Some Useful Properties of the Permutohedral Lattice for Gaussian Filtering

Jongmin Baek and Andrew Adams

Stanford University

The aim of this self-contained supplement is to rigorously establish the mathematical properties of the permutohedral lattice, to adapt it for the use in Gaussian filtering, and to further justify its use. For discussion of a CPU implementation, applications and empirical comparison to other methods, see [ABD10].

The rest of the paper is organized as follows: in Section 1, we give preliminary definitions of relevant terms and notation; in Section 2, we formally define the permutohedral lattice in several ways and prove useful structural properties stemming from the definition; in Section 3, we suggest criteria for picking the ideal lattice with which to perform Gaussian filtering, and present an argument that the permutohedral lattice is the most appropriate choice within a large family of lattices; in Section 4, we discuss how to perform each step of the Gaussian filter using the permutohedral lattice. Finally, in Section 5, we analyze the overall algorithm and compare it to the implementation on $\mathbb{Z}^d$ in [PD09].

1. Preliminaries

We begin by declaring the notation used hereafter in Table 1. Other standard notation such as $\mathbb{R}, \mathbb{Z}$ applies.

d	The dimensionality of the underlying space.
$\vec{x}$	A vector of dimensionality $d+1$, unless specified otherwise: $\{x_0, x_1, \ldots, x_d\}$.
$\vec{1}$	The $d+1$-dimensional vector $\{1, 1, \ldots, 1\}$.
H_d	The d-dimensional hyperplane $\{\vec{x} \mid \vec{x} \cdot \vec{1} = 0\} \subset \mathbb{R}^{d+1}$.
A_d^*	The permutohedral lattice of dimensionality d, lying inside H_d.
$T(\cdot)$	The projection mapping $\mathbb{R}^{d+1}$ onto H_d along the vector $\vec{1}$. (See Proposition 2.3.)
$N(\vec{x}; \theta)$	Multi-variate Gaussian of covariance matrix θI, evaluated at $\vec{x}$.
$\vec{u}_k$	The k-th standard basis vector, indexed from zero, i.e. $\vec{u}_0 = \{1, 0, 0, \ldots, 0\}$.

Table 1: *Common notation used in the paper.*

Now we may review the basic definitions that will be useful in the subsequent sections. Our first definition is, not surprisingly, the standard definition of a lattice:

Definition 1.1. A *lattice* L is a discrete additive subgroup of a Euclidean space.

An additive subgroup of a Euclidean space is closed under addition and subtraction, and contains the origin. Hence, one can alternatively characterize a lattice as all linear combinations of a set of vectors called the *frame* (akin to the basis of a vector space) with integer weights. For example, the lattice $\mathbb{Z}^d$ corresponds to the frame consisting of standard basis vectors $\vec{u}_0, \vec{u}_1, \ldots$.

The fact that a lattice is an additive group immediately gives rise to many useful properties, including translational symmetry; the local structure of a lattice around each lattice point is invariant. A typical way to study the local structure of any discrete point set is to examine its Voronoi cells:

Definition 1.2. Let $\vec{x}$ be a point in a lattice $L \subset \mathbb{R}^d$. Then the *Voronoi cell* of $\vec{x}$ is the set of points in $\mathbb{R}^d$ nearer to $\vec{x}$ than to any other lattice point, i.e.

$$\left\{ \vec{y} \in \mathbb{R}^d \;\middle|\; \forall \vec{x'} \in L,\ \|\vec{x} - \vec{y}\|^2 \leq \|\vec{x'} - \vec{y}\|^2 \right\}.$$

It is clear that the Voronoi cells of a lattice are uniform and related to one another by translation. They also tessellate the underlying space $\mathbb{R}^d$, since any point in $\mathbb{R}^d$ can be associated with a lattice point nearest to it.

Definition 1.3. A *Delaunay cell* of a lattice is the convex hull of all lattice points whose Voronoi cells share a common vertex.

The Delaunay cells are the dual of the Voronoi cells, and yield a natural notion of "nearby" lattice points for an arbitrary point in space. They also tessellate the space, but they need not be uniform. Figure 1 demonstrates the Voronoi and Delaunay cells for a pair of two-dimensional lattices.

Note that $\mathbb{R}^d$ in the above definitions can be replaced with any set isometric to $\mathbb{R}^d$. For instance, H_d is a d-dimensional hyperplane in $\mathbb{R}^{d+1}$ and is therefore isometric to $\mathbb{R}^d$. Any discrete additive subgroup of H_d is a lattice, and its Voronoi and Delaunay cells resides in H_d. (This is the case with the permutohedral lattice, as we shall see shortly in Section 2.)

2. The Permutohedral Lattice

The permutohedral lattice has seen applications in many fields ranging from crystallography to communication. It arises ubiquitously in nature; its two-dimensional and three-dimensional instances are the hexagonal honeycomb—see

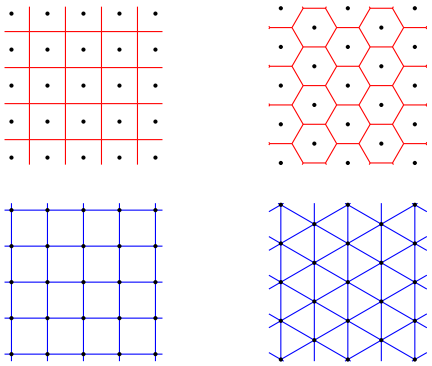

Figure 1: *Voronoi and Delaunay tessellations of some two-dimensional lattices. Top: the Voronoi cells of $\mathbb{Z}^2$ and A_2^* are shown, along with lattice points. The two lattices have equal density. Bottom: the Delaunay cells induced by the Voronoi cells above. All cells of $\mathbb{Z}^2$ are squares, whereas A_2^* has hexagonal Voronoi cells and triangular Delaunay cells.*

Figure 1—and the face-centered cubic lattice seen in the packing structure of diamond, respectively. We will define the permutohedral lattice in several ways and derive its structural properties.

2.1. Definition

Traditionally, the permutohedral lattice is defined as the dual of the root lattice A_d. Note that the dual of a lattice is not the vertices of its Voronoi cells, as typically intuited; see Definition 2.2. Both A_d and the permutohedral lattice are typically embedded in $H_d \subset \mathbb{R}^{d+1}$ for ease of manipulation.

Definition 2.1. The root lattice A_d is

$$\left\{ \vec{x} = (x_0, \ldots, x_d) \in \mathbb{Z}^{d+1} \;\middle|\; \vec{x} \cdot \vec{1} = 0 \right\}.$$

In other words, $A_d = \mathbb{Z}^{d+1} \cap H_d$. Quite clearly, $A_d \subset H_d$.

Definition 2.2. The *permutohedral lattice*, denoted by A_d^*, is the dual lattice of A_d inside H_d:

$$A_d^* := \left\{ \vec{x} \in H_d \;\middle|\; \forall \vec{y} \in A_d,\ \vec{x} \cdot \vec{y} \in \mathbb{Z} \right\}. \tag{2.1}$$

For example, A_2^* is spanned by $\left(\frac{1}{3}, \frac{1}{3}, -\frac{2}{3}\right)$ and $\left(\frac{1}{3}, -\frac{2}{3}, \frac{1}{3}\right)$.

We shall state two equivalent definitions of A_d^* scaled up by $d+1$ in each dimension. The scale factor ensures that all coordinates will be integers, and these definitions more clearly elucidate the properties of the lattice.

Proposition 2.3. The following two definitions of A_d^* are equivalent to (2.1) scaled up by $d+1$:

$$A_d^* := \left\{ T(\vec{x}) \;\middle|\; \vec{x} \in (d+1)\mathbb{Z}^{d+1} \right\}, \tag{2.2}$$

where T is the projection of $\mathbb{R}^{d+1}$ onto H_d, namely

$$T : \vec{x} \mapsto \vec{x} - \left(\frac{\vec{x} \cdot \vec{1}}{\vec{1} \cdot \vec{1}} \right) \vec{1}.$$

$$A_d^* := \bigcup_{k=0}^{d} \{ \vec{x} \in H_d \mid \vec{x} \text{ is a remainder-}k \text{ point.} \}, \tag{2.3}$$

where we call $\vec{x} \in H_d$ a *remainder-k* point for some $k \in \{0, \ldots, d\}$ iff all coordinates are congruent to k modulo $d+1$.

