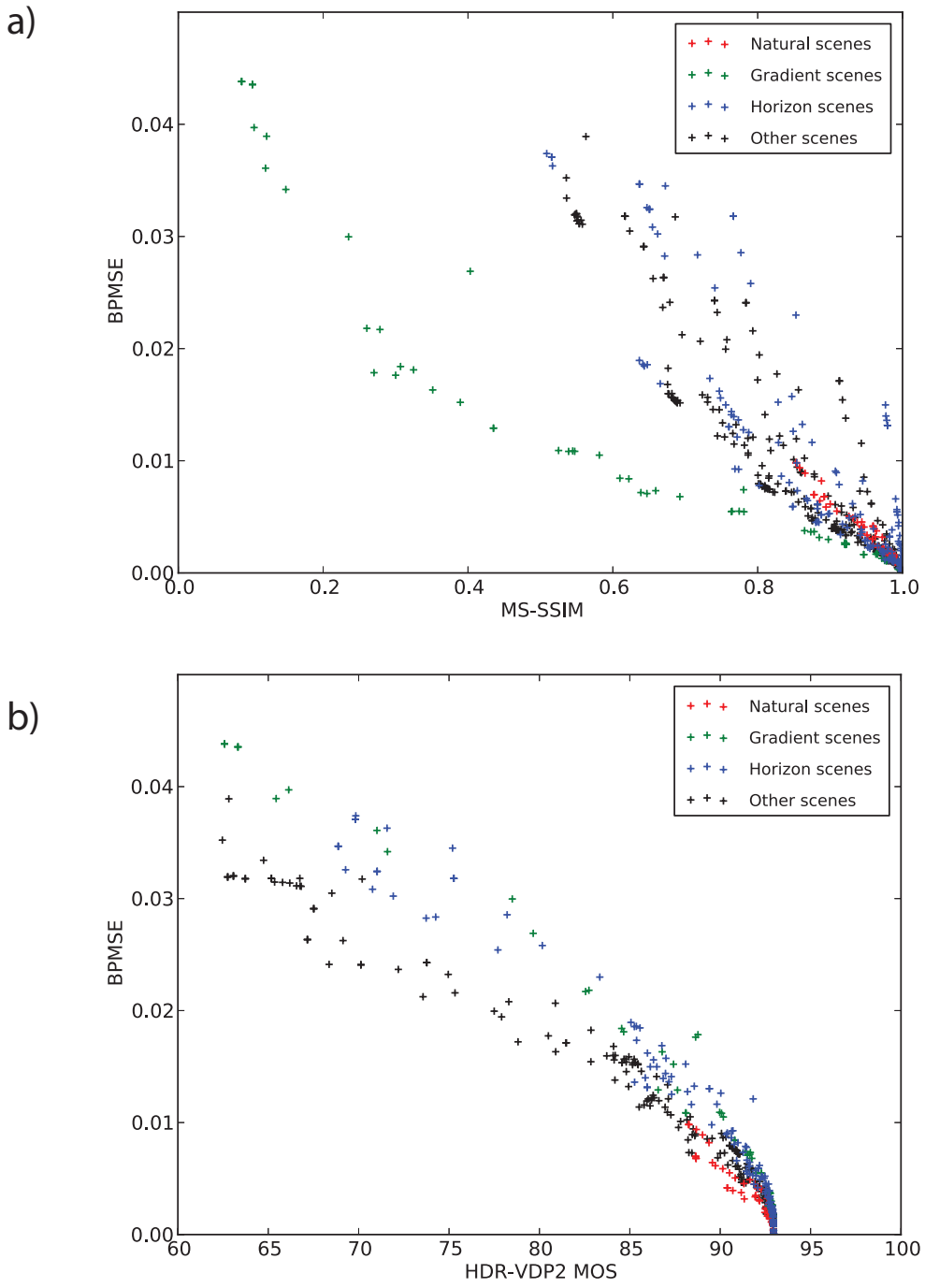


Figure 6.2: a) BPMSE measurements plotted against MS-SSIM measurements. b) BPMSE measurements plotted against HDR-VDP2 mean opinion score predictions.

weights derived from the CSF model and we have reason to suspect the validity of these weights, the original weights from the CSF model were chosen to be used. The weights were normalized so that their sum is 1 and then rounded to three significant

digits. The weights from the CSF model and the weights resulting from evolving them can be found in Table 6.3.

6.8 Measurement Results for the Artificial Scenes

There was a very clear pattern to the typical quality measurements for any single scene. The HWLDS pattern, which has good co-operation between samples for different pixels, dominated at lower sample densities. It was followed by the Latin hypercube and the padded stratified sampling patterns, which were equal to the random pattern at 1 sample per pixel, but caught on to the HWLDS at higher sample densities. HWLDS proved worse than the Latin hypercube and padded stratified sampling patterns only in a few scenes where using it resulted in distracting structured artifacts.

The best-candidate patterns had co-operation between different pixels, but it was not quite as effective as with the HWLDS pattern. The pattern with the non-Euclidean distance measure was clearly better than the ordinary Poisson-disk pattern, which was to be expected. However, it still failed to match the stratified pattern in quality at higher sample densities. This confirms the assumption that it is better to generate distributions separately for the spatial and aperture domains, though this complicates achieving co-operation between pixels.

The random pattern performed the worst in but a few scenes, where it occasionally surpassed the HWLDS exhibiting structured artifacts. As expected, it always performed worse than the Latin hypercube and padded stratified sampling patterns. Overall, the BPMSE measurement results were found to be well in line with subjective quality. The typical pattern seen in the measurement results is illustrated in Figure 6.3a.

The worst structured artifacts could be found from the checkerboard scenes with defocus blur. An example of the HWLDS image compared to the image from padded stratified sampling is shown in Figure 6.3d.

Some of the most interesting results were measured from the checkerboard scene with small amount of parallel motion. Here, some structured artifacts appeared near the horizon when using the HWLDS scheme even at high sample densities. It seems like some systematic flaw in the sampling pattern prevented the rendered image from converging towards the correct result even as the sample density was increased. The measurement results from this scene can be found from Figure 6.3b, and the artifacts from Figure 6.3c.

