Textures on Rank-1 Lattices

S. Dammertz[1] and H. Dammertz[1] and A. Keller[2] and H. P. A. Lensch[3]
[1](sabrina, holger).dammertz@uni-ulm.de Ulm University, Germany
[2]alex@mental.com, mental images GmbH, Germany
[3]hendrik.lensch@uni-ulm.de Ulm University, Germany

Figure 1: *Comparison of standard textures with rank-1 lattice textures where each image contains the same number of picture elements* (64×64)*. Rank-1 lattices can very closely approximate the hexagonal lattice. This reduces aliasing and results, among other things, in an improved reproduction of non axis-parallel lines as can be seen clearly in the middle image.*

Abstract

Storing textures on orthogonal tensor product lattices is predominant in computer graphics, although it is known that their sampling efficiency is not optimal. In two dimensions, the hexagonal lattice provides the maximum sampling efficiency. However, handling these lattices is difficult, because they are not able to tile an arbitrary rectangular region and have an irrational basis. By storing textures on rank-1 lattices, we resolve both problems: Rank-1 lattices can closely approximate hexagonal lattices, while all coordinates of the lattice points remain integer. At identical memory footprint texture quality is improved as compared to traditional orthogonal tensor product lattices due to the higher sampling efficiency. We introduce the basic theory of rank-1 lattice textures and present an algorithmic framework which easily can be integrated into existing off-line and real-time rendering systems.

1. Introduction

The traditional orthogonal equidistant lattice with its square texture elements is far from optimal for storing and displaying general images. In the context of sampling for signal processing this has been recognized already very early by Petersen et al. [PM62]. They quantify the quality of a sampling lattice by its sampling efficiency

$$\eta := \frac{R}{P},$$

where R is the area of the in-circle of the fundamental Voronoi cell and P denotes the volume of the fundamental Voronoi cell itself in the dual lattice. The sampling efficiency measures how well the sampling points of a given lattice capture the *isotropic* spectrum of a band-limited function. In this context a sampling efficiency of 100% would allow to perfectly represent the band-limited function. Petersen et al. derive the optimal sampling lattices for up to 8 dimensions. While in 2D the optimal lattice is the well known hexagonal lattice with $\eta = 90.7\%$, the 2D square lattice has a sampling efficiency of only $\eta = 78.5\%$.

Hexagonal sampling lattices have been investigated for more than 40 years, giving rise to the field of hexagonal image processing. With applications in medicine, cameras, or hexagonal displays, hexagonal image processing (HIP) is an active field in computer vision. Middleton et al. [MS05] give a comprehensive survey of research on this topic and present a framework for hexagonal image processing based on hierarchical aggregates.

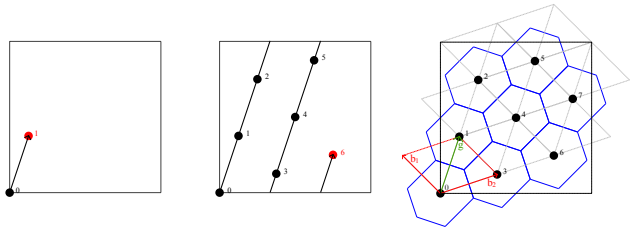


Figure 2: Generating a rank-1 lattice with $n = 8$ points and the generator vector $\mathbf{g} = (1, 3)$. For each image the multiple $\{\frac{i}{8} \cdot \mathbf{g}\}_1$, $i = 1, 6, 7$ of the generator vector $\mathbf{g}$ is plotted. The rightmost image depicts the whole lattice $L_{8,(1,3)}$ including the Voronoi cells and the Delaunay triangulation.

Both, square and hexagonal texture elements have the property of tiling the plane. Thus hexagonally and square sampled images can be considered as a periodic monohedral tiling [MS05]. These tiles represent the Voronoi cells of the sampling points. The basis of a hexagonal lattice is $B_{\mathrm{H}} = ((1,0)^\top, 0.5 \cdot (1,\sqrt{3})^\top)$, with the first basis vector being parallel to the x-axis and the second one being rotated by 60° from the first. Due to the irrational basis the points of a hexagonal lattice cannot be represented exactly on a computer and it is impossible to tile the unit square. Generally, for both the square and hexagonal lattice there are numbers n of texture elements, for which the unit square $I = [0,1)^2$ cannot be tiled, e.g. for n being prime numbers.

In this paper we introduce rank-1 lattices for texturing with the following advantages:

- An arbitrary number of texture elements can be used in contrast to square and hexagonal lattices where the number of texture elements is always a product of x- and y-resolution.
- All computations on rank-1 textures can be performed using fast integer arithmetic.

- The new texture scheme provides a close rational approximation to hexagonal textures for any resolution (even power of 2). We also show the equivalence of hexagonal and rank-1 textures for a subset of rank-1 lattices.
- We provide the necessary theory and algorithms to efficiently implement rank-1 textures into any rendering system.

Figure 2 conceptually illustrates the construction of a rank-1 lattice from a single generator vector (hence rank 1). Looking at the geometry of such lattices, a rank-1 lattice tiles the plane by its Delaunay triangulation and Voronoi diagram. Thus we can structure the pixel layout by the Voronoi cells of a rank-1 lattice. Contrary to the hexagonal and square lattice, rank-1 lattices perfectly tile the unit square $I = [0,1)^2$ periodically for any number n of points. Therefore more flexibility is offered with respect to the shape of the image domain and the number of pixels can be chosen freely. As shown in Figure 3 for $n = 56$, rank-1 lattices can be chosen such that they approximate the hexagonal grid. This results in a sampling efficiency close to the optimum [MS05, PM62]. Compared to the traditional square lattice a better angular resolution is achieved as there are more nearest neighbors, i.e. adjacent lattice cells.

2. Previous Work

A lot of research has been performed with respect to generating hexagonally sampled images. For example [MS05, VVPL02, CVF08] address the problem of resampling between orthogonal and hexagonal images. However, hexagonal images have not been used as textures in rendering so far.

In three dimensions, non-Cartesian lattices have been studied with respect to the visualization of volumetric data. [EVM08] use box splines in order to efficiently reconstruct volumetric data sampled on the Body Centered Cubic (BCC) lattice, taking advantage of its optimal spectral sphere packing property. In [NM02], 4D BCC grids are examined in the context of visualizing 4D data sets. [Csb05] proposes a novel high-quality reconstruction scheme to reconstruct volumetric data sampled on an optimal BCC grid.