Proof. (2.2)$\subseteq$(2.3): Let $\vec{x} \in (d+1)\mathbb{Z}^{d+1}$. Then we can write $\vec{x} = (d+1)\vec{y}$ for some $\vec{y} \in \mathbb{Z}^{d+1}$. This yields,

$$\begin{aligned} T(\vec{x}) &= (d+1)\vec{y} - \frac{(d+1)(\vec{y} \cdot \vec{1})}{d+1} \vec{1} \\ &= (d+1)\vec{y} - (\vec{y} \cdot \vec{1})\vec{1}. \end{aligned}$$

We see that each component of $T(\vec{x})$ is an integer and has a consistent remainder modulo $d+1$, namely $(\vec{y} \cdot \vec{1})$.

(2.3)$\subseteq$(2.1): Let $\vec{x}$ be a remainder-k point for some $k \in \{0, \ldots, d\}$. Then we can write $\vec{x} = (d+1)\vec{y} + k\vec{1}$ for some $\vec{y} \in \mathbb{Z}^{d+1}$. To show that $\vec{x}/(d+1)$ is in the dual of A_d, note that for all $\vec{z} \in A_d$,

$$\begin{aligned} \frac{\vec{x}}{d+1} \cdot \vec{z} &= \left(\vec{y} + \frac{k}{d+1} \vec{1} \right) \cdot \vec{z} \\ &= \vec{y} \cdot \vec{z} + \frac{k}{d+1} \vec{z} \cdot \vec{1} \\ &= \vec{y} \cdot \vec{z} \in \mathbb{Z}, \qquad \text{since } \vec{z} \cdot \vec{1} = 0. \end{aligned}$$

(2.1)$\subseteq$(2.2): Let $\vec{x}/(d+1)$ be in the dual lattice of A_d. Note that for all $i \neq j$, the vector $\vec{u}_i - \vec{u}_j$ belongs to A_d, because $(\vec{u}_i - \vec{u}_j) \cdot \vec{1} = 0$. Hence,

$$\frac{\vec{x}}{d+1} \cdot (\vec{u}_i - \vec{u}_j) = \frac{x_i}{d+1} - \frac{x_j}{d+1} \in \mathbb{Z}.$$

This implies that all pairs of coordinates x_i, x_j differ by multiples of $d+1$. Writing $x_i = (d+1)y_i + k$ for some $y_i \in \mathbb{Z}, k \in \mathbb{R}$, we see that $\vec{x}$ is the projection of $(d+1)\vec{y}$ obtained by subtracting a multiple of $(1, \ldots, 1)$. Because $\vec{x} \in H_d$, this projection must be equivalent to T. □

According to Proposition 2.3, the simplest way to characterize A_d^* is to project $\mathbb{Z}^{d+1}$ along the long diagonal vector $\vec{1}$, with a scalar factor of $d+1$ to keep the coordinates in $\mathbb{Z}$, as seen in Figure 2. From now on, we will assume the scaled version of A_d^*, i.e. (2.2) or (2.3), rather than the original (2.1).

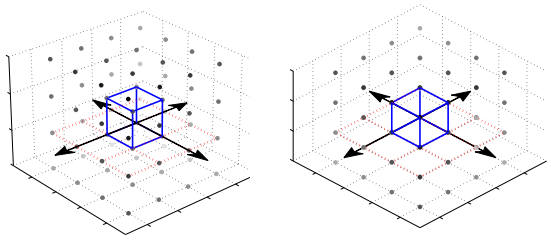

Figure 2: *Construction of A_d^* in case $d = 2$. Left: a regular grid. The xy-plane is shown in red, along with the unit cube in blue. Right: the same figure in orthographic projection, seen along the vector $\vec{1}$. Note that many of the points line up, indicating that they will project to the same point on H_d. The set of projected points form A_d^*, with appropriate scaling.*

2.2. Structural Properties

As with other lattices, A_d^* induces a tessellation of H_d with its Voronoi cells. The Voronoi cells of A_d^* are polytopes called *permutohedra* because they are obtained by permuting the coordinates of any single vertex. The name of these polytopes also lends itself to this lattice.

Proposition 2.4. [CS98] The Voronoi cell of the origin in A_d^* is a permutohedron with vertices

$$\left\{ \rho\left(\frac{d}{2}, \frac{d-2}{2}, \ldots, \frac{-d+2}{2}, \frac{-d}{2}\right) \;\middle|\; \rho \in S_{d+1} \right\},$$

where S_{d+1} is the symmetric group on $d+1$ elements, i.e. the set of all permutations.

The permutohedra in the first three dimensons are a line segment, a hexagon (Figure 1) and a uniform truncated octahedron (Figure 3), respectively.

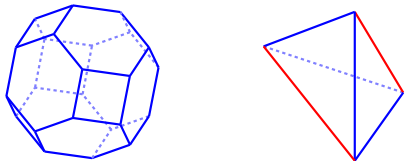

Figure 3: *Left: the Voronoi cell of A_3^* (a permutohedron in $\mathbb{R}^3$), which is also the uniform truncated octahedron. Right: the Delaunay cell of A_3^*, which is an isosceles tetrahedron. The red sides are longer than the blue sides.*

Theorem 2.5. The Delaunay cells of A_d^* are d-simplices, and are related via permutation and translation to the *canonical simplex* whose vertices are the following:

$$\begin{gathered}
(0,0,\ldots,0,0,0),\\
(1,1,\ldots,1,1,-d),\\
\vdots\\
(\underbrace{k,\ldots,k}_{d+1-k},\underbrace{k-(d+1),\ldots,k-(d+1)}_{k}),\\
\vdots\\
(d,-1,\ldots,-1,-1,-1).
\end{gathered}$$

Proof. Because of the translational symmetry in A_d^*, it suffices to characterize only the Delaunay cells containing the origin. Let $\vec{x}$ be the vertex shared by a set of Voronoi cells, one of which is the Voronoi cell of the origin. By Proposition 2.4, $\vec{x}$ is some permutation of $(\frac{d}{2}, \frac{d-2}{2}, \ldots, \frac{-d}{2})$. WLOG, assume $\vec{x} = (\frac{d}{2}, \frac{d-2}{2}, \ldots, -\frac{d}{2})$.

The Delaunay cells are defined by the lattice points closest to $\vec{x}$. By Proposition 2.3, lattice points in A_d^* are remainder-k points, having the form $\vec{y} = (d+1)\vec{z} + k\vec{1}$ where $\vec{z} \in \mathbb{Z}^{d+1}$. Since $\vec{y} \cdot \vec{1} = 0$, consequently $\vec{z} \cdot \vec{1} = -k$.

By choice of $\vec{z}$, the distance between $\vec{x}$ and $\vec{y}$ is minimized. In particular,

$$\begin{aligned}
\operatorname{argmin}_{\vec{z}} \|\vec{y} - \vec{x}\|^2 &= \operatorname{argmin}_{\vec{z}} \sum |(d+1)z_i + k - x_i|^2 \\
&= \operatorname{argmin}_{\vec{z}} \sum (d+1)^2 z_i^2 + 2(d+1)z_i(k - x_i) \\
&= \operatorname{argmin}_{\vec{z}} \sum (d+1)^2 z_i^2 - 2(d+1)z_i x_i \\
&= \operatorname{argmin}_{\vec{z}} \|(d+1)\vec{z} - (d/2, \ldots, -d/2)\|^2 \\
&= (\underbrace{0,\ldots,0}_{d+1-k}, \underbrace{-1,\ldots,-1}_{k}).
\end{aligned}$$

Plugging this expression back into $\vec{y} = (d+1)\vec{z} + k\vec{1}$, we obtain $\vec{y} = (k, \ldots, k, k-(d+1)\ldots, k-(d+1))$, showing that there is a unique nearest lattice point of remainder k. Next, we can check in a straightforward manner that the nearest remainder-k points for all k are all equidistant from $\vec{x}$. Hence, varying k over $\{0, \ldots, d\}$ yields the canonical simplex as stated above.

If $\vec{x}$ is some permutation ρ of $\left(\frac{d}{2}, \ldots, \frac{-d}{2}\right)$, the above derivation commutes fully with ρ, yielding a simplex that is related to the canonical simplex via ρ. □

Corollary 2.6. Each Delaunay cell contains exactly one lattice point of remainder k, for each $k \in \{0, \ldots, d\}$.

Proof. By construction, the canonical simplex contains exactly one lattice point of remainder k. Because other simplices are obtained by permuting the components of its vertices and translating them, the same holds. □

In fact, we can significantly strengthen Corollary 2.6:

Corollary 2.7. For every $\vec{x} \in H_d$, the Delaunay cell containing it is the set of the closest remainder-k point for each $k \in \{0, \ldots, d\}$.