The horizon scene with defocus blur had earlier revealed a flaw in the earlier version of the HWLDS implementation, which did not correctly permute the sample locations in the x, y dimensions with high sample counts. This flaw was then fixed to make the measurements. It seems like this class of scenes is generally very good

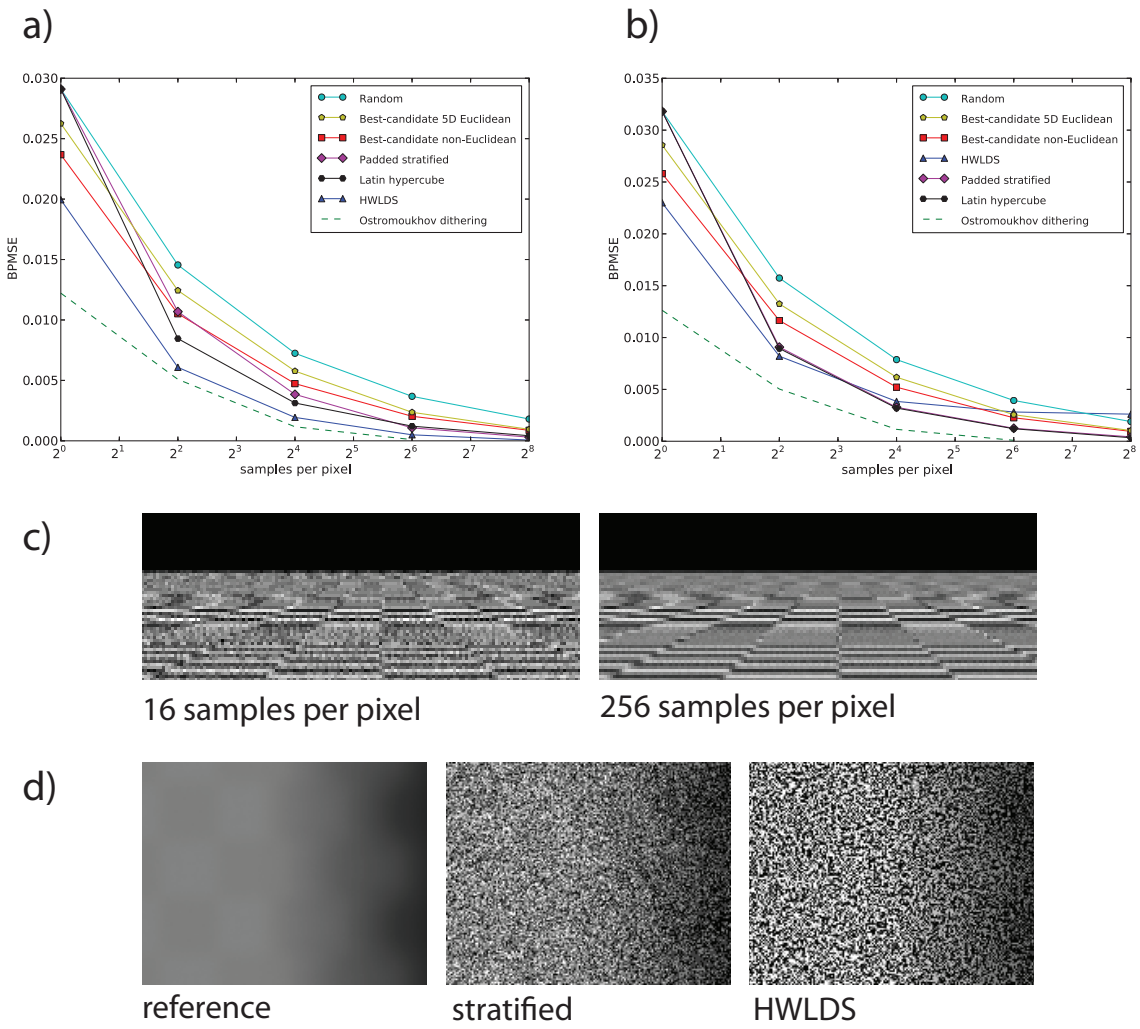


Figure 6.3: a) The typical pattern found from the measurement results. b) In this scene (checkerboard with low amount of parallel motion), renderings using HWLDS failed to converge towards the correct result even as the sample density was increased. Results from the padded stratified and Latin hypercube patterns are practically identical, since the error comes mostly from motion blur. c) The artifacts seen near the horizon which did not disappear as the sample density of HWLDS was increased. The images are from 16 SPP and 256 SPP renderings. d) Structured artifacts from 4 SPP HWLDS compared to random noise from 4 SPP padded stratified sampling and the reference image from the 16×16 checkerboard in defocus scene.

for revealing errors from the sampling patterns. This should come as no surprise, since the scenes contain patterns with a wide range of frequencies with respect to the x, y dimensions. In the presence of motion and defocus, these patterns also interact with each other.

From these results we can determine that even slight structure in the sampling pattern can result in visible flaws in some worst-case situations. The scenes we measured were not too far off from content in real-life applications, so some of these

worst-case situations are bound to be found in real-life applications. The HWLDS sampling pattern exhibited generally very good results, but it would need further refinement to make it robust enough for production use.

The difference between Latin hypercube and padded stratified sampling schemes was not usually very large. Generally speaking, the Latin hypercube scheme performed better with lower sample densities and showed superior spatial anti-aliasing, but the padded stratified scheme often surpassed it at 256 samples per pixel. This suggests that the padded stratified sampling scheme would generally be a better choice for offline rendering.

Due to the amount of computational resources needed, we did not duplicate the measurements with images rendered with different random seeds, so we are not able to give error estimates for our results. However, manual inspection of the quality of images rendered with different instances of the random pattern did not reveal significant differences, and the regularity of our results suggests that there is no significant error.