Rank-1 lattice rules were first proposed by Korobov [Kor59] and have been widely examined since then [Nie92, HW81]. Sloan and Joe [SJ94] give a mathematical survey on rank-1 lattices. The basic theory of 2D rank-1 lattices with maximized sampling efficiency is described in [DK08]. In this paper the use of rank-1 lattices for anti-aliasing is analyzed and the idea of using rank-1 lattices for storage of images and building displays is first mentioned without further investigation. Here we develop the necessary data-layout, neighborhood computation and addressing scheme for an efficient use of rank-1 lattices for texturing. Additionally, we provide numerical evidence for the improved texturing quality.

3. Rank-1 Lattices

In this section we give a theoretical overview of rank-1 lattices and how to approximate the hexagonal lattice.

The points $\mathbf{x}_i$ of an s-dimensional rank-1 lattice [Nie92, SJ94] in the unit cube $I^s = [0,1)^s$

$$L_{n,\mathbf{g}} := \left\{ \mathbf{x}_i := \left\{ \frac{i}{n}\mathbf{g} \right\}_1 \Big| i = 0, \ldots, n-1 \right\} \tag{1}$$

are generated by using one suitable integer generator vector $\mathbf{g} \in \mathbb{N}^s$ for a fixed number $n \in \mathbb{N}$ of points, where $\{\mathbf{x}\}_1$ denotes the fractional part of a vector $\mathbf{x}$, i.e. $\{\mathbf{x}\}_1 := \mathbf{x} \bmod 1$. The *mod 1* operation restricts the lattice to the unit square resulting in a one-periodic pattern. In two dimensions the generator vector corresponds to $\mathbf{g} = (g_1, g_2)$ with $g_1, g_2 \in \{0, \ldots, n-1\}$ due to the modulo operation.

Generally, a lattice L in $\mathbb{R}^s$ is a discrete subset of $\mathbb{R}^s$ which is closed under addition and subtraction and consequently always contains the origin. Lattices can be classified by their rank r, which corresponds to the minimum number of vectors being necessary to generate L [SJ94]. Examples for lattices of rank 2 are the well known square and hexagonal lattice, for which the dimension coincides with the rank. Contrary, an s-dimensional rank-1 lattice is defined by only one generator vector $\mathbf{g}$.

Figure 2 visualizes the geometric properties of rank-1 lattices. The solid lines depict the Voronoi diagram. Each cell is generated by a lattice point and contains the points of the unit torus, which are closer to this lattice point than to any other. In addition, the lattice points are the centroids of the Voronoi cells. The dashed lines represent the dual graph, i.e. the Delaunay tessellation, which can be generated by dividing the fundamental parallelepiped along its shorter diagonal.

Note that even though for easier explanation we only show rank-1 lattices on square regions in this paper, they are not restricted to the unit square, but can be constructed for any rectangular region for any number of lattice points by means of weighted norms. This is illustrated in Figure 4.

3.1. Lattice Bases

The basis of a lattice L is an $s \times s$ matrix $B = (\mathbf{b_1}, \ldots, \mathbf{b_s})$ if every point in the lattice can be generated by an integer linear combination of the basis vectors $\mathbf{b_1}, \ldots, \mathbf{b_s}$.

$$L = \{\mathbf{x} = B \cdot \mathbf{l} : \mathbf{l} \in \mathbb{Z}^s\} \tag{2}$$

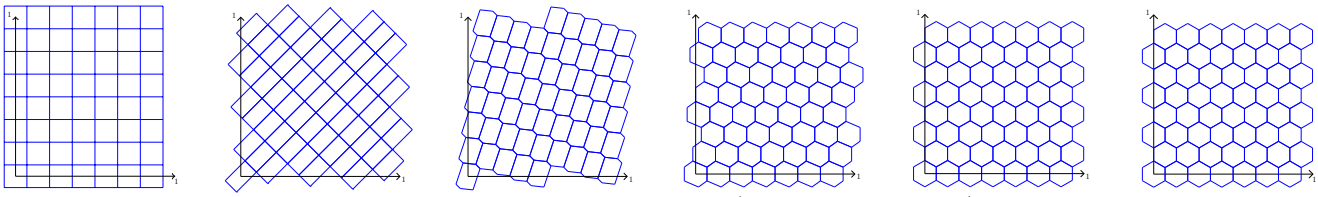
The number s of basis vectors is said to be the dimension of L. The basis vectors and the origin span the so-called fundamental parallelepiped which induces a partition of $\mathbb{R}^s$ into lattice cells of the same volume and orientation, as illustrated by the red parallelogram Λ in the rightmost image of Figure 2.

$$\begin{aligned} \Lambda &= \Lambda(\mathbf{b_1}, \ldots, \mathbf{b_s}) \\ &= \{\gamma_1 \mathbf{b_1} + \ldots + \gamma_s \mathbf{b_s} : 0 \le \gamma_i \le 1, 1 \le i \le s\}, \end{aligned} \tag{3}$$

A lattice has infinitely many different bases which all share the same determinant, i.e. the volume of the fundamental parallelepiped. This means that $\det(L) := |\det(B)|$ is invariant for each lattice L, therefore being called the lattice determinant. Hence, a matrix B of linear independent vectors only represents a valid basis for L if its determinant equals $\det(L)$ [Kan87]. For example the vectors $(\mathbf{b}_1, \mathbf{b}_2)$, $(\mathbf{b}_1, \mathbf{g})$, as well as $(\mathbf{g}, \mathbf{b}_2)$ or $(\mathbf{g} + \mathbf{b}_2, \mathbf{b}_2)$ are bases of the rank-1 lattice plotted in the rightmost image of Figure 2. Contrary, the vectors $(\mathbf{b}_1, 2 \cdot \mathbf{b}_2)$ do not constitute a basis of $L_{8,(1,3)}$. In the following we will use a Minkowski reduced basis, as visualized by the vectors $\mathbf{b}_1$ and $\mathbf{b}_2$ in Figure 2. Such a basis contains the s shortest linearly independent vectors and can be computed by either a so-called basis reduction or a linear search over all lattice points [Kan87, AG85, Hel85].

3.2. Approximating the Hexagonal Lattice

We now investigate how to choose a 2D generator vector to approximate the hexagonal structure by a rank-1 lattice. This is related to densest sphere packing in two dimensions (circle packing) which has been studied for a long time [CSB87, Mar03, Sie89] and is connected to constructing lattices with longest possible shortest nonzero vectors for a given lattice density $\det(L)$. This is equivalent to maximizing the minimum



square, $\eta = 78.5\%$ | $\mathbf{g} = \binom{3}{11}, \eta = 44.9\%$ | $\mathbf{g} = \binom{1}{9}, \eta = 56.1\%$ | $\mathbf{g} = \binom{1}{21}, \eta = 81.3\%$ | $\mathbf{g} = \binom{4}{7}, \eta = 89.8\%$ | hex., $\eta = 90.7\%$

Figure 3: In the sequence of rank-1 lattices $L_{56,\mathbf{g}}$ the sampling efficiency increases from left to right. For comparison the square lattice is added to the left, whereas the hexagonal lattice is plotted to the right of the image sequence, each having $n = 7 \cdot 8 = 56$ points. For this number of points both the square and hexagonal lattice do not tile the unit square.