To demonstrate this, we first show a lemma characterizing points near an arbitrary lattice point:

Lemma 2.8. (The Range Lemma) Let $\vec{x} \in H_d$, and let $\vec{y} \in A_d^*$ be a remainder-k point. Then, $\vec{y}$ is the closest remainder-k point to $\vec{x}$ iff

$$\max_i (x_i - y_i) - \min_i (x_i - y_i) \leq d + 1.$$

Proof. Because of translational symmetry, it suffices to prove the claim for $\vec{y} = (0, \ldots, 0)$. By inspection, all vertices of the canonical simplex obey the following inequalities:

$$x_0 \geq x_1 \geq x_2 \geq \cdots \geq x_d, \text{ and } x_0 - x_d \leq d + 1. \tag{2.4}$$

Since any point in the simplex is a convex combination of the vertices, the above inequality must also hold for any point inside. In particular, for any of the $d+1$ inequalities in (2.4), all but one vertex of the canonical simplex satisfy it with equality. Since a linear inequality is a $(d-1)$-dimensional hyperplane, which is completely determined by d points, it must be that the face formed by the d vertices corresponds to the hyperplane. Therefore, the region bounded by the faces (i.e. the simplex) corresponds exactly to the set of points satisfying (2.4).

Hence, for any $\vec{x}$ in the canonical simplex, $\max_i x_i - \min_i x_i = x_0 - x_d \leq d + 1$. Likewise, for any simplex that is a permutation of the canonical simplex, we have $x_j - x_k \leq d + 1$ for some j, k where $x_j = \max_i x_i$ and $x_k = \min_i x_i$. □

Corollary 2.7 then follows: if $\vec{x}$ is in the canonical simplex, we can easily check that the vertices given in Theorem 2.5 satisfy the premise of the Range Lemma. Therefore, they must be the closest remainder-k points to $\vec{x}$ as desired. Note that the expression $\max_i(x_i - y_i) - \min_i(x_i - y_i)$ is invariant under permutation of the components or translation of $\vec{x}$ and $\vec{y}$, so the argument applies to other simplices in A^*d.

Corollary 2.7 is of main interest to us because it suggests an efficient way to calculate the vertices of the Delaunay cell containing an arbitrarily specified point. It suffices to simply find the closest remainder-k point for each $k \in \{0, \ldots, d\}$.

Lemma 2.9. [CS98] Given $\vec{x} \in H_d$, the closest remainder-0 points in A_d^* can be found with the following procedure:

1. Define $\vec{y}$ where y_i is the multiple of $d+1$ closest to x_i.
2. Compute the difference $\vec{\Delta} = \vec{x} - \vec{y}$, and $h = \dfrac{\sum_{i=0}^{d} y_i}{d+1}$.
3. If $h < 0$, add $d+1$ to each y_j where Δ_j is one of the $|h|$-largest differences. If $h > 0$, subtract $d+1$ from each y_j where Δ_j is one of the h-smallest differences.

Proof. Note that the set of remainder-0 points in A_d^* is equivalent to A_d scaled up by $d+1$. Conway gives an analogous algorithm for finding the nearest lattice point in A_d with proof [CS98]. □

Theorem 2.10. Given $\vec{x} \in H_d$, the Delaunay cell containing it can be found by the following procedure in $O(d^2)$:

1. Find the nearest remainder-0 point $\vec{y}$.
2. Compute the differential $\vec{\Delta} = \vec{x} - \vec{y}$.
3. Find the permutation ρ that sorts $\vec{\Delta}$ in decreasing order.
4. For each k, the nearest remainder-k point is given by $\rho^{-1}(\vec{v}_k) + \vec{y}$ where $\vec{v}_k$ is the remainder-k point in the canonical simplex.

Proof. The theorem follows directly from Lemma 2.9 and Theorem 2.5. Each step runs in $O(d \log d)$ or $O(d^2)$, so the overall time complexity is $O(d^2)$. □

In summary, we have explored the structure of the permutohedral lattice in terms of its Voronoi and Delaunay cells, along with useful results on how to quickly locate nearby lattice points or verify them.

We end the section by briefly discussing the volumes of the Voronoi and Delaunay cells, as it will be necessarily in the subsequent sections:

Proposition 2.11. The volume of a permutohedron is given by $(d+1)^{d-1/2}$. The volume of the canonical simplex is $(d+1)^{d-1/2}/d!$.

Proof. By inspection, A_d^* is spanned by any d of the permutations of $(1, 1, \ldots, -d)$. The volume of the parallelopiped constructed by a basis is given by the square root of its Gram determinant, which can be easily computed to be $(d+1)^{d-1/2}$. The volume of the permutohedron must equal this number, because both the parallelopiped and the permutohedron tessellate the space with equal density, namely the density of the lattice points in space. Lastly, a simple counting argument shows that there are $d!$ simplices for every permutohedron. □

3. Evaluating Lattices for High-Dimensional Filtering

There are various spatial data structures for accelerating high-dimensional Gaussian filtering, such as the bilateral grid [CPD07], the Gaussian KD-tree [AGDL09] and point clusters [GS91]. However, their efficacy can be limited by poor scaling behavior in both time and memory, numerical instability and complexity.

These approaches bifurcate into two branches: one relies on discretization of data and predicable placements of points—a prime example is a lattice. The latter is more data-dependent in its organization, e.g. trees and clusters. We shall restrict ourselves to the former class. The use of the permutohedral lattice can then be motivated by examining how a Gaussian filter might be applied on lattices and identifying useful criteria.

3.1. Framework for Arbitrary Lattices

We first review the bilateral filter on a regular grid $\mathbb{Z}^d$, generalizing to an arbitrary lattice L at the same time. This allows us to identify properties that are critical for efficient and accurate Gaussian filtering. In doing so, we adopt the terminology of [AGDL09] for the steps in Gaussian filtering: *splatting*, *blurring* and *slicing*. For more detailed overview of Gaussian filtering on particular data structures, see [CPD07] [PD09] [AGDL09] [YDGD03].

Splatting

The first step in Gaussian filtering is to *splat*, or embed, the input signal into a higher-dimensional Euclidean space. For each sample of the input signal, its value is accumulated at one or more spatially nearby lattice points belonging to L. In case only one point in L is chosen, the process is effectively equivalent to quantization. Choosing multiple points with nonnegative weights that sum to 1 corresponds to an interpolation (for example, bilinear weights in 2D), and generally improves accuracy.

For $\mathbb{Z}^d$, the standard multi-linear weights can be used to splat the sample point into 2^d positions. More generally, for arbitrary lattices, the sample can be splatted onto the vertices of the Delaunay cell containing it, with an appropriate scheme to assign weights. See Figure 4 for examples.

Blurring

The value stored at each location is then blurred with those at nearby locations in L, with weights that decay as the Gaussian of their L_2-distance. This step is referred to as *blurring*, as it accomplishes blurring of the data held at the lattice points. In case of $\mathbb{Z}^d$, its separable nature enables us to blur each dimension in isolation, requiring access only to neighbors along each axis. For other lattices, we may consider as neighbors all vertices that share at least one Delaunay cell with the given vertex. Unfortunately, most lattices are not separable, complicating this process.

Slicing

Slicing is the process of sampling the data structure at the original input locations, and is exactly analogous to splat-

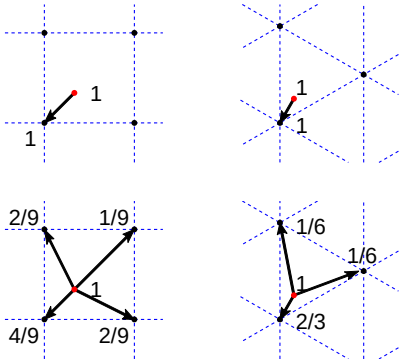

Figure 4: *The splatting process on $\mathbb{Z}^2$ and A_2^*. The Delaunay tessellations are shown in dotted guidelines. Top: a sample point is rounded to the nearest lattice point. Bottom: a sample point is spread onto the vertices of the Delaunay cell containing it, using bilinear coordinates for $\mathbb{Z}^2$ and barycentric coordinates for A_2^*.*

Lattice	# of vertices in Delaunay cell
A_d	$\binom{d+1}{k}$ for $k \in \{1,\ldots,d\}$
A_d^*	$d+1$
$B_d = B_d^* = \mathbb{Z}^d$	2^d
D_d	$2d$ or $2^{d-1}+1$
D_d^*	$4^{\lfloor d/2 \rfloor}$
E_6	27
E_7	8 or 56
E_8	9 or 16
F_4	8
G_2	3

Table 2: *Classification of irreducible root lattices in finite dimensions. The dual lattices are also given for ones that generalize across dimensions. The right-hand column gives the number of vertices in the Delaunay cells; some lattices may have more than one type of Delaunay cells that occur. Note that A_d^* is the only family of root lattices that generalizes across d, tessellates the space with a uniform Delaunay cell, and features a small number of vertices in each cell.*

ting: each sample of the output signal is reconstructed from nearby positions, with the same set of weights as in splatting.