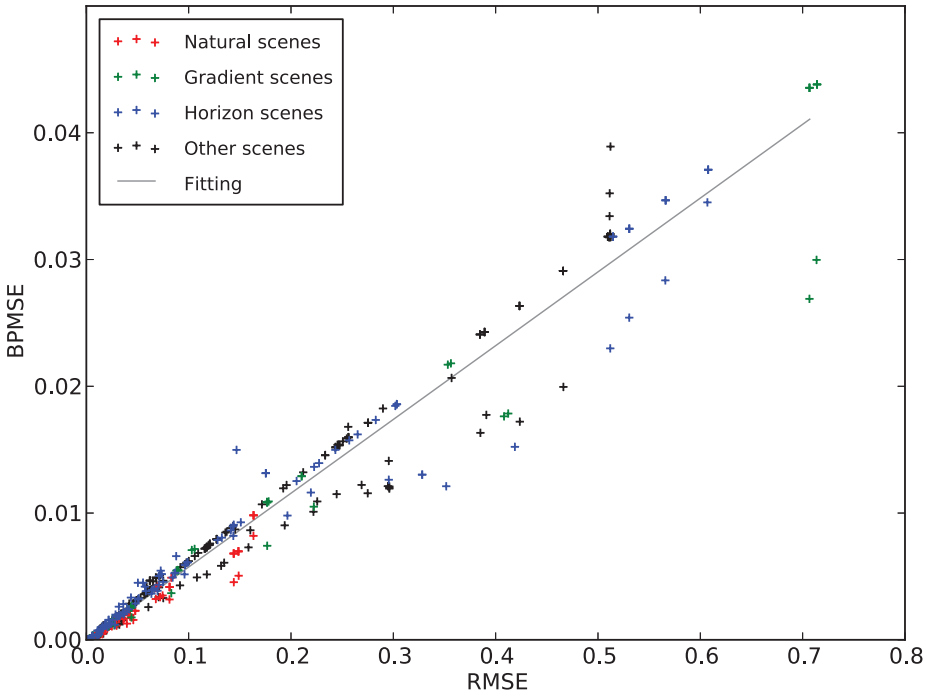


Figure 6.4: BPMSE-RMSE correlation. In the results of our measurements, there are many cases where RMSE significantly overestimates the perceptual error as measured by BPMSE, and some cases where it underestimates it.

The value that BPMSE adds over RMSE is evident from some outliers in the BPMSE-RMSE correlation. RMSE tends to underestimate perceptual error in some

cases where there was significant structured error, such as in the checkerboard scenes rendered with the HWLDS pattern with low sample density. It would be easy to construct more artificial examples where RMSE underestimates the perceptual error. On the other hand, RMSE overestimates perceptual error in the images where there is good co-operation between pixels, especially images rendered with HWLDS at 1 SPP and corresponding dithered images. BPMSE results are plotted against RMSE results in Figure 6.4.

6.9 Measurement Results for the Natural Scenes

The measurement results for the natural scenes were generally in line with those for the artificial scenes. HWLDS dominated at lower sample densities, but the Latin hypercube and the padded stratified patterns caught on at higher sample densities.

The degree of error in the natural scenes was generally lower than in the artificial scenes. This was to be expected, since the natural scenes were not specifically designed to reveal errors in the sampling patterns, and the resulting images have large areas with little or no blur.

It is notable that both MS-SSIM and HDR-VDP2 struggled to quantify quality differences between 16 and 256 samples per pixel in some of the natural scenes. They unavoidably underestimated the errors due to only operating on single-channel images, and HDR-VDP2 likely classified some of them as sub-threshold. BPMSE operating on RGB images could more easily detect these smaller errors, giving more fine-grained information on the image quality. BPMSE measurements taken directly from the RGB images were often 1.5 times as high as BPMSE measurements taken from their single-channel conversions.

6.10 Ranking Sampling Schemes by Perceptual Quality

We have shown that we are able to automatically assess image quality produced by different sampling schemes with good accuracy. However, this does not yet give us a way to assign a quality score to a sampling scheme, since we do not have a well-defined method to summarize the image quality measurements across different scenes with a single numerical value. In this section, we will consider such methods and also compare them with sampling pattern discrepancy measurements.

The BPMSE values are obviously scene-dependent — the measurements we showed in the preceding chapters for different scenes followed the same general pattern, but the overall extent of the error varied depending on how much blurring and how much contrasting patterns the scene contained. To compensate for this, we can normalize a series of measurements from a scene by scaling them to cover a range from 0 to 1. We do this by using

$$\mathrm{BPMSE}_n(A,B) = \frac{\mathrm{BPMSE}(A,B) - min}{max - min + C}, \tag{6.1}$$

where C is a small stabilizing constant, min is the minimum measurement from the given scene and max is the maximum measurement from the given scene. This makes the measurements from different scenes comparable to each other.

Defining a representative set of test scenes is not as simple. Our artificial test scenes clearly separate different aspects of the sampling patterns from each other, but this occurs to some extent also with the natural scenes we are targeting. Judging which aspects are most important is not trivial. We generally want good average performance out of the sampling pattern, but on the other hand would like to minimize worst-case error that actually manifests with the targeted scenes.

In the end, manually inspecting BPMSE results across different scenes is the only way to get a complete understanding of how the sampling pattern behaves. This is still less time-consuming than subjectively assessing each image individually. Additionally, a simple average over the normalized results from some representative set of scenes can be used as an overall quality measure, but the averaged results might hide important errors in specific scenes.

Discrepancy still remains a widely used tool to measure sampling pattern quality, even if it has its shortcomings discussed in Section 3.5. Star discrepancy can be measured from an arbitrarily sized tile in the image plane. We measured how star discrepancy measurements from tiles with different widths compare to average normalized BPMSE across our entire set of test scenes.

We tested image plane tiles sized 1×1, 2×2, 3×3, and 4×4 pixels. To avoid random variability, 9 tiles of pixels were chosen inside the sampling pattern and the final discrepancy value was averaged from their approximate measurements. We measured the Spearman rank correlation coefficient, Kendall rank correlation coefficient, and maximum Pearson correlation coefficient from a set of different fitting functions between the star discrepancy measurements and the average normalized BPMSE measurements. These correlation coefficients are plotted against the tile width used in measuring the discrepancy in Figure 6.5a.

The correlation is the highest when the star discrepancy is measured from 2×2 pixel tiles. This is understandable, since it takes co-operation between pixels into account. Measuring from a larger tile than 2×2 pixels results in no additional benefit in this sense, and the discrepancy measures become more similar for different patterns, which increases the potential for error. The relationship between discrepancy and average BPMSE is also closest to linear with 2×2 pixel tiles. So, we recommend that star discrepancy measurements should be taken from approximately 2×2 pixel tiles.