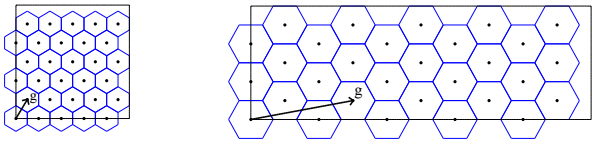


Figure 4: Rank-1 lattices on a square and non-square region, having the same number of $n = 30$ lattice points. Note that the left lattice is not a scaled version of the right one, as the almost hexagonal Voronoi cells show.

n	$\mathbf{g}$	$\text{index}_{\mathbf{b}_1}$	$\text{index}_{\mathbf{b}_2}$	η
16384	$(137, 13)$	2511	2512	90.4%
30976	$(188, 17)$	1	1813	90.3%
50176	$(211, 116)$	4756	4757	90.2%
135424	$(13, 395)$	52112	52113	90.6%
147456	$(413, 27)$	54627	54626	90.4%
160000	$(430, 17)$	65860	65861	90.5%
262144	$(1, 1990)$	527	395	90.3%

Table 1: Lattice parameters for some maximized minimum distance lattices used in the results section, including the indices of the basis vectors $\text{index}_{\mathbf{b}_1}$ and $\text{index}_{\mathbf{b}_2}$, and the sampling efficiency.

mutual distance in the sampling pattern using the L_2 norm on the unit torus. In computer graphics this concept appears for example in Lloyd's relaxation algorithm [Llo82], which yields sampling patterns with blue noise properties when applied to an initially random point set.

Similarly we can select rank-1 lattice generator vectors such that the minimum distance in the lattice is maximized: From all possible generator vectors $\mathbf{g} = (g_1, g_2) \in \{1, \ldots, n-1\}^2$ for given n, we choose one for which the corresponding rank-1 lattice $L_{n,\mathbf{g}}$ has the maximum minimum distance [DK08]. As illustrated in the two rightmost images of Figure 3, these so-called maximized minimum distance rank-1 lattices closely approximate the hexagonal lattice. A direct construction scheme for maximized minimum distance lattices is only known for $n = 2F_mM_m$ points [CR97] with F_M and M_m given by the relation $\left(\frac{F_m}{M_m}\right)_{m=1\ldots\infty} = \frac{2}{1}, \frac{5}{3}, \frac{7}{4}, \frac{19}{11}, \ldots$, which is the sequence of convergents of the continued fraction equal to $\sqrt{3}$. In this context a convergent is a rational approximation of the irrational number, which is represented by the continued fraction, with the odd-numbered ones being larger and the even-numbered ones being smaller than this number. In Appendix B we prove the following theorem:

Theorem 1 *Using a hexagonal lattice for texturing on* $[0, 1)^2$ *with* $n = 2F_mM_m$ *points is* equivalent *to using the maximized minimum distance rank-1 lattice with* $n = 2F_mM_m$ *points.*

For other n the generator vector can be determined by computer search [DK08] to maximize the sampling efficiency.

Table 1 lists the lattice parameters and the sampling efficiency for some maximized minimum distance lattices which are used in the results section. The sampling efficiency for those lattices is very close to the sampling efficiency of the hexagonal lattice. Thus maximized minimum lattices can hardly be visually distinguished from the hexagonal lattice.

4. Rank-1 Lattice Texture Storage and Access

Contrary to the square lattice the points of a rank-1 lattice cannot easily be addressed by 2-dimensional integer Cartesian coordinates, since they are not aligned in two orthogonal directions. In this section we show how data on a rank-1 lattice can be stored as a linear (1D) array and how to compute the address from 2D texture coordinates. In the following we perform all computations on $[0, n)^2$ by multiplying the original definition of Equation 1 by n. This allows computations to use only integer arithmetic.

In order to store a complete rank-1 lattice texture we need the number of texture elements n, a generator vector $\mathbf{g}$ and the corresponding Minkowski reduced basis along with the indices of the basis vectors being necessary for address computations. These indices can be acquired during the linear search over all lattice points in the process of computing the lattice basis. The image data is organized as a linear array of size $3 \cdot n$ (RGB color model). Exploiting the property that every lattice point $\mathbf{x}_i$ can be generated by a single generator vector $\mathbf{g}$ allows for a very simple addressing scheme. As

$$\mathbf{x}_i = \{i \cdot \mathbf{g}\}_n, \quad \{\cdot\}_n := \cdot \bmod n$$

is uniquely defined by i, this index can be used for accessing the data array.

4.1. Nearest Neighbor Look-Up

The simplest texture look-up strategy uses the color of the texture element closest to the texture coordinate. This means to compute the nearest lattice point to $(u, v) \in [0, 1)^2$ in Cartesian coordinates. For this purpose the texture coordinate (u, v) has to be transformed into the lattice basis B, yielding the lattice coordinates $(s, t) = B^{-1}(u, v)^T$. Then $(d_1, d_2) = \lfloor s, t \rfloor$ is the anchor point of the lattice cell induced by the fundamental parallelepiped (in lattice coordinates) in which the point (u, v) lies in $[0, 1)^2$. The nearest lattice point to (u, v) is contained in the set $M = \{(d_1, d_2), (d_1 + 1, d_2), (d_1, d_2 + 1), (d_1 + 1, d_2 + 1)\}$ representing the vertices of the lattice cell. Hence it results as the vertex $\mathbf{v} \in M$ with the shortest Euclidean distance to (u, v). Note that the distance computation has to be performed in the Cartesian coordinate system, as the Voronoi cell is distorted in the rank-1 lattice coordinate system. Since the lattice coordinates of $\mathbf{v}$ are known, its index can simply be computed according to the addressing scheme described in the next section.

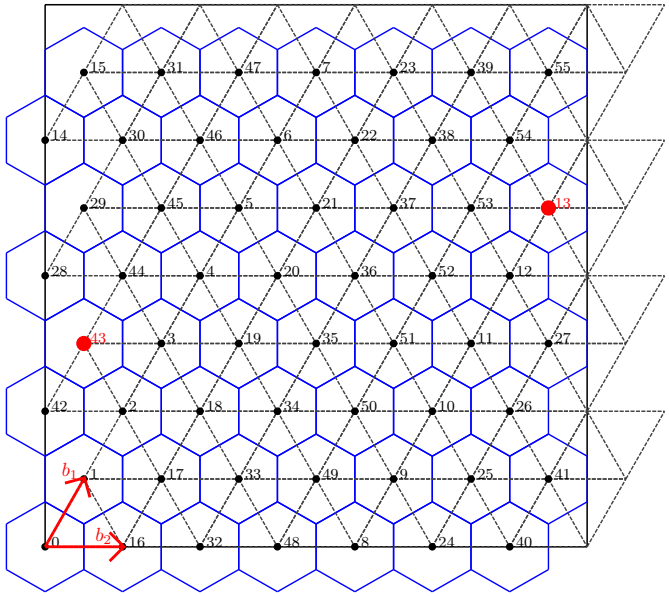


Figure 5: Example for computing the index of an arbitrary lattice point for the lattice $L_{56,(4,7)}$ with $index_{b_1} = 1$ and $index_{b_2} = 16$.