3.2. Criteria for the Ideal Lattice

The framework in the previous section guides the designation of appropriate criteria for selecting the ideal lattice for Gaussian filtering. In forming the criteria, we concern ourselves with numerical accuracy, computational efficiency and generality:

- (Translational symmetry) For every point $\vec{x} \in \mathbb{R}^d$, the splatting, blurring and slicing kernels are spatially invariant. This also ensures that the calculations will be uniform across the space, simplifying the algorithms required and ensuring numerical stability. By definition, lattices exhibit this property.
- (Even distribution) The distribution of the points in space is as even and isotropic as possible. This requirement is in place to ensure that the kernels will resemble the Gaussian as much as possible.
- (Fast splat and slice) Given $\vec{x} \in \mathbb{R}^d$, one must be able to quickly and systematically locate nearby lattice points, i.e. find the Delaunay cell containing $\vec{x}$. The number of vertices in the Delaunay cell should scale slowly with d.
- (Fast blur) Given an arbitrary lattice point, one must be able to quickly and systematically locate nearby lattice points with which to blur. We prefer that the blurring step be separable if at all possible.
- (Generality) There should exist an analogous construction of discrete sets for each dimensionality d.

3.3. Finding the Ideal Lattice

The classification of lattices in arbitrary dimensions is an open research problem, and is beyond the scope of this paper. However, it does happen that an important class of lattices can be constructed from a rather simple set of lattices called *root lattices*. For instance, all significant lattices in 3D crystallograpy (the 14 Bravais lattices) can be decomposed into root lattices and their duals up to affine transform.

Theorem 3.1. (Witt, 1941) [CS98] Let L be an integral lattice, meaning that $\forall \vec{x},\vec{y} \in L$, $\vec{x}\cdot\vec{y} \in \mathbb{Z}$. Then, all sublattices of L generated by vectors of norm 1 and 2 are direct sums of root lattices.

Formally, root lattices are lattices generated by the roots of semisimple Lie algebras [Ser87]. Fortunately, irreducible root lattices in finite dimensions have been fully enumerated.

Theorem 3.2. [Sam69] [Ser87] There exist exactly the following irreducible root lattices:

$$A_d, B_d, D_d, E_6, E_7, E_8, F_4 \text{ and } G_2.$$

Note that B_d is equivalent to the standard lattice $\mathbb{Z}^d$.

We may then restrict our search to these irreducible root lattices and their duals. The Voronoi and Delaunay cells of irreducible root lattices are given in [MP92]. Those of the dual lattices can be found without much difficulty. Table 2 lists the root lattices along with their duals, and the number of vertices in each of their Delaunay cells. This number must be small in order to support fast splatting and slicing, and the lattice should generalize across dimensions. As shown, only A_d^*, the permutohedral lattice, meets these criteria. (In fact, $d+1$ is the fewest number of vertices possible for any d-dimensional Delaunay cell.)

Moreover, A_d^* has been shown to exhibit very efficient packing; it is the best known lattice for sphere covering up to $d = 22$ [CS98]. The problem of sphere covering seeks to place uniform spheres at lattice point to cover the entire space, while minimizing the relative density of the lattice. This loosely indicates that the distribution of neighbors of a lattice point in A_d^* is as even and isotropic as possible.

Finally, we claim that A_d^* enables fast blurring by being separable in a weak sense, as will be demonstrated in Sec-

tion 4.2. We have already shown that nearest lattice points are easily located in A_d^*.

In summary, the permutohedral lattice A_d^* is the only root lattice or dual of a root lattice that meets the criteria spelled out in Section 3.2. It is computationally efficient (fast splatting, blurring and slicing), exhibits good packing behavior, and generalizes across dimensions. While this does not rule out lattices not covered by Witt's Theorem from being superior, it does strongly suggest the optimality of A_d^*.

Remark 3.3. Permutohedra fall under the family of polytopes called *parallelohedra*, which are polytopes that give a face-to-face tiling of Euclidean space by translation. Parallelohedra whose Delaunay cells are simplices are called *primitive*. Voronoi proved that any primitive parallelohedon is affinely equivalent to the Voronoi cell of some integral lattice [Vor52] [Dol07] [Rys98]. This justifies our focus on lattices, which are translation-invariant, and in particular, on integral lattices.

4. Algorithms

Now we proceed to describe in full how to perform Gaussian filtering using the permutohedral lattice A_d^*. Each of splatting, blurring, slicing steps is discussed in detail, with pseudocode given when applicable.

4.1. Splatting

Splatting a point involves computing its neighbors in A_d^*, namely the vertices of the Delaunay cell containing it, and accumulating the value at the vertices with appropriate weights. For simplices, the most natural interpolation scheme is barycentric coordinates.

Lemma 4.1. Denote by $\vec{v}_k$ the remainder-k vertex of the canonical simplex. Let $\vec{x}$ be an arbitrary point in the canonical simplex, and let $b_0,\dots,b_d$ be its barycentric coordinates in the simplex, i.e.

$$\vec{x} = \sum_{k=0}^{d} b_k \vec{v}_k, \quad \text{and} \quad \sum_{k=0}^{d} b_k = 1.$$

Then,

$$b_k = \begin{cases} \dfrac{x_{d-k} - x_{d+1-k}}{d+1}, & k \neq 0, \\ 1 - \dfrac{x_0 - x_d}{d+1}, & k = 0. \end{cases}$$

Proof. Because the vertices of a (non-degenerate) simplex span the underlying Euclidean space, a barycentric decomposition exists and is unique. Thus it suffices to show that the given weights do yield $\vec{x}$. Let $\vec{y} = \sum_{k=0}^{d} b_k \vec{v}_k$. Then,

$$\begin{aligned} y_j &= \sum_{k=0}^{d} b_k \cdot (v_k)_j \\ &= \left[\sum_{k=0}^{d-j} b_k k\right] + \left[\sum_{k=d-j+1}^{d} b_k (k-(d+1))\right] \\ &= \left[\sum_{k=0}^{d} b_k k\right] - \left[(d+1)\sum_{k=d-j+1}^{d} b_k\right] \\ &= \left[\left(\frac{x_{d-1}-x_d}{d+1}\right) + 2\left(\frac{x_{d-2}-x_{d-1}}{d+1}\right) + \cdots \right. \\ &\qquad \left. + d\left(\frac{x_0 - x_1}{d+1}\right)\right] - \left[(d+1)\sum_{k=d-j+1}^{d} b_k\right] \\ &= \frac{-x_d - x_{d-1} - \cdots - x_1 + d x_0}{d+1} - (x_0 - x_j) \\ &= \frac{-x_d - x_{d-1} - \cdots - x_1 - x_0}{d+1} + x_j \\ &= x_j, \end{aligned}$$

as desired. Lastly, the weights $b_0,\dots,b_d$ sum to 1 by construction. □

Lemma 4.1 easily generalizes to arbitrary points in H_d by utilizing the fact that other simplices are given by permutation and translation of the canonical simplex.

Proposition 4.2. Let $\vec{x}$ be an arbitrary point in H_d, and let $\vec{v}_k$ be the remainder-k point in the Delaunay cell containing $\vec{x}$, for each $k \in \{0,\dots,d\}$. Also, let $\rho \in S_{d+1}$ be the permutation that sorts the components of $\vec{\Delta} = \vec{x} - \vec{v}_0$ in decreasing order. Then, the barycentric coordinates for $\vec{x}$ is given by $\vec{b}$ where

$$b_k = \begin{cases} \dfrac{\rho(\vec{\Delta})_{d-k} - \rho(\vec{\Delta})_{d+1-k}}{d+1}, & k \neq 0, \\ 1 - \dfrac{\rho(\vec{\Delta})_0 - \rho(\vec{\Delta})_d}{d+1}, & k = 0. \end{cases}$$

Proof. The claim follows from Lemma 4.1 and Theorem 2.10. □

Hence the barycentric coordinates of a point in H_d can be computed in $O(d \log d)$, and if ρ and $\vec{v}_0$ are already available, in $O(d)$.