That is not to say that star discrepancy in five dimensions should necessarily be

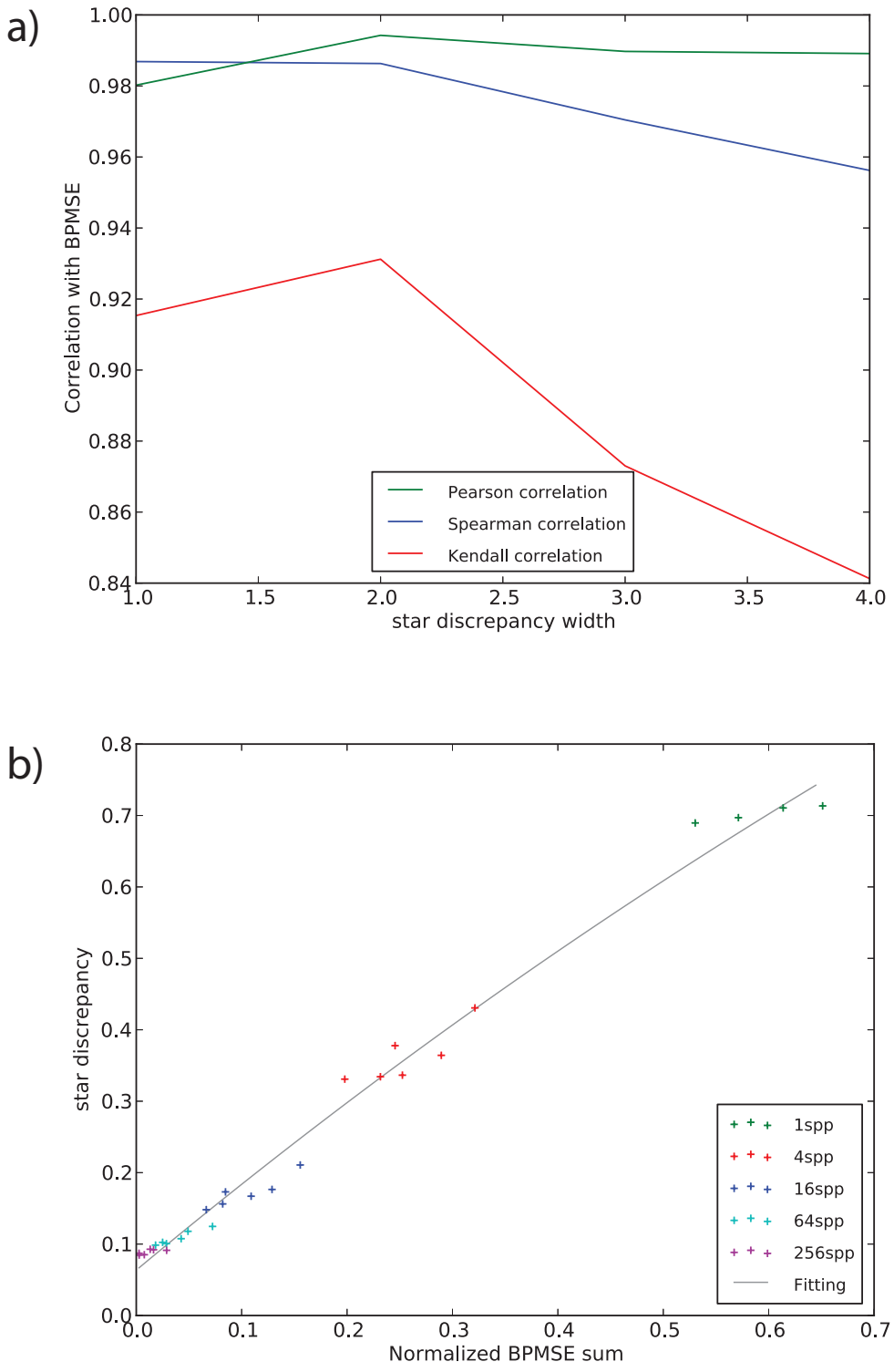


Figure 6.5: a) Correlation coefficients between average normalized BPMSE measurements and star discrepancy measurements taken from different-sized pixel tiles. b) Star discrepancy measurements from 2×2 pixel tiles plotted against average normalized BPMSE measurements.

used at all. The correlation coefficients we got overestimate its ability to predict which sampling scheme performs the best, as large differences in discrepancy are seen between sample densities, but not as much between different sampling patterns at the same sample density. To illustrate this, our average normalized image quality measurements are plotted against 2×2 pixel tile star discrepancy in Figure 6.5b.

7. CONCLUSIONS AND FURTHER RESEARCH

The BPMSE metric met the goals we set for it. The metric is conceptually simple, efficient to compute compared to leading perceptual metrics, and reflects the actual image quality of images rendered using stochastic rasterization better than simple RMSE. The simplicity of BPMSE is enabled by the realization that masking phenomena do not affect worst-case perceptual quality of stochastic rasterization.

Our measurement results from images rendered with different sampling schemes are mostly in line with earlier results. Using low-discrepancy sequences is usually very effective, but they can be prone to structured artifacts. The artifacts could be surprisingly severe in some cases that we would expect to manifest also in real-world content. Padded stratified sampling and Latin hypercube sampling performed relatively well while avoiding these artifacts, but they can not be implemented as efficiently. Best-candidate sampling schemes generated in 5D space could achieve some co-operation between pixels, but clearly lost to per-pixel padded stratified and Latin hypercube schemes on higher sample counts.

There seem to be no shortcuts to assigning an overall quality score for a sampling scheme even if we have a very good image quality metric in our disposal. The quality should always be evaluated in the context of the desired application. Sampling pattern discrepancy does not tell the whole truth of the resulting image quality either, though star discrepancy computed from 2×2 pixel tiles of the sampling pattern achieved reasonable correlation with average image quality. The test scenes we used provide a reasonable basis for assessing different basic characteristics of the sampling patterns, but the comprehensiveness of the test scene set is by no means proven.

There are many opportunities for further research in the area. Constructing a database of subjective image quality measurements of rendered images would ease evaluating the performance of perceptual metrics for rendering. Existing widely used databases rely almost solely on photographic images with artificially added corruption.