4.2. Address Computation

Let $B = (\mathbf{b}_1, \mathbf{b}_2)$ be a Minkowski-reduced basis of the lattice $L_{n,\mathbf{g}}$ and let $k = \text{index}_{b_1}$ and $l = \text{index}_{b_2}$ be the indices of the lattice points corresponding to $\mathbf{b}_1$ and $\mathbf{b}_2$. Given any lattice point $\mathbf{x}_i$, the first step in computing its point index i consists in transforming $\mathbf{x}_i$ into the lattice basis, i.e. its coordinates $\mathbf{x}_{i,B} = (s,t)$ in the lattice basis have to be computed, such that $\mathbf{x}_i$ can be expressed as

$$\mathbf{x}_i = s \cdot \mathbf{b}_1 + t \cdot \mathbf{b}_2, \quad s,t \in \mathbb{Z}.$$

Then the index i of a lattice point $\mathbf{x}_i$ can be computed according to the following theorem.

Theorem 2 *Let $\mathbf{x}_{i,B} = (s,t)$ be the representation of the lattice point $\mathbf{x}_i$ in the Minkowski reduced basis B of $L_{n,\mathbf{g}}$ and let $k = index_{b_1}$ and $l = index_{b_2}$ be the indices of the basis vectors $\mathbf{b}_1$ and $\mathbf{b}_2$. Then the index i of $\mathbf{x}_i$ results as*

$$i = \{s \cdot k + t \cdot l\}_n$$

The proof can be found in Appendix A.

Figure 5 illustrates the address computation for the lattice $L_{56,(4,7)}$ with the basis $((4,7)^T, (8,0)^T)$ and the base indices $k = 1$, $l = 16$. Consider the lattice point $\mathbf{x} = (4,21) = 3 \cdot \mathbf{b_1} + (-1) \cdot \mathbf{b_2}$. Then $i = \{3 \cdot 1 + (-1) \cdot 16\}_{56} = \{-13\}_{56} = 43$. Similarly, for the point $\mathbf{x} = (52,35)$ we have $\mathbf{x} = 5 \cdot \mathbf{b_1} + 4 \cdot \mathbf{b_2}$, yielding $i = \{5 \cdot 1 + 4 \cdot 16\}_{56} = \{69\}_{56} = 13$. $\mathbf{x}_{43}$ and $\mathbf{x}_{13}$ are highlighted in Figure 5.

Neighborhood Computation. The indices of neighboring lattice points are simply computed by addition or subtraction of the indices k, l and $m = \{l - k\}_n$ modulo n, where m corresponds to the index of the vector $\mathbf{b}_2 - \mathbf{b}_1$. This means that the neighborhood of a lattice point $\mathbf{x}_i$ is defined by the set of indices $N(i) = \{i, \{i+k\}_n, \{i+l\}_n, \{i+m\}_n, \{i-k\}_n, \{i-l\}_n, \{i-m\}_n\}$. These are the six closest neighboring lattice points and form a hexagonal structure. For example $N(35) = \{35, 36, 51, 50, 34, 19, 20\}$ with $k = 1$, $l = 16$ and $m = \{16 - 1\}_{56} = 15$.

5. Implementation

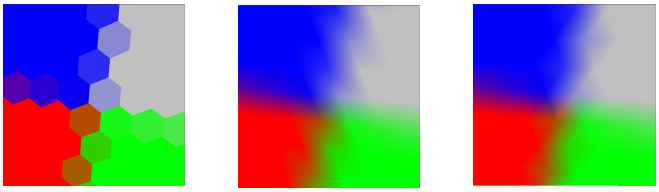
Here we provide the necessary details to integrate the rank-1 lattice textures into a rendering system.

5.1. Linear Interpolation

For texture magnification some form of interpolation is needed. This is also an important issue in real-time applications where textures are often only available in lower resolutions. As for the square lattice there are several possibilities regarding texture interpolation. Additionally to the commonly used bilinear interpolation, the geometric structure of rank-1 lattices also implies interpolation on the Delaunay triangles. In the following we describe both interpolation methods.

Bilinear Interpolation on Lattice Cells Induced by the Fundamental Parallelepiped: A bilinear interpolation is performed on the lattice cell $\mathbf{x}_i + \Lambda$ (see Equation 3) in which the texture coordinate lies. The anchor point of the lattice cell (in lattice coordinates) is given by $(d_1, d_1) = \lfloor s, t \rfloor$, with (s,t) representing the lattice coordinates of (u,v). Hence the parallelepiped is spanned by the vertices $\{(d_1, d_2), (d_1+1, d_2), (d_1, d_2+1), (d_1+1, d_2+1)\}$ and the colors of the associated r1-texels are interpolated.

Barycentric Interpolation on Delaunay Triangle: After the lattice cell $\mathbf{x}_i + \Lambda$ in which (u,v) lies has been determined, the texel colors of both Delaunay triangles are weighted using barycentric interpolation. That way the triangle in which the texture coordinate lies is detected automatically. This means that the texture coordinate is located in that triangle for which the sum of the barycentric coordinates is less or equal to one and the coordinates themselves are greater or equal to zero.



Nearest neighbor. Bilinear. Barycentric.

Figure 6: Visual comparison of texture look-up schemes on rank-1 lattices.

Comparison. Figure 6 compares close-ups of the nearest neighbor look-up and the two linear interpolation methods. While parallelepipeds become visible in the case of the bilinear interpolation, the Delaunay triangles appear in the barycentric interpolation scheme. These structures only become visible for significant magnification. The bilinear interpolation needs four array accesses while the barycentric interpolation only uses three values. It is also possible to use the higher order interpolation methods defined on lattices.

5.2. Tiling and Multi-Resolution

In order to use large textures efficiently it is necessary to support tiling and cached access. Peachy [Pea90] proposed the textures on demand (TOD) approach, a technique designed to structure and efficiently access large amounts of stored texture data such that the I/O and CPU cost of access is minimized. The key elements of the system are texture tiles (i.e. the texture is tiled in square units adapted to fixed I/O size) and prefiltered textures provided by means of resolution sets.