The splatting step is complete once every sample from the input signal has been splatted. Algorithm 1 summarizes the process. Note that at each lattice point, nearby samples from the input are accumulated with weights that depend on the offset vector $\vec{\Delta}$. So the splatting process is a convolution with a kernel in H_d (followed by sampling at the lattice points.)

Algorithm 1 The splatting algorithm. Given the input signal $I : X \to \mathbb{R}^n$, where X is the discrete subset of H_d, it embeds the signal in A_d^* and returns the result.

Require: $I : X \to \mathbb{R}^n$.
  $V(\vec{y}) \leftarrow 0, \quad \forall \vec{y} \in A_d^*$
  for all $\vec{x} \in X$ **do**
    $\vec{y} \leftarrow$ the closest remainder-0 point to $\vec{x}$ (Lemma 2.9)
    $\vec{\Delta} \leftarrow \vec{x} - \vec{y}$
    $\rho \leftarrow$ the permutation that sorts $\vec{\Delta}$ in decreasing order
    for $k = 0$ to d **do**
      $\vec{v}_k \leftarrow$ the closest remainder-k point (Lemma 2.10)
      $b_k \leftarrow$ the barycentric coordinate (Proposition 4.2)
      $V(\vec{y}) \leftarrow V(\vec{y}) + b_k I(\vec{x})$
    end for
  end for
  return $V : A_d^* \to \mathbb{R}^n$

Proposition 4.3. The splatting process is equivalent to convolution with the following kernel:

$$K_s : \vec{x} \mapsto 1 - \frac{\max_i x_i - \min_i x_i}{d+1}.$$

Proof. It suffices to examine the values accumulated at the origin. For each Delaunay cell containing the origin, which is obtained by permuting the dimensions of the canonical simplex, the weight associated to the origin is $1 - \frac{\rho(\vec{x})_0 - \rho(\vec{x})_d}{d+1}$, where ρ sorts $\vec{x}$ in decreasing order. (See Proposition 4.2.) Hence $K_s(\vec{x}) = 1 - \frac{\max_i x_i - \min_i x_i}{d+1}$ as desired. □

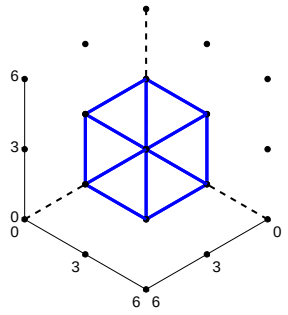

Figure 5: *A plot of the 3-dimensional cube $[0,3]^3$, along with points in $3\mathbb{Z}^3$. The plot is oriented such that $(1,1,1)$ is normal to the plane of the figure. Note that the flattening of the cube along $(1,1,1)$ yields the union of six simplices, which equals the support for K_s for $d = 2$.*

The splatting kernel K_s can be described as a tent whose support is the union of the Delaunay cells meeting at the origin. There is also a surprisingly elegant way of characterizing K_s, based on one of the definitions of A_d^* in Section 2.1.

Proposition 4.4. Let $T : \mathbb{R}^{d+1} \to H_d$ be the projection onto H_d as before. Then

$$K_s(\vec{x}) = \frac{1}{(d+1)\sqrt{d+1}} \lambda\left(\{\vec{y} \in [0, d+1]^{d+1} \mid T(\vec{y}) = \vec{x}\}\right),$$

where λ is the 1-dimensional Lebesgue measure, i.e. the length of the set. Equivalently, K_s is the density of a uniform cube $[0, d+1]^{d+1}$ after it is flattened onto H_d, up to a multiplicative factor.

Proof. The set $\{\vec{y} \in [0, d+1]^{d+1} \mid T(\vec{y}) = \vec{x}\}$ contains elements of the form $\vec{y} = \vec{x} + q \cdot \vec{1}$. Since $\vec{y} \in [0, d+1]^{d+1}$, it must be that $0 \le x_i + q \le d+1$ for all i. This is equivalent to

$$-\min_i x_i \le q \le d + 1 - \max_i x_i.$$

The set is a linear segment parallel to $\vec{1}$, so its measure is given by its projected length along any of the dimensions multipled by $\|\vec{1}\| = \sqrt{d+1}$. Its projected length is simply the extent of q, which is $d + 1 - (\max_i x_i - \min_i x_i)$. Therefore,

$$\begin{aligned}
&\frac{1}{(d+1)\sqrt{d+1}} \lambda\left(\{\vec{y} \in [0, d+1]^{d+1} \mid T(\vec{y}) = \vec{x}\}\right) \\
&\quad = \frac{1}{(d+1)\sqrt{d+1}} \sqrt{d+1}(d + 1 - (\max_i x_i - \min_i x_i)) \\
&\quad = 1 - \frac{\max_i x_i - \min_i x_i}{d+1} \\
&\quad = K_s(\vec{x}) \qquad \text{by Proposition 4.3,}
\end{aligned}$$

as desired. □

See Figure 5 for illustration of Proposition 4.4. This "unflattened" representation facilitates numerical analysis of the splatting kernel, as seen in the following two results.

Corollary 4.5.

$$\int_{H_d} K_s(\vec{x}) d\vec{x} = (d+1)^{d - \frac{1}{2}}.$$

Proof. We can "unproject" K_s back into a cube with uniform density, and the total mass would remain constant. Hence it becomes $\frac{1}{(d+1)\sqrt{d+1}} (d+1)^{d+1} = (d+1)^{d-\frac{1}{2}}$ as desired. □

Proposition 4.6. The variance of K_s is $\frac{d(d+1)^2}{12}$.

Proof. By definition,

$$\mathrm{Var}(K_s) = \frac{\int_{H_d} \|\vec{x}\|^2 K_s(\vec{x}) d\vec{x}}{\int_{H_d} K_s(\vec{x}) d\vec{x}}.$$

The denominator is given by Corollary 4.5. For the numerator, the sum of $\|\vec{x}\|^2$ over H_d weighted by the density of K_s is equivalent to an unweighted sum of $\|T(\vec{y})\|^2$ over $[0, d+1]^{d+1}$. Hence,

$$\begin{aligned}
\mathrm{Var}(K_s) &= \frac{\int_{[0,d+1]^{d+1}} \|T(\vec{y})\|^2 \frac{1}{(d+1)\sqrt{d+1}} d\vec{y}}{(d+1)^{d-1/2}} \\
&= \frac{1}{(d+1)^{d+1}} \int_{[0,d+1]^{d+1}} \|T(\vec{y})\|^2 d\vec{y} \\
&= \frac{1}{(d+1)^{d+1}} \int_{[0,d+1]^{d+1}} \left\| \vec{y} - \frac{\sum y_i}{d+1} \vec{1} \right\|^2 d\vec{y} \\
&= \frac{1}{(d+1)^{d+1}} \int_{[0,d+1]^{d+1}} \sum y_i^2 + \frac{(\sum y_i)^2}{d+1} - \frac{2(\sum y_i)^2}{d+1} d\vec{y} \\
&= \frac{1}{(d+1)^{d+1}} \int_{[0,d+1]^{d+1}} \frac{d \sum y_i^2}{d+1} - \frac{\sum_{i \ne j} y_i y_j}{d+1} d\vec{y} \\
&= \frac{1}{(d+1)^{d+1}} \sum_i \int_0^{d+1} (d+1)^d \frac{d \cdot y_i^2}{d+1} dy_i \\
&\quad - \frac{1}{(d+1)^{d+1}} \sum_{i \ne j} \iint_0^{d+1} (d+1)^{d-1} \frac{y_i y_j}{d+1} dy_i dy_j \\
&= \frac{d(d+1)^2}{3} - \frac{d(d+1)^2}{4} \\
&= \frac{d(d+1)^2}{12},
\end{aligned}$$

as desired. □

4.2. Blurring

A standard multi-variate Gaussian blur in a d-dimensional space is separable into d Gaussian blurs along each of the d independent axes. The advantage of using $\mathbb{Z}^d$ as the lattice is that the choice of the d axes is straightforward, and that each of the d directional blurs is simple: for all $\vec{x} \in \mathbb{R}^d$, $\vec{x}$ can be blurred with points of the form $\vec{x} \pm q \cdot \vec{u}_k$, where $q \in \mathbb{Z}$. For A_d^*, however, the choice of d principal axes in H_d with such property is not clear. In fact, there is no natural set of d independent axes that align with the spatial arrangement of points in A_d^*. (See Figure 1.)