Investigating the possibility of a no-reference image quality metric for rendered images would also be interesting. In stochastic rasterization, error typically manifests as noise at specific frequencies, and in some cases it could be isolated from the image without having a reference. This is the case especially with the kind of

artificial test scenes we mostly used in our measurements, where there is no high-frequency texture that could be misinterpreted as noise.

Sampling patterns could also be optimized using BPMSE or other perceptual metrics. Potential optimization approaches include evolving Latin hypercube permutations or evolving parameters for generating low-discrepancy sequences.

We also left out comparisons of different low-pass and reconstruction filters, and opted to use the simple box filter for all of the rendered images. Better reference images could be generated by choosing an appropriate high-quality filter. This would enable measuring the effects of different filters also with lower sample densities.

We would expect the rasterization algorithms to be predominantly used to render video, and measuring video quality would be yet another research opportunity. We would expect the time dimension behavior of the sampling pattern to become more important in the case of video, and some artifacts in the bitmap frames could appear more or less severe in the presence of motion.

As a closing note, it was surprising to note how temporal reconstruction is usually left to so little attention in the literature. Most research just assumes that the cinematographic image function is used. Even if the shutter closing and opening is accounted for, this corresponds roughly to a box reconstruction filter in t. In terms of sampling theory, the box filter is suboptimal. The choice of filter might be a practical consideration in interactive applications and a stylistic choice in the case of offline rendered animation, but most of the research fails to convey a deep understanding of temporal reconstruction. The formulae are often presented as if integrating over t would not have a perfectly valid signal processing interpretation.

We found only one practical introduction to temporal anti-aliasing presented in [17]. This formulation of the reconstruction with respect to time presented in Chapter 3 can be used whether the aim is to emulate cinematography or to reconstruct the original time-dependent image function as accurately as possible. Especially 3D games and stereoscopic animated movies which try to immerse the viewer into the scene could benefit from bypassing the camera analogy.

REFERENCES

[1] Tomas Akenine-Möller and Eric Haines. *Real-Time Rendering*. A. K. Peters, Ltd., Natick, MA, USA, 2nd edition, 2002.

[2] Tomas Akenine-Möller, Eric Haines, and Naty Hoffman. *Real-Time Rendering*. A. K. Peters, Ltd., Natick, MA, USA, 3rd edition, 2008.

[3] Tomas Akenine-Möller, Jacob Munkberg, and Jon Hasselgren. Stochastic Rasterization Using Time-continuous Triangles. In *Proceedings of the 22nd ACM SIGGRAPH/Eurographics Symposium on Graphics Hardware*, pages 7–16, Aire-la-Ville, Switzerland, 2007. Eurographics Association.

[4] Solomon Boulos, Dave Edwards, Jesse Dylan Lacewell, Joe Kniss, Jan Kautz, Peter Shirley, and Ingo Wald. Interactive Distribution Ray Tracing. Technical report, SCI Institute, University of Utah, 2006.

[5] Robert L. Cook, Thomas Porter, and Loren Carpenter. Distributed Ray Tracing. *ACM SIGGRAPH Computer Graphics*, 18(3):137–145, 1984.

[6] Scott Daly. The Visible Differences Predictor: An Algorithm for the Assessment of Image Fidelity. In *Digital Images And Human Vision*, pages 179–206. MIT Press, 1993.

[7] Niranjan Damera-venkata, Thomas D. Kite, Wilson S. Geisler, Brian L. Evans, and Alan C. Bovik. Image Quality Assessment Based on a Degradation Model. *IEEE Transactions on Image Processing*, 9:636–650, 2000.

[8] François-Michel De Rainville, Christian Gagné, Olivier Teytaud, and Denis Laurendeau. Optimizing Low-discrepancy Sequences with an Evolutionary Algorithm. In *Proceedings of the 11th Annual Conference on Genetic and Evolutionary Computation*, GECCO '09, pages 1491–1498, New York, NY, USA, 2009.

[9] Rachid Deriche. Fast Algorithms for Low-Level Vision. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 12(1):78–87, 1990.

[10] Michael P. Eckert and Andrew P. Bradley. Perceptual Quality Metrics Applied to Still Image Compression. *Signal Processing*, 70:177–200, 1998.

[11] David J. Field. Relations Between the Statistics of Natural Images and the Response Properties of Cortical Cells. *Journal of the Optical Society of America A*, 4:2379–2394, 1987.

[12] John P. Frisby and James V. Stone. *Seeing - The Computational Approach to Biological Vision*. MIT Press, 2nd edition, 2010.

[13] Andrew S. Glassner. *Principles of Digital Image Synthesis*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1994.

[14] Rafael C. Gonzalez and Richard E.Woods. *Digital Image Processing*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 2nd edition, 2001.

[15] David Hubel. Eye, Brain and Vision. http://hubel.med.harvard.edu/ [cited 2011-11-15].

[16] Quan Huynh-Thu and Mohammed Ghanbari. Scope of Validity of PSNR in Image/Video Quality Assessment. *Electronics Letters*, 44(13):800–801, 2008.

[17] Frank Dachille Ix and Arie Kaufman. High-Degree Temporal Antialiasing. In *Proceedings of the Computer Animation*, CA '00, pages 49–, Washington, DC, USA, 2000.

[18] Jorge Jimenez, Jose I. Echevarria, Tiago Sousa, and Diego Gutierrez. SMAA: Enhanced Morphological Antialiasing. *Computer Graphics Forum (Proceedings of EUROGRAPHICS 2012)*, 31(2), 2012.

[19] Jorge Jimenez, Diego Gutierrez, Jason Yang, Alexander Reshetov, Pete Demoreuille, Tobias Berghoff, Cedric Perthuis, Henry Yu, Morgan McGuire, Timothy Lottes, Hugh Malan, Emil Persson, Dmitry Andreev, and Tiago Sousa. Filtering Approaches for Real-Time Anti-Aliasing. In *ACM SIGGRAPH Courses*, 2011.

[20] Stephen Joe and Frances Y. Kuo. Constructing Sobol Sequences with Better Two-Dimensional Projections. *SIAM Journal on Scientific Computing*, 30(5):2635–2654, 2008.

[21] Thouis R. Jones. Efficient Generation of Poisson-Disk Sampling Patterns. *Journal of Graphics, GPU, and Game Tools*, 11(2):27–36, 2006.