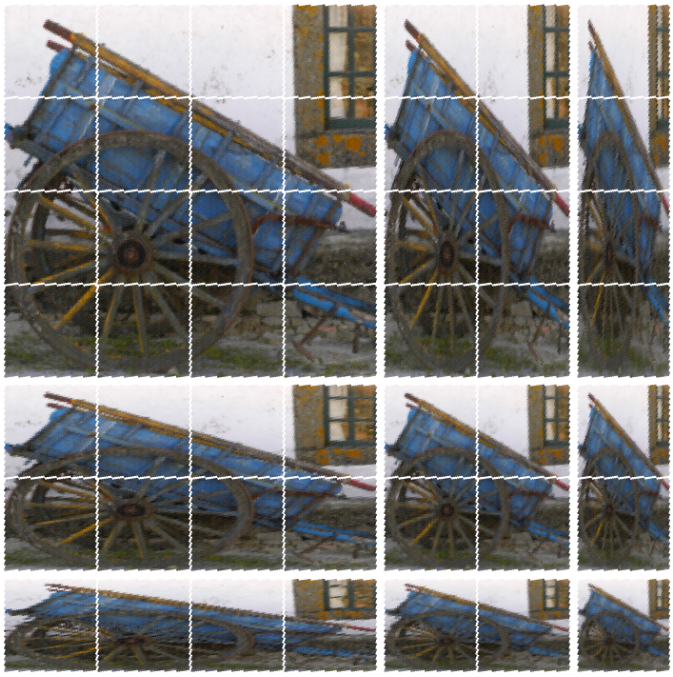


Figure 7: Tiled rank-1 lattice r-set with $[S_0, T_0] = [4, 4]$ and $L^{tile}_{2048,(31,39)}$. The rank-1 lattice r-set thus consists of the following tiled images $[4, 4]_n$, $[4, 2]_n$, $[4, 1]_n$, $[2, 4]_n$, $[1, 4]_n$, $[2, 2,]_n$, $[2, 1]_n$, $[1, 2]_n$, $[1, 1]_n$, $n = 2048$.

Due to their periodicity rank-1 lattices are suited very well to tiled texture layout. Tiled rank-1 lattice resolution sets (rank-1 lattice r-sets) are built following the example of [Pea90] and by taking advantage of the rank-1 lattice structure. The tile size n, i.e. number of lattice points per tile, can be chosen freely, and thus is independent of source image resolution. For example n can always be chosen as *any* integer power of 2, as a rank-1 lattice image can be generated for any target resolution with arbitrary width-to-height ratio. This is not always possible for images on square lattices, where the resolution is given as the product of the x- and y-resolution. A tiled image on a rank-1 lattice is composed of $S \times T$ tiles, denoted by $[S, T]_n$. Therefore, the image resolution corresponds to $S \cdot T \cdot n$.

The size of the rank-1 lattice r-set is determined by the number of tiles $[S_0, T_0]_n$ in the highest resolution. The resolution of the images in the rank-1 lattice r-set is set to be $[S, T]$ where S and T are integers with $S_0 \geq S \geq 0$ and $T_0 \geq T \geq 0$. One possibility to define a *complete* rank-1 lattice r-set is to demand the rank-1 lattice r-set to consist of all downsampled versions of the source image, where the parameters S and T are determined by consecutive integer bisection so that there are $\lfloor \log(S_0) + 1 \rfloor \times \lfloor \log(T_0) + 1 \rfloor$ elements in the rank-1 lattice r-set. An example for a complete rank-1 lattice r-set according to this scheme is shown in Figure 7. If S_0 and T_0 are a power of 2 the storage cost for a complete rank-1 lattice r-set can be computed in the same way as in [Pea90] and therefore is at most four times the storage cost of the original image $[S_0, T_0]$.

The TOD implementation based on rank-1 lattices principally follows the structure of [Pea90]. The only difference consists in the rank-1 lattice r-set member identification, which is performed in two steps. At first we determine the rank-1 lattice r-set members satisfying the width-to-height ratio of the texture region $(s_{\text{width}}, t_{\text{width}})$ with center (s, t) of which the renderer requests a filtered value. Among those candidates we select the one with $\frac{1}{n \cdot S \cdot T} \geq s_{\text{width}} \cdot t_{\text{width}}$, i.e. we choose the resolution for which the pixel area corresponds to the area of the required texture region. Given a suitable member of the rank-1 lattice r-set, the tile number, which is necessary to build the tile key, is obtained by $\lfloor s \cdot S \rfloor, \lfloor t \cdot T \rfloor$. In order to fulfill the texture pixel demand of the renderer the tile itself is indexed as described in Section 4.1. The use of rank-1 lattices allows for a much more flexible and easier implementation of a TOD system than possible with hexagonal lattices.

5.3. Acquisition and Resampling

Similar to hexagonally sampled images there are two methods to acquire images on rank-1 lattices. The first one is to use specialized hardware for image acquisition, (in [MS05] a survey is given in the context of hexagonal sensors, also being apparent in medical imaging and remote sensing), but up to now there exists no rank-1 lattice hardware like displays or acquisition devices. The second approach works by using software conversion of regularly acquired images. Converting an image represented in one lattice to another lattice representation is called resampling. The typical procedure is to first reconstruct a continuous image from the original samples by using an interpolation function. This reconstructed image is then resampled with the new lattice. In an implementation reconstruction and resampling are usually combined in a single step.

Middleton et al. generate hexagonally sampled images according to this interpolation scheme and compare nearest neighbor, bilinear and bicubic interpolation [MS05]. They conclude that a bilinear filter kernel is sufficient for the purpose of generating hexagonally sampled images when the source material has a similar resolution to the target lattice. In [VVPL02] the authors propose an algorithm for image resampling between orthogonal and hexagonal images with similar sampling resolutions, which is based on the least squares approach, thus minimizing loss of information by incorporating a spline transform. [CVF08] present a new grid conversion approach in order to resample between hexagonal and Cartesian lattices with the same sampling density. This method allows for recovering the initial data exactly, when the same algorithm is applied for the converse operation. All these methods can be directly applied to resampling on rank-1 lattices when using the associated basis vectors.

To compute rank-1 lattice textures for the experiments in the result section we have chosen the very simple generation approach of downsampling a very high resolution source image to a rank-1 lattice image of maximal $\frac{1}{16}$ of the source image resolution. This corresponds to the "box-filter" interpolation and therefore facilitates the comparison of rank-1 to square lattice textures, since both kinds of textures are computed using the same algorithm.