Recall Proposition 2.3, which states A_d^* is obtained by applying the projection T on $(d+1)\mathbb{Z}^{d+1} \in \mathbb{R}^{d+1}$. The original space is spanned by the vectors in the set $\{(d+1)\vec{u}_k\}$. Because T is linear, we can equivalently state that A_d^* is spanned by their projections $\vec{w}_k$, i.e.

$$\vec{w}_k := \frac{T(\vec{u}_k)}{\|T(\vec{u}_k)\|} = \frac{1}{\sqrt{d(d+1)}}(\underbrace{-1,\ldots,-1}_{k},d,-1,\ldots,-1).$$

The set $\{\vec{w}_k\}$ represents a natural choice of $d+1$ axes along which to blur.

Of course, $\{\vec{w}_k\}$ is not linearly independent, since its cardinality exceeds the rank of the space. Nevertheless, $d+1$ directional blurs along these axes compose to form a standard multi-variate Gaussian blur in H_d, as shown in the following proposition.

Proposition 4.7. Define $\mathcal{G}_k$ to be the operator that performs Gaussian blur along $\vec{w}_k$ with variance θ:

$$\mathcal{G}_k(f) : \vec{x} \mapsto \int_{\mathbb{R}} f(\vec{x} - t \cdot \vec{w}_k) N(t;\, \theta) dt,$$

where $f : H_d \to \mathbb{R}$. Then, $\mathcal{G} = \mathcal{G}_0 \circ \cdots \circ \mathcal{G}_d$ performs the standard multi-variate Gaussian blur in H_d with variance $\theta\frac{d}{d+1}$.

Proof. Composing $\mathcal{G}_0, \ldots, \mathcal{G}_k$ yields

$$\mathcal{G}(f) : \vec{x} \mapsto \int_{\mathbb{R}^{d+1}} f\left(\vec{x} - \sum t_k \cdot \vec{w}_k\right) N(\vec{t};\, \theta) d\vec{t}.$$

$\mathcal{G}$ is a sequence of convolutions, and thus corresponds to a convolution kernel $K_b : H_d \to \mathbb{R}$:

$$\mathcal{G}(f) : \vec{x} \mapsto \int_{H_d} f(\vec{x} - \vec{y}) K_b(\vec{y}) d\vec{y}.$$

Equating the two expressions above, we obtain

$$K_b(\vec{y}) = \int_{\mathbb{R}^{d+1}} N(\vec{t};\, \theta)\delta\left(\sum t_k \cdot \vec{w}_k - \vec{y}\right) d\vec{t}.$$

Since the equation $\sum t_k \cdot \vec{w}_k = \vec{y}$ is underconstrained by a single dimension, the set of solutions has one degree of freedom. By inspection, $\vec{t} = \vec{s} + q \cdot \vec{1}$ where $\vec{s} \in H_d$ is a solution and $q \in \mathbb{R}$. Hence we can write

$$\begin{aligned} K_b(\vec{y}) &= \int_{\mathbb{R}} N(\vec{s} + q \cdot \vec{1};\, \theta) \left\| \frac{d\vec{t}}{dq} \right\| dq \\ &= \int_{\mathbb{R}} N(\vec{s};\, \theta) N(q \cdot \vec{1};\, \theta) \sqrt{d+1} dq \\ &\qquad \text{because } \vec{s} \text{ and } \vec{1} \text{ are orthogonal,} \\ &= N(\vec{s};\, \theta) \int_{\mathbb{R}} N(q \cdot \vec{1};\, \theta) \sqrt{d+1} dq \\ &= N(\vec{s};\, \theta). \end{aligned}$$

Meanwhile,

$$\begin{aligned} \|\vec{y}\|^2 &= \left\|\sum s_k \cdot \vec{w}_k\right\|^2 \\ &= \sum_i {s_i}^2 \vec{w}_i \cdot \vec{w}_i + \sum_{i \neq j} s_i s_j \vec{w}_i \cdot \vec{w}_j \\ &= \sum_i {s_i}^2 - \sum_{i \neq j} \frac{s_i s_j}{d} \\ &= \frac{d+1}{d} \sum_i s_i^2 - \frac{1}{d}\left(\sum_i s_i\right)^2 \\ &= \frac{d+1}{d} \|\vec{s}\|^2. \end{aligned}$$

It follows that $K_b(\vec{y}) = N\left(\vec{y};\, \theta\frac{d}{d+1}\right)$ as claimed. □

Proposition 4.7 demonstrates that a Gaussian blur can be performed on H_d in a "separable" fashion. To adapt this approach onto A_d^*, simply define $K_b^k : H_d \to \mathbb{R}$ as the discretization of $\mathcal{G}_k$:

$$K_b^k : \vec{x} \mapsto \sum_{q=-Q}^{Q} N\left(\vec{x};\, \theta\frac{d}{d+1}\right) \cdot \delta(\vec{x} - q \cdot \vec{t}),$$

where $\vec{t} = (\underbrace{-1,\ldots,-1}_{k},d,-1,\ldots,-1)$. Note that K_b^k is nonzero only at lattice points along the projection of the k-th standard axis. See Algorithm 2 for the exact description of the steps.

Algorithm 2 The blurring algorithm. Data stored in the permutohedral lattice in the form $V : A_d^* \to \mathbb{R}^n$ is blurred with variance θ and stored in $W : A_d^* \to \mathbb{R}^n$. In practice, the domain of V and W is restricted to some finite subset $Y \subset A_d^*$.

Require: $V : A_d^* \to \mathbb{R}^n$
Require: $Q \in \mathbb{N}$
Require: $\theta \in \mathbb{R}_+$
  for $k = 0$ to d **do**
    $\vec{t} \leftarrow (\underbrace{1,\ldots,1}_{k},-d,1\ldots,1)$
    for all $\vec{x} \in A_d^*$ **do**
      $W(\vec{x}) \leftarrow \sum_{q=-Q}^{Q} V(\vec{x} + q \cdot \vec{t}) \cdot N(q \cdot \vec{t};\, \theta(d+1)/d)$
    end for
    $V \leftarrow W$
  end for
  return $I : X \to \mathbb{R}^n$

One may realize that the composition of $K_b^0, \ldots, K_b^d$ does not equal K_b, a true Gaussian blur. This is the discretization error inherent to any discrete spatial data structures such as lattices. $K_b^0 \circ \cdots \circ K_b^d$ is akin to approximating a multi-variate Gaussian kernel by sampling it at points in A_d^*, up to some discretization error.

In practice, one can use a simpler and more compact set of weights for the directional blur, e.g.,

$$W(\vec{x}) \leftarrow \frac{V(\vec{x} - \vec{t})}{4} + \frac{V(\vec{x})}{2} + \frac{V(\vec{x} + \vec{t})}{4}, \tag{4.1}$$

Equation (4.1) amounts to setting $Q = 1$ and $\theta = \frac{\ln\sqrt{2}}{(d+1)^2}$.

4.3. Slicing

The output signal can be reconstructed from the permutohedral lattice by sampling from nearby lattice points, as in splatting. The splatting algorithm, given in Algorithm 1, can be adapted with very little change.

Algorithm 3 The slicing algorithm. Given data stored in the permutohedral lattice in the form $W : A_d^* \to \mathbb{R}^n$, the output $I : X \subset H_d \to \mathbb{R}^n$ is reconstructed.

Require: $W : A_d^* \to \mathbb{R}^n$.
  $I(\vec{x}) \leftarrow 0, \qquad \forall \vec{x} \in X$
  for all $\vec{x} \in X$ **do**
    $\vec{y} \leftarrow$ the closest remainder-0 point (Lemma 2.9)
    $\vec{\Delta} \leftarrow \vec{x} - \vec{y}$
    $\rho \leftarrow$ the permutation sorting $\vec{\Delta}$ in decreasing order
    for $k = 0$ to d **do**
      $\vec{v}_k \leftarrow$ the closest remainder-k point (Lemma 2.10)
      $b_k \leftarrow$ the barycentric coordinate (Proposition 4.2)
      $I(\vec{x}) \leftarrow I(\vec{x}) + b_k W(\vec{y})$
    end for
  end for
  return $I : X \to \mathbb{R}^n$

Note that for all $\vec{x} \in X$, the closest remainder-0 point $\vec{y}$ and the permutation ρ were previously computed in the splatting algorithm. By storing these results during the splatting step, we can reduce the runtime complexity of the slicing step to $O(|X|d)$, paying storage cost of $O(|X|d)$.