[22] Alexander Keller and Wolfgang Heidrich. Interleaved Sampling. In *Proceedings of the Eurographics Workshop on Rendering*, 2001.

[23] Laboratory for Image and Video Engineering, University of Texas. Image and Video Quality Assessment at LIVE. http://live.ece.utexas.edu/research/quality/ [cited 2012-02-21].

[24] Ares Lagae and Philip Dutré. A Comparison of Methods for Generating Poisson Disk Distributions. *Computer Graphics Forum*, 27(1):114–129, 2008.

[25] Samuli Laine and Timo Aila. A Weighted Error Metric and Optimization Method for Antialiasing Patterns. *Computer Graphics Forum*, 25(1):83–94, 2006.

[26] Samuli Laine, Timo Aila, Tero Karras, and Jaakko Lehtinen. Clipless Dual-Space Bounds for Faster Stochastic Rasterization. *ACM Transactions on Graphics*, 30(4), 2011.

[27] Samuli Laine and Tero Karras. Stratified Sampling for Stochastic Transparency. *Computer Graphics Forum (Proceedings of Eurographics Symposium on Rendering 2011)*, 30(4), 2011.

[28] Christian Lauterbach, Sung eui Yoon, and Dinesh Manocha. RT-DEFORM: Interactive Ray Tracing of Dynamic Scenes using BVHs. In *Proceedings of the 2006 IEEE Symposium on Interactive Ray Tracing*, pages 39–45, 2006.

[29] James Mannos and David Sakrison. The Effects of a Visual Fidelity Criterion on the Encoding of Images. *IEEE Transactions on Information Theory*, 20(4):525–536, 1974.

[30] Rafal Mantiuk. High Dynamic Range Visual Differences Predictor. http://www.mpi-inf.mpg.de/resources/hdr/vdp/ [cited 2012-02-20].

[31] Rafal Mantiuk, Kil Joong Kim, Allan G. Rempel, and Wolfgang Heidrich. HDR-VDP-2: A Calibrated Visual Metric for Visibility and Quality Predictions in All Luminance Conditions. *ACM Transactions on Graphics*, 30:1–14, 2011.

[32] Michael McCool and Eugene Fiume. Hierarchical Poisson Disk Sampling Distributions. In *Proceedings of the Conference on Graphics Interface 1992*, pages 94–105, San Francisco, CA, USA, 1992.

[33] Morgan McGuire, Eric Enderton, Peter Shirley, and David Luebke. Real-Time Stochastic Rasterization on Conventional GPU Architectures. In *Proceedings of High Performance Graphics 2010*, 2010.

[34] Don P. Mitchell. Spectrally Optimal Sampling for Distribution Ray Tracing. *ACM SIGGRAPH Computer Graphics*, 25(4):157–164, 1991.

[35] Don P. Mitchell and Arun N. Netravali. Reconstruction Filters in Computer Graphics. In *Proceedings of ACM SIGGRAPH 1988*, pages 221–228, New York, NY, USA, 1988.

[36] Theophano Mitsa and Krishna L. Varkur. Evaluation of Contrast Sensitivity Functions for the Formulation of Quality Measures Incorporated in Halftoning

Algorithms. In *IEEE International Conference on Acoustics, Speech, and Signal Processing*, volume 5, pages 301–304, 1993.

[37] Jacob Munkberg, Petrik Clarberg, Jon Hasselgren, Robert Toth, Masamichi Sugihara, and Tomas Akenine-Möller. Hierarchical Stochastic Motion Blur Rasterization. In *High Performance Graphics*, pages 107–118, 2011.

[38] Shinji Nishimoto, An T. Vu, Thomas Naselaris, Yuval Benjamini, Bin Yu, and Jack L. Gallant. Reconstructing Visual Experiences from Brain Activity Evoked by Natural Movies. *Current Biology*, 21:1641 – 1646, 2011.

[39] Victor Ostromoukhov. A Simple and Efficient Error-Diffusion Algorithm. In *Proceedings of ACM SIGGRAPH 2001*, pages 567–572, 2001.

[40] Marius Pedersen. and Joe Y. Hardeberg. Survey of Full-reference Image Quality Metrics. Technical Report 5, Høgskolen i Gjøviks rapportserie, 2009.

[41] Eli Peli. Contrast in Complex Images. *Journal of the Optical Society of America A*, 7:2032–2040, 1990.

[42] Matt Pharr and Greg Humphreys. *Physically Based Rendering: From Theory To Implementation.* Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2nd edition, 2010.

[43] Nikolay Ponomarenko. Tampere Image Database 2008, version 1.0. http://www.ponomarenko.info/tid2008.htm [cited 2012-03-19].

[44] Nikolay Ponomarenko, Vladimir Lukin, Alexander Zelensky, Karen Egiazarian, Jaakko Astola, Marco Carli, and Federica Battisti. TID2008 - A Database for Evaluation of Full-Reference Visual Quality Assessment Metrics. *Advances of Modern Radioelectronics*, 10:30–45, 2009.

[45] Jonathan Ragan-Kelley, Jaakko Lehtinen, Jiawen Chen, and Michael Doggett. Decoupled Sampling for Graphics Pipelines. *ACM Transactions on Graphics*, 30:17:1–17:17, 2011.

[46] Hamid R. Sheikh, Muhammad F. Sabir, and Alan C. Bovik. A Statistical Evaluation of Recent Full Reference Image Quality Assessment Algorithms. *IEEE Transactions on Image Processing*, 15(11):3440–3451, 2006.

[47] Hamid R. Sheikh, Zhou Wang, Lawrence Cormack, and Alan C. Bovik. LIVE Image Quality Assessment Database Release 2. http://live.ece.utexas.edu/research/quality [cited 2012-03-03].

[48] Peter Shirley, Timo Aila, Jonathan Cohen, Eric Enderton, Samuli Laine, David Luebke, and Morgan McGuire. A Local Image Reconstruction Algorithm for Stochastic Rendering. In *Proceedings of ACM SIGGRAPH 2011 Symposium on Interactive 3D Graphics and Games*, pages 9–13. ACM Press, 2011.

[49] Peter Shirley and Kenneth Chiu. A Low Distortion Map Between Disk and Square. *Journal of Graphics Tools*, 2:45–52, 1997.