5.4. Optimizations

The structure of rank-1 lattices allows for further optimizations. The number of lattice points can be chosen to fit for example the caching structure of the target system. The optimal size is usually a power of 2. Having $n = 2^K$ allows then to additionally replace the costly modulo operation in the lattice computations by a simple binary AND.

Another optimization is to use only rank-1 lattices in Korobov form. Korobov lattices $L_{n,a}$ are a special class of rank-1-lattices. In two dimensions, their generator vector has the form $\mathbf{g} = (1, a)$, thus being uniquely determined by the tuple (n, a). While this restricts the possible choice of generator vectors, maximized minimum distance rank-1 lattices can always be represented in Korobov form for n being a power of 2. Now

the x-coordinate of each lattice points $x_i = i \cdot \mathbf{g}_n$ is directly the index i.

For a given lattice L there exists more than one generator vector $\mathbf{g}$ producing the same lattice. For $n = 2^k$ there are 2^{k-1} different $\mathbf{g}$ for the identical lattice. As the generator vector is responsible for the data layout in the texture array this allows for some freedom in choosing this layout. For example it can be optimized for higher locality for filtered texture access.

5.5. GLSL Example

The concept of rank-1 lattice textures can be used in real time rendering using OpenGL or DirectX for example. Therefore, the pixel shaders used in the rendering engine have to be extended by the addressing scheme from Section 4.1. A GLSL example for rank-1 texture access in Korobov form is given below. This code is written to illustrate an implementation and could be further optimized. In many shaders the memory access is the bottleneck and the additional address computation can be neglected. For $n = 2^k$ the computations are even more simplified and an efficient hardware implementation would be possible.

```
varying vec4 uv;            // 2d texture coordinate
uniform sampler1D r1data; // image on the rank-1 lattice
uniform vec4 L;             // lattice parameters (n,-,a,d)
uniform vec4 b;             // the tow basis vectors for L

void main(void) {
    // Project uv into lattice basis
    vec2 ft = floor(vec2(uv.x * b.q - uv.y * b.p,
                         uv.y * b.s - uv.x * b.t)*L.w);

    // 4 indices of the 4 corner points
    vec4 idx = mod(vec4(ft.x*b.s+ft.y*b.p,
                        (ft.x+1.0)*b.s+ft.y*b.p,
                        (ft.x+1.0)*b.s+(ft.y+1.0)*b.p,
                        ft.x*b.s+(ft.y+1.0)*b.p), L.x);

    // corner points in Cartesian coordinates
    vec4 px = idx/L.x;
    vec4 py = mod((idx*L.z)/L.x, 1.0);

    // squared distance to the 2d texture coordinate
    vec4 dp = (px-uv.x)*(px-uv.x) + (py-uv.y)*(py-uv.y);

    // get the index based on the horizontal minimum mm;
    float mm = min(min(dp.x, dp.y), min(dp.z, dp.w));
    float i = (mm==dp.y)?idx.y:
              ((mm==dp.z)?idx.z:((mm==dp.w)?idx.w:idx.x));

    gl_FragColor = texture1D(r1data, i);
}
```

6. Results

We have implemented textures on rank-1 lattices in a ray tracing system and compare them both visually and numerically to textures using square pixels. For the numerical comparison we use the L_2-norm, which has been shown to yield similar results as visually motivated error metrics [Edw07].

To show the advantage of these textures compared to square lattices we use four different textures at different resolutions. The rendering system creates high dynamic range images and the comparison computes the mean square error between the reference image and the downsampled texture images. The reference image was computed using 1024 random samples per pixel at a resolution of 512×512 pixels (using a high resolution texture on the square lattice). The comparison images were computed with 128 random samples per pixel.

The resolution of the source image was 2048×2048 and we vary the target resolution from 128×128 to 512×512 in steps of 16×16. The target texture is always a downsampled version of the source image using a box filter in both cases. The rank-1 lattice texture contains exactly the same number of points as the square lattice texture. For a direct visual comparison see Figure 1, which contains example textures for $n = 64 \times 64$ texture elements. Figure 8 shows the results of the numerical comparison. Except for the checker board texture, rank-1 lattice textures consistently outperform the square lattice textures for any resolution as expected due to the increased sampling efficiency. The checker board is the worst-case comparison for maximized minimum distance rank-1 lattice textures to square lattices as the checker board itself can be perfectly represented by a square lattice. Nevertheless, the rank-1 lattice shows a consistent behavior and outperforms the square lattice when the resampled resolution is not a divider of the source resolution.

The addressing on rank-1 lattices is slightly more expensive than the addressing of square lattice textures. In our rendering system the performance drop was about 5% for all tested scenes. Since these scenes contain only simple geometry and lighting calculations the relative performance cost is even less when a more complex scene is rendered.

Using rank-1 lattices for texture storage non-axis aligned structures are visually more pleasing when the textures are magnified and due to the good isotropic properties of MMD rank-1 lattices the quality does not depend largely on the orientation of the basis vectors. Nevertheless the basis vectors define the preferred directions for straight lines as in any lattice. For texture minification our numerical results clearly show the improved sampling efficiency over regular lattices.

7. Conclusions and Future Work

We have presented a new image and texture representation, which easily can be integrated in ray tracing and real-time rendering systems. For that purpose we have introduced an addressing scheme based on the lattice point indices, as apparent in the rank-1 lattice definition. Thereby the natural structure of rank-1 lattices is exploited. The periodicity of rank-1 lattice allows for a simple tiled image representation, with the number n of lattice points per tile being independent of the original image resolution. Therefore n always can be chosen freely, according to the application need, as a prime number or a power of 2 for example. Note that even though most of the examples were on square regions maximized minimum distance rank-1 lattices can be constructed for any aspect ratio and thus used for non-square textures as well without changing any of the computations.

Thanks to a higher sampling efficiency rank-1 lattice images provide a better image quality than images on the square lattice at same storage cost. In fact, the sampling efficiency of maximized minimum distance rank-1 lattices very closely approximates the one of the hexagonal lattice. We showed that for a subset of the rank-1 lattices they are in fact equal to the hexagonal lattice for texturing.

Using the proposed addressing scheme neighboring lattice points are not necessarily neighbors in the image array on disk, which can influence cache performance. In Section 5.4 we already indicated beneficial choices of generator vectors, however, an investigation with respect to cache performance still needs to be performed.

While for general isotropic textures the rank-1 lattices outperform the traditional square lattice, textures containing mostly vertical and horizontal lines can not be represented smoothly. In this case the rendering system should allow for traditional texturing. We have not yet investigated embedded lattices that would allow for classical image pyramids. Thus our multiresolution textures need to be always computed from the full resolution source image.