The slicing process is again equivalent to a convolution. The kernel is simply K_s as before, with variance $\frac{d(d+1)^2}{12}$.

4.4. Summary

All three steps are linear in the input, so their composition can be described as a spatially-varying convolution shown in Figure 6. To control the variance of the Gaussian blur, one may scale the input signal appropriately before feeding it into the pipeline. Figure 7 visualizes the components for $d = 2$.

5. Analysis

For analysis of time and space complexity of Gaussian filtering on the permutohedral lattice, see [ABD10], which gives an implementation using a sparse permutohedral lattice built on a hash table. [ABD10] further demonstrates that it is the first known method for Gaussian filtering with runtime that is linear in the number of samples and polynomial in the dimensionality, and showcases real-time applications.

The rest of this section is concerned with measuring the numerical accuracy of the filter. Ideally, we would like our process to mimic a Gaussian blur as much as possible. Denote by $K_{\vec{x}}$ the convolution kernel used to evalute the output at $\vec{x}$, i.e. using the terminology in Figure 6,

$$I''(\vec{x}) = \int_{H_d} K_{\vec{x}}(\vec{y}) \cdot I(\vec{x}+\vec{y}) d\vec{y},$$

assuming I has already been rotated into H_d. (See the last

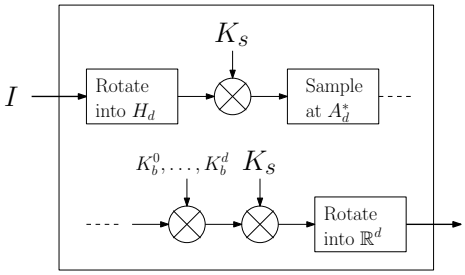

Figure 6: *Summary of Gaussian filtering with A_d^*. The input signal I resides in $\mathbb{R}^d$, and is reparametrized into H_d via an appropriate rotation, yielding a map $I' : H_d \to \mathbb{R}^n$. Each component of I' is then convolved with K_s in the splatting step, is sampled at positions in A_d^*, is convolved with $K_b^0, \ldots, K_b^d$ and K_s in the blurring step and the slicing step, respectively, and rotated back into $\mathbb{R}^d$ to yield an output signal $I'' : \mathbb{R}^d \to \mathbb{R}^n$.*

plot of Figure 7.) Analogously, let $Z_{\vec{x}}$ be the convolution kernel at $\vec{x}$ for $\mathbb{Z}^d$. For convenience, we presume that the blur kernel in each dimension is $\left[\frac{1}{4}\frac{1}{2}\frac{1}{4}\right]$, as in Equation (4.1).

Proposition 5.1.

$$Z_{\vec{x}}(\vec{y}-\vec{x}) = \prod_{i=1}^{d} \sum_{p,q \in \mathbb{Z}} B_p(x_i) B_q(y_i) b(p-q),$$

where

$$b : \vec{x} \mapsto \begin{cases} \frac{1}{2}, & x = 0, \\ \frac{1}{4}, & x = \pm 1, \\ 0, & \text{otherwise.} \end{cases}, \quad B_p : x \mapsto \max(1 - |p - x|, 0).$$

Proof. The input signal at $\vec{y}$ is splatted onto each lattice point $\vec{q}$ with multi-linear weight $\prod_{i=1}^{d} B_{q_i}(y_i)$, which is then blurred onto every other lattice point $\vec{p}$ with weight $\prod_{i=1}^{d} b(p_i - q_i)$. Then the output at $\vec{x}$ is reconstructed by summing over each lattice point $\vec{p}$ with weight $\prod_{i=1}^{d} B_{p_i}(x_i)$. The claim in the proposition then follows once we rewrite the sum of product terms as the product of sums. □

Proposition 5.2.

$$K_{\vec{x}}(\vec{y}-\vec{x}) = \sum_{\vec{p},\vec{q} \in A_d^*} B_{\vec{p}}(\vec{x}) B_{\vec{q}}(\vec{y}) b(\vec{p}-\vec{q})$$

where $B_{\vec{p}}(\vec{x})$ is the barycentric weight assigned to point $\vec{p}$ for the interpolation of $\vec{x}$, and $b(\cdot)$ is the appropriate blurring kernel given by convolving $K_b^0, \ldots, K_b^d$.

Proof. The proof is analogous to that of Proposition 5.1. □

5.1. Kernel Distance

One conceivable error metric is the squared L_2-norm between $K_{\vec{x}}$ (or $Z_{\vec{x}}$) and an isotropic Gaussian kernel, where the Gaussian kernel has the same total variance as $K_{\vec{x}}$ (or $Z_{\vec{x}}$). This measure is the continuous equivalent of the Frobenius norm for matrices, and bounds from above the relative error on arbitrary input signals. As the kernels depend on $\vec{x}$, the L_2-norm should be power-averaged over all possible $\vec{x}$. It should also be normalized by the power of the Gaussian

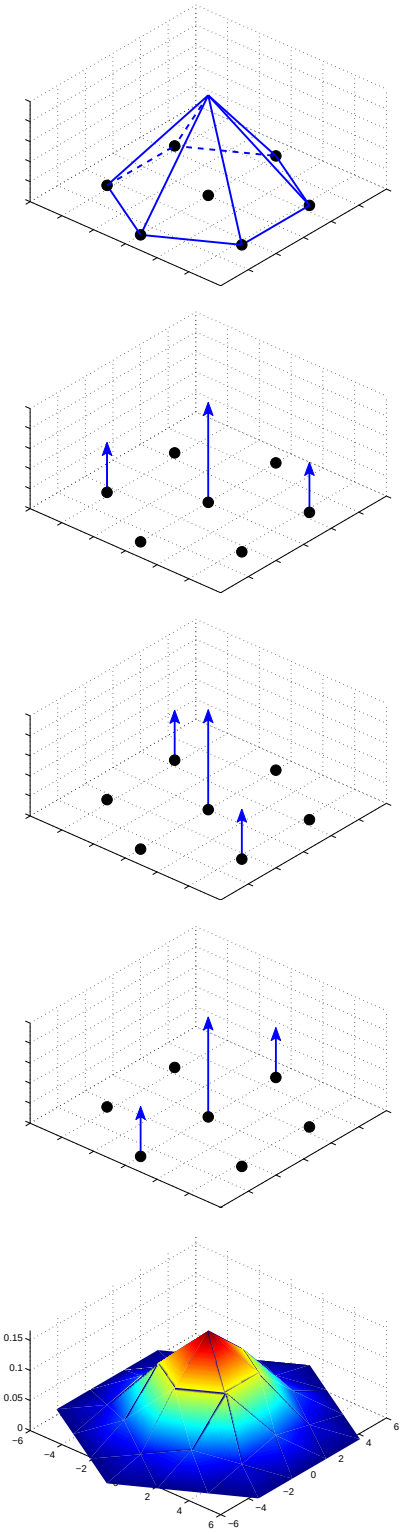

Figure 7: *Visualization of convolution kernels for each step in Gaussian filtering for $d = 2$. Top: the splatting or slicing kernel K_s. Next three: the three blur kernels K_b^0, K_b^1, K_b^2. Bottom: the overall kernel for $\vec{x} = 0$. All figures have the same scale except for the bottommost plot.*

kernel. We can consider this to be the inverse of the "signal-to-noise ratio" (SNR) where $K_{\vec{x}}$ is the observed signal and the Gaussian is the ground truth. Recall that SNR is conventionally defined as,

$$\frac{\text{Power of the signal}}{\text{Power of the noise = observation - signal}}.$$

Our definition is fully analogous, and is formalized below:

Definition 5.3. Given a family of kernels $f_{\vec{x}} : \mathbb{R}^d \to \mathbb{R}$ parametrized by $\vec{x} \in \mathbb{R}^d$ that approximates a Gaussian blur, its *signal-to-noise* ratio (SNR) is

$$\eta := \frac{\int_{\mathbb{R}^d} N(\vec{y};\theta)^2 d\vec{y}}{E_{\vec{x}}\left[\int_{\mathbb{R}^d} \left(f_{\vec{x}}(\vec{y}-\vec{x}) - N(\vec{y}-\vec{x};\,\theta)\right)^2 d\vec{y}\right]},$$

where θ is the expected variance of $f_{\vec{x}}$ over all $\vec{x}$. Also, it is assumed that $f_{\vec{x}}$ is normalized to sum to 1 over $\mathbb{R}^d$.