[50] Steven W. Smith. The Scientist and Engineer's Guide to Digital Signal Processing.
http://dspguide.com/ [cited 2012-03-03].

[51] Russell L. De Valois, Duane G. Albrecht, and Lisa G. Thorell. Spatial Frequency Selectivity of Cells in Macaque Visual Cortex. *Vision Research*, 22:545–559, 1982.

[52] Zhou Wang, Alan C. Bovik, Hamid R. Sheikh, and Eero P. Simoncelli. The SSIM Index for Image Quality Assessment.
http://www.cns.nyu.edu/ lcv/ssim/ [cited 2012-02-13].

[53] Zhou Wang, Alan C. Bovik, Hamid R. Sheikh, and Eero P. Simoncelli. Image Quality Assessment: From Error Visibility to Structural Similarity. *IEEE Transactions on Image Processing*, 13(4):600 –612, 2004.

[54] Zhou Wang, Eero P. Simoncelli, and Alan C. Bovik. Multi-Scale Structural Similarity for Image Quality Assessment. In *Proceedings of IEEE Asilomar Conference on Signals, Systems, and Computers*, pages 1398–1402, 2003.

[55] Stefan Winkler. Issues in Vision Modeling for Perceptual Video Quality Assessment. *Signal Processing*, 78:231–252, 1999.

[56] Yuting Ye. Implementing Bridson's Fast Poisson Disk Sampling in PBRT. Technical report, Georgia Institute of Technology, 2007.

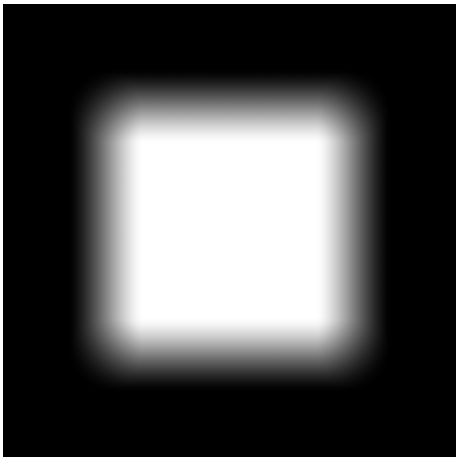
[57] Bingfeng Zhou and Xifeng Fang. Improving Mid-tone Quality of Variable-coefficient Error Diffusion Using Threshold Modulation. *ACM Transactions on Graphics*, 22:437–444, 2003.

A. TEST SCENES

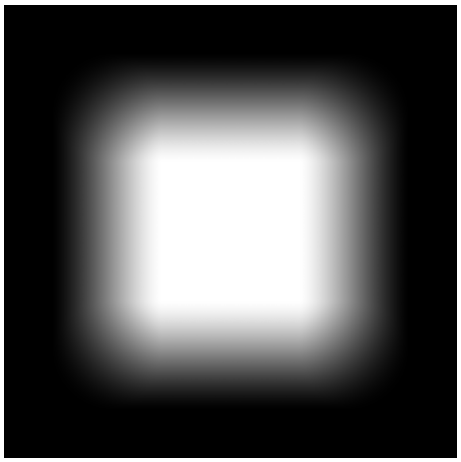
1. Blank billboard in motion
A billboard filling the viewport at t_0 and moving completely out of view towards the right at t_1. Produces a linear gradient from black to white.

2. Blank billboard in motion and defocus
A billboard filling the viewport at t_0 and moving out of view at t_1 with additional defocus blur. Produces a smooth gradient.

3. Blank billboard in 1x defocus
A blank white billboard centered in the viewport in defocus with aperture radius r

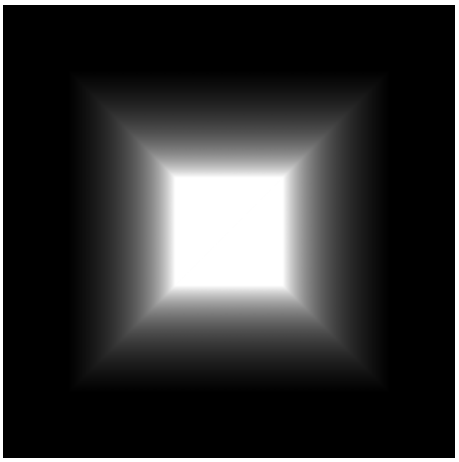


4. Blank billboard in 2x defocus
A blank white billboard centered in the viewport in defocus with aperture radius $2 \cdot r$



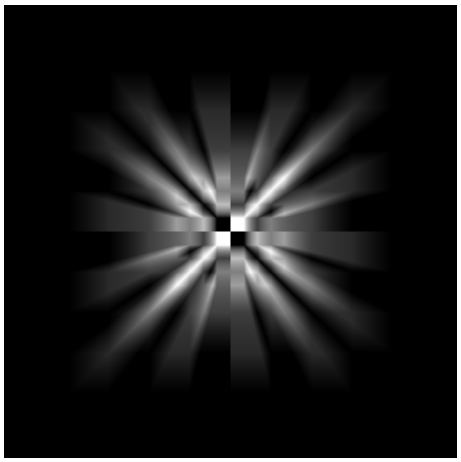
5. Blank billboard in 3x defocus

A blank white billboard centered in the viewport in defocus with aperture radius $3 \cdot r$



6. Blank billboard with camera zooming

A blank white billboard centered in the viewport with the camera moving directly towards it from t_0 to t_1



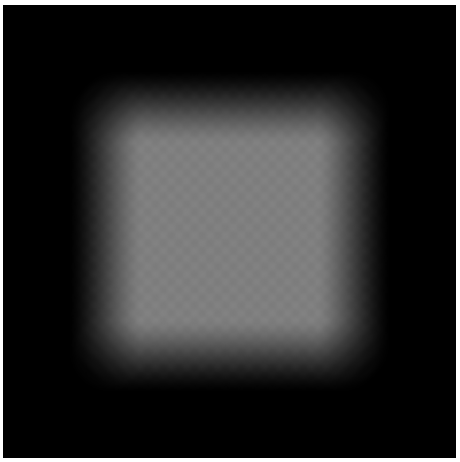
7. 8x8 checkerboarded billboard with camera zooming