As rank-1 lattices are not restricted to the isotropic spectrum, they are also able to catch anisotropic structures by choice of

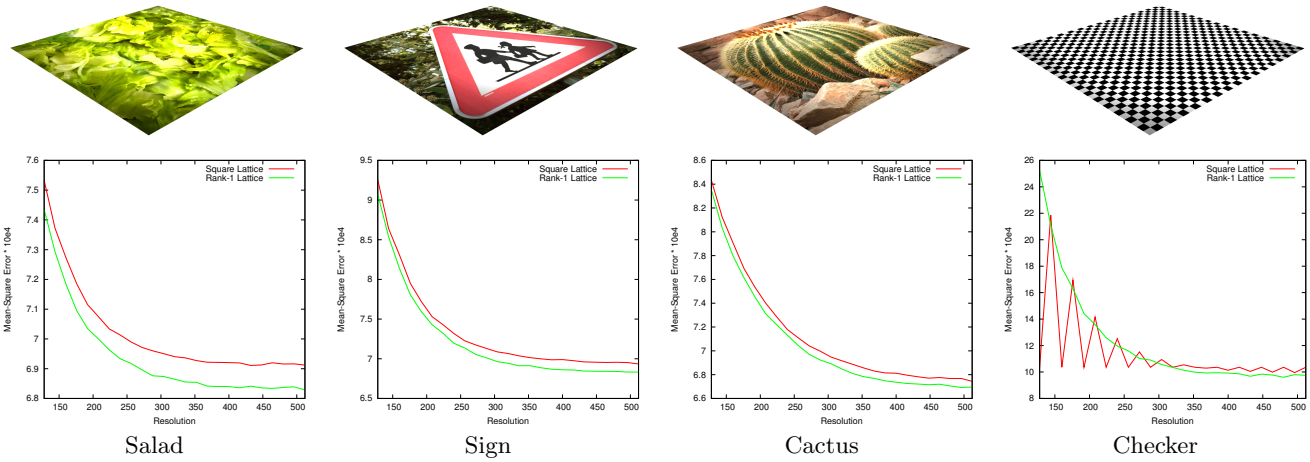


Figure 8: Results of the comparison of square lattice textures to rank-1 lattice textures. The top row shows the reference images and the bottom row the scaled mean square error at different resolutions.

a different generator vector, which could be taken advantage of for anisotropic textures. In the future we would like to study texture synthesis based on Fourier transform and the implementation of image processing algorithms, such as edge detection, using the rank-1 texture framework.

Acknowledgements

The first two authors would like to thank mental images GmbH for funding and support of this work. This work has been partially funded by the DFG Emmy Noether fellowship (Le 1341/1-1).

A. Proof for the rank-1 lattice addressing scheme.

Due to the periodicity of the lattice

$$\mathbf{b}_{1,j} = \begin{cases} \{k \cdot g_j\}_n & \text{if } \{k \cdot g_j\}_n \leq n - \{k \cdot g_j\}_n \\ \{k \cdot g_j\}_n - n & \text{otherwise} \end{cases} \quad (4)$$

$$\mathbf{b}_{2,j} = \begin{cases} \{l \cdot g_j\}_n & \text{if } \{l \cdot g_j\}_n \leq n - \{l \cdot g_j\}_n \\ \{l \cdot g_j\}_n - n & \text{otherwise} \end{cases} \quad (5)$$

holds for the basis vectors for $j \in \{1, 2\}$. We first consider the case for $\{k \cdot g_j\}_n \leq n - \{k \cdot g_j\}_n$ and $\{l \cdot g_j\}_n \leq n - \{l \cdot g_j\}_n$ in equations (4) and (5). Then for the basis vectors $\mathbf{b}_1$ and $\mathbf{b}_2$ we have

$$\mathbf{b}_1 = \mathbf{x}_k = \{k \cdot \mathbf{g}\}_n \quad (6)$$

$$\mathbf{b}_2 = \mathbf{x}_l = \{l \cdot \mathbf{g}\}_n. \quad (7)$$

Writing $\mathbf{x}_i$ as a linear combination of the basis vectors and inserting equations (6) and (7) for $\mathbf{b}_1$ and $\mathbf{b}_2$ yields

$$\begin{aligned} \mathbf{x}_i &= (s \cdot \mathbf{b}_1 + t \cdot \mathbf{b}_2) = \{s \cdot \mathbf{x}_k + t \cdot \mathbf{x}_l\}_n \\ &= \{s \cdot \{k \cdot \mathbf{g}\}_n + t \cdot \{l \cdot \mathbf{g}\}_n\}_n \quad (8) \\ &= \{s \cdot k \cdot \mathbf{g} + t \cdot l \cdot \mathbf{g}\}_n \\ &= \{(s \cdot k + t \cdot l) \cdot \mathbf{g}\}_n = \{\{(s \cdot k + t \cdot l)\}_n \cdot \{\mathbf{g}\}_n\}_n, \end{aligned}$$

where we use the rules from modular arithmetic that $\{a+b\}_n = \{\{a\}_n + \{b\}_n\}_n$ and $\{a \cdot b\}_n = \{\{a\}_n \cdot \{b\}_n\}_n$ [Ste01]. Finally, by comparison of coefficients we have $i = \{s \cdot k + t \cdot l\}_n$, as $\mathbf{x}_i = \{i \cdot \mathbf{g}\}_n$. Now let

$$\mathbf{b}_1 = \{k \cdot \mathbf{g}\}_n - n \text{ and } \mathbf{b}_2 = \{l \cdot \mathbf{g}\}_n - n. \quad (9)$$

In this case

$$\begin{aligned} \mathbf{x}_i \cdot n &= (s \cdot \mathbf{b}_1 + t \cdot \mathbf{b}_2) \\ &= \{s \cdot (\{k \cdot \mathbf{g}\}_n - n) + t \cdot (\{l \cdot \mathbf{g}\}_n - n)\}_n \\ &= \{s \cdot \{k \cdot \mathbf{g}\}_n - s \cdot n + t \cdot \{l \cdot \mathbf{g}\}_n - t \cdot n\}_n \quad (10) \\ &= \{s \cdot \{k \cdot \mathbf{g}\}_n + t \cdot \{l \cdot \mathbf{g}\}_n\}_n. \quad (11) \end{aligned}$$

As $-s \cdot n \bmod n = 0$ and $-t \cdot n \bmod n = 0$, equation (10) yields in equation (11), which already has been covered in equation (8).

The remaining two cases for

$$\mathbf{b}_1 = \{k \cdot \mathbf{g}\}_n, \quad \mathbf{b}_2 = \{l \cdot \mathbf{g}\}_n - n \quad (12)$$

and

$$\mathbf{b}_1 = \{k \cdot \mathbf{g}\}_n - n, \quad \mathbf{b}_2 = \{l \cdot \mathbf{g}\}_n \quad (13)$$

can be shown in a similar way.