A checkerboarded billboard centered in the viewport with the camera moving directly towards it from t_0 to t_1

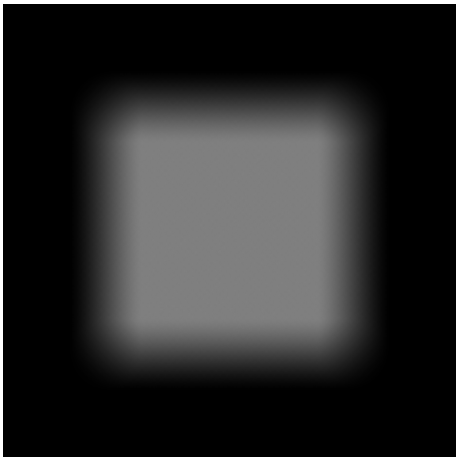
8. 8x8 checkerboarded billboard in defocus

A billboard with 8x8 checkerboard centered in the viewport in defocus with aperture radius $2 \cdot r$

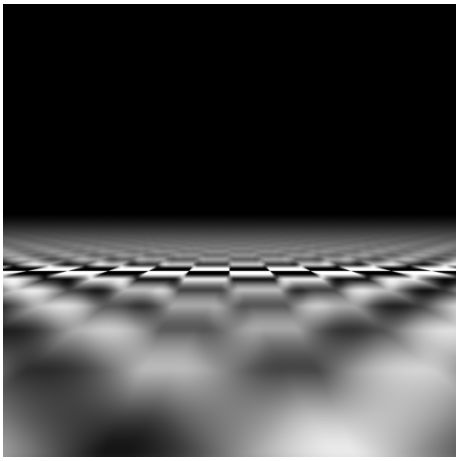
9. 16x16 checkerboarded billboard in defocus
A billboard with 16x16 checkerboard centered in the viewport in defocus with aperture radius $2 \cdot r$



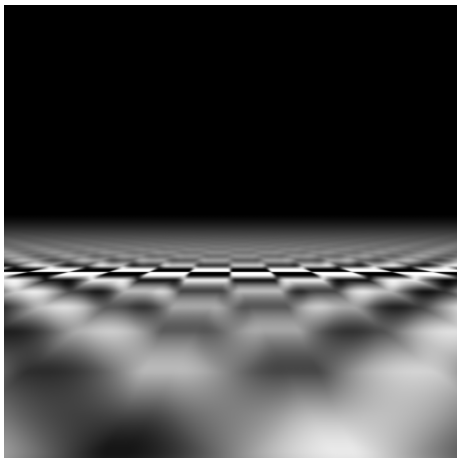
10. 32x32 checkerboarded billboard in defocus
A billboard with 32x32 checkerboard centered in the viewport in defocus with aperture radius $2 \cdot r$



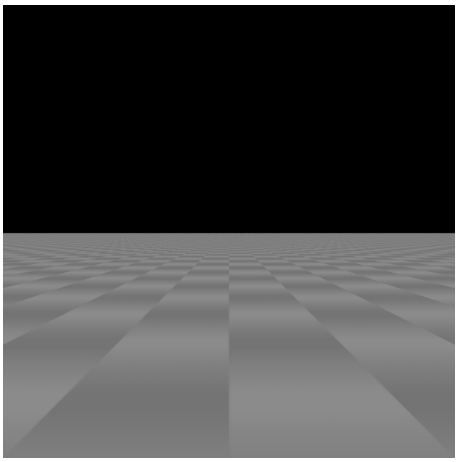
11. 64x64 checkerboarded billboard in defocus
A billboard with 64x64 checkerboard centered in the viewport in defocus with aperture radius $2 \cdot r$



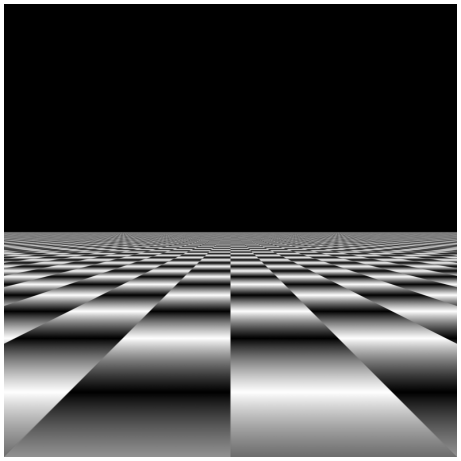
12. Checkerboard in partial defocus
A checkerboard of black and white tiles perpendicular to the image plane in partial defocus



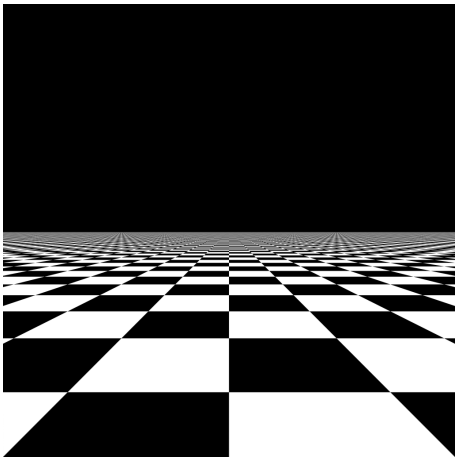
13. Checkerboard in partial defocus with texture filtering
A checkerboard of black and white tiles perpendicular to the image plane in partial defocus. The checkerboard pattern comes from a texture with trilinear filtering.



14. Checkerboard with high amount of parallel motion
A checkerboard of black and white tiles perpendicular to the image plane in complete focus. The camera is moving parallel to the checkerboard towards the horizon across 9 rows of checkerboard tiles.



15. Checkerboard with low amount of parallel motion
A checkerboard of black and white tiles perpendicular to the image plane in complete focus. The camera is moving parallel to the checkerboard towards the horizon across one row of checkerboard tiles.



16. Checkerboard in focus
A checkerboard of black and white tiles perpendicular to the image plane in complete focus.

17. Fairy Forest
A scene depicting a fairy in a forest, with the camera stationary near the fairy's hand.

18. Mustang in defocus and motion
A Mustang in partial defocus and moving forward on a blank blue background.

19. Mustang in motion
A Mustang in complete focus moving forward on a blank blue background.

20. Mustang in defocus
A Mustang in partial defocus on a blank blue background.