B. Proof of Equivalence of Hexagonal Lattices and Rank-1 Lattices for Texturing when $n = 2M_mF_m$.

Lattices used for texturing have to tile $[0, 1)^2$. The hexagonal lattice tiles the plane, but not $[0, 1)^2$ for any uniform scaling factor $\mathbf{s}$ due to its irrational basis $B_{\mathrm{H}} = \begin{pmatrix} \binom{1}{0} & \binom{\frac{1}{2}}{\frac{\sqrt{3}}{2}} \end{pmatrix}$. Assuming n_x hexagonal elements in x-direction (i.e. the basis is scaled by $\frac{1}{n_x} B_{\mathrm{H}}$) at most $n_y = \left\lfloor \frac{2n_x}{\sqrt{3}} \right\rfloor$ hexagonal elements in y-direction fit. Non-uniform rescaling in y-direction with a factor $s_y = \frac{\frac{2n_x}{\sqrt{3}}}{\left\lfloor \frac{2n_x}{\sqrt{3}} \right\rfloor}$ allows for perfect tiling. The basis scaling factor is $\mathbf{S} = \begin{pmatrix} \binom{\frac{1}{n_x}}{0} & \binom{0}{s_y} \end{pmatrix}$. This factor in fact removes the irrational value from the basis $(\mathbf{S} \cdot B_{\mathrm{H}})$.

For the lattices constructed from the continued fraction of $\sqrt{3}$ (see Section 3.2), the generator vector is $\mathbf{g} = \binom{M_m}{F_m}$ for a lattice with $n = 2M_mF_m$ points. Computing on $[0, n)^2$, we construct a basis from the generator vector by the following observation: On the x- (i.e. $y = 0$) axis we have F_m points at a distance of $2M_m$, because $y_i = (iF_m) \bmod (2M_mF_m) = 0 \Leftrightarrow i = 2M_m$. Now we select as the first basis vector $\mathbf{b}_1 = \binom{2M_m}{0}$. As the second basis vector we choose the point with index $i = 1$: $\mathbf{b}_2 = \binom{M_m}{F_m}$. This is surly a basis of the rank-1 lattice because $b_1 \times b_2 = 2M_mF_m = n$.

This rank-1 lattice has a basis of the form $B_L = \mathbf{S}_\mathrm{r}\left(\binom{1}{0}\binom{\frac{1}{2}}{1}\right)$, with $\mathbf{S}_\mathrm{r} = \left(\binom{\frac{1}{n_x}}{0}\binom{0}{\frac{1}{n_y}}\right)$, and is equivalent to a hexagonal lattice scaled to tile the square with $n_x = F_m$ and $n_y = 2M_m$. This is true for all lattices with $n = 2F_mM_m$.

References

[AG85] Afflerbach L., Grothe H.: Calculation of Minkowski-reduced Lattice Bases. *Computing 35*, 3-4 (1985), 269–276.

[CR97] Cools R., Reztsov A.: Different Quality Indexes for Lattice Rules. *J. Complex. 13*, 2 (1997), 235–258.

[CSB87] Conway J., Sloane N., Bannai E.: *Sphere-packings, Lattices, and Groups.* Springer-Verlag New York, Inc., 1987.

[Csb05] Csbfalvi B.: Prefiltered Gaussian reconstruction for high-quality rendering of volumetric data sampled on a body-centered cubic grid. In *IEEE Visualization* (2005), IEEE Computer Society, p. 40.

[CVF08] Condat L., Van De Ville D., Forster-Heinlein B.: Reversible, fast, and high-quality grid conversions. *IEEE Transactions on Image Processing 17*, 5 (May 2008), 679–693.

[DK08] Dammertz S., Keller A.: Image Synthesis by Rank-1 Lattices. In *Monte Carlo and Quasi-Monte Carlo Methods 2006*, Keller A., Heinrich S., Niederreiter H., (Eds.). Springer, 2008, pp. 217–236.

[Edw07] Edwards D.: *Practical Sampling for Ray-Based Rendering.* PhD thesis, University of Utah, 2007.

[EVM08] Entezari A., Ville D. V. D., Möller T.: Practical box splines for reconstruction on the body centered cubic lattice. *IEEE Transactions on Visualization and Computer Graphics 14*, 2 (2008), 313–328.

[Hel85] Helfrich B.: Algorithms to construct Minkowski-reduced and Hermite-reduced lattice bases. *Theor. Comput. Sci. 41*, 2-3 (1985), 125–139.

[HW81] Hua L., Wang Y.: *Applications of Number Theory to Numerical Analysis.* Springer-Verlag and Science Press, Berlin-New York, and Beijing, 1981.

[Kan87] Kannan R.: Algorithmic Geometry of Numbers. *Annual Reviews in Computer Science 2* (1987), 231–267.

[Kor59] Korobov N.: The Approximate Computation of Multiple Integrals. *Dokl. Akad. Nauk SSR 124* (1959), 1207–1210 (in Russian).

[Llo82] Lloyd S.: Least Squares Quantization in PCM. *IEEE Transactions on Information Theory 28*, 2 (1982), 129–137.

[Mar03] Martinet J.: *Perfect Lattices in Euclidean Spaces.* Springer-Verlag, 2003.

[MS05] Middleton L., Sivaswamy J.: *Hexagonal Image Processing: A Practical Approach (Advances in Pattern Recognition).* Springer-Verlag New York, Inc., 2005.

[Nie92] Niederreiter H.: *Random Number Generation and Quasi-Monte Carlo Methods.* SIAM, Philadelphia, 1992.

[NM02] Neophytou N., Mueller K.: Space-time points: 4d splatting on efficient grids. In *VVS '02: Proceedings of the 2002 IEEE symposium on Volume visualization and graphics* (Piscataway, NJ, USA, 2002), IEEE Press, pp. 97–106.

[Pea90] Peachy D.: *Texture On Demand.* Pixar technical memo 07-06, Pixar, 1990.

[PM62] Petersen D. P., Middleton D.: Sampling and Reconstruction of Wave-Number-Limited Functions in N-Dimensional Euclidean Spaces. *Information and Control 5*, 4 (1962), 279–323.

[Sie89] Siegel C.: *Lectures on Geometry of Numbers.* Springer-Verlag, 1989.

[SJ94] Sloan I., Joe S.: *Lattice Methods for Multiple Integration.* Clarendon Press, Oxford, 1994.

[Ste01] Steger A.: *Diskrete Strukturen (Band 1).* Springer-Verlag, 2001.

[VVPL02] Van De Ville D., Van De Walle R., Philips W., Lemahieu I.: Image Resampling Between Orthogonal and Hexagonal Lattices. In *Proceedings of the 2002 IEEE International Conference on Image Processing* (2002), vol. III, pp. 389–392.