Lemma 5.4. The expected variance of $Z_{\vec{x}}$ where $\vec{x}$ is chosen randomly in $\mathbb{R}^d$ is $5d/6$. The expected variance of $K_{\vec{x}}$ where $\vec{x}$ is chosen randomly in H_d is $2d(d+1)^2/3$.

Proof. For $\mathbb{Z}^d$, the multi-linear splatting and slicing are simply the d-dimensional tent functions with variance $d/6$. The blur stage has variance $d \cdot \left[\frac{1}{4}(-1)^2 + \frac{1}{2}(0)^2 + \frac{1}{4}(1^2)\right] = \frac{d}{2}$. Hence the total variance is $(d/6)\cdot 2 + (d/2) = 5d/6$.

For A_d^*, Proposition 4.6 tells us that the variance of the splatting and slicing step is $d(d+1)^2/12$ each. The variance of the blurring step is,

$$(d+1)\cdot\left[\frac{1}{4}(-\sqrt{d(d+1)})^2 + \frac{1}{2}(0)^2 + \frac{1}{4}(\sqrt{d(d+1)})^2\right],$$

which equals $d(d+1)^2/2$. Together, the total expected variance is $2d(d+1)^2/3$. The term *expected* variance is used because the actual variance of a particular kernel $Z_{\vec{x}}$ or $K_{\vec{x}}$ depend on $\vec{x}$, but the additive nature of the variances of each step is clear from the proofs of Propositions 5.1 and 5.2. □

Note that the expected variance is the sum of variance over all dimensions, so it should be divided by d to yield the variance of a comparable isotropic multi-variate Gaussian.

The calculation of SNR for $Z_{\vec{x}}$ and $K_{\vec{x}}$ are given in Appendix B and Appendix C, respectively. For $Z_{\vec{x}}$, a closed form as a function of the dimensionality d can be derived; for $K_{\vec{x}}$, a recipe for numerically estimating the SNR is provided.

The SNRs for the two kernels are plotted in Figure 8. The SNRs of the two lattices are comparable, with $K_{\vec{x}}$ outperforming $Z_{\vec{x}}$ at low dimensionality and $Z_{\vec{x}}$ outperforming $K_{\vec{x}}$ at higher dimensionality. This is explained by the fact that a sample in $\mathbb{Z}^d$ is splatted to exponentially many lattice points, whereas in A_d^* it is splatted only to few points $(d+1)$. We also compare the SNR with that of true Gaussian kernels that are truncated at certain distances from the origin.

6. Conclusion

We have demonstrated interesting structural properties of the permutohedral lattice and used them to argue that the permutohedral lattice represents an appropriate choice of data

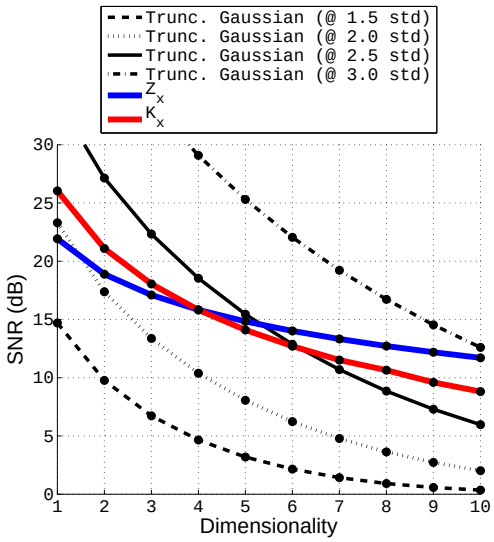

Figure 8: *The signal-to-noise ratio of the kernels $K_{\vec{x}}$ and $Z_{\vec{x}}$ for $d = 1,\ldots,10$, along with truncated Gaussian kernels. Note that $K_{\vec{x}}$ exhibits SNR that is similar to that of $Z_{\vec{x}}$, despite touching only polynomially-many vertices; $Z_{\vec{x}}$ touches exponentially many vertices. Because of this difference, the SNR for $K_{\vec{x}}$ suffers slightly more when d grows high.*

structure for performing Gaussian filtering. We have also described how to efficiently perform Gaussian filtering on the permutohedral lattice, and its numerical stability relative to that of a regular lattice.

References

[ABD10] ADAMS A., BAEK J., DAVIS M. A.: High-dimensional filtering with the permutohedral lattice. To appear, Eurographics, 2010.

[AGDL09] ADAMS A., GELFAND N., DOLSON J., LEVOY M.: Gaussian kd-trees for fast high-dimensional filtering. *ACM Transactions on Graphics (Proc. SIGGRAPH '09)* (2009), 1–12.

[CPD07] CHEN J., PARIS S., DURAND F.: Real-time edge-aware image processing with the bilateral grid. *ACM Transactions on Graphics (Proc. SIGGRAPH '07)* (2007), 103.

[CS98] CONWAY J. H., SLOANE N. J. A.: *Sphere Packings, Lattices and Groups*. Springer, 1998.

[Dol07] DOLBILIN N. P.: Minkowski's theorems on parallelohedra and their generalizations. *Russian Mathematical Surveys 62* (2007), 793–795.

[GS91] GREENGARD L. F., STRAIN J. A.: The fast gauss transform. *SIAM Journal on Scientific and Statistical Computing 12* (1991), 79–94.

[MP92] MOOD R. V., PATERA J.: Voronoi and delaunay cells of root lattices: classification of their faces and facets by coxeter-dynkin diagrams. *Journal of Physics A: Mathematical and General 25* (1992), 5089–5134.

[PD09] PARIS S., DURAND F.: A fast approximation of the bilateral filter using a signal processing approach. *Int. J. Comput. Vision 81*, 1 (2009), 24–52.

[Rys98] RYSHKOV S. S.: The structure of primitive parallelohedra and voronoi's last problem. *Russian Mathematical Surveys 53* (1998), 403–405.

[Sam69] SAMELSON H.: *Notes on Lie Algebras*. Van Nostrand Reinhold Company, 1969.

[Ser87] SERRE J.-P.: *Complex Semisimple Lie Algebras*. Springer-Verlag, 1987.

[Vor52] VORONOI G. F.: *Studies of Primitive Parallelotopes*, vol. 2. Ukrainian Academy of Sciences Press, 1952, pp. 239–368.

[YDGD03] YANG C., DURAISWAMI R., GUMEROV N. A., DAVIS L.: Improved fast gauss transform and efficient kernel density estimation. In *Proc. ICCV '03* (2003), vol. 1, pp. 664–671.

Appendix A: Useful Lemmas

The appendix contains proofs of several miscellaneous facts about simplices, the Gaussian distribution and the permutohedral lattice, used to support Appendix B and C.

Lemma A.1. Let $\mathcal{S}$ be a d-dimensional simplex defined by vertices $\vec{s}_0,\ldots,\vec{s}_d$, and let $B_0(\vec{x}),B_1(\vec{x}),\ldots,B_d(\vec{x})$ be the barycentric coordinates of $\vec{x}$ for the vertices in the given order. Then, $\forall k,j \in \{0,\ldots,d\}$,

$$\frac{\int_{\mathcal{S}} B_k(\vec{x})B_j(\vec{x})d\vec{x}}{\int_{\mathcal{S}} d\vec{x}} = \begin{cases} \dfrac{2}{(d+1)(d+2)}, & k=j, \\ \dfrac{1}{(d+1)(d+2)}, & k\neq j. \end{cases}$$

Proof. Let P be the face defined by all vertices except $\vec{s}_k$. Then $\mathcal{S}$ is a pyramid with base P and apex $\vec{s}_k$. Let h be the height of the altitude and let $\vec{y}$ be the foot of the altitude. Consider slicing $\mathcal{S}$ along a hyperplane parallel to P that contains $(1-t)\cdot\vec{s}_k + t\cdot\vec{y}$, for some $t\in[0,1]$. The slice will be similar to P with a scale factor of t, and by definition of barycentric coordinates, $B_k(\cdot)$ will evaluate to a constant, namely $1-t$, across the whole slice. Therefore,

$$\begin{aligned}
\int_{\mathcal{S}} B_k(\vec{x})^2 d\vec{x} &= \int_0^1 \int_{(1-t)\vec{s}_k+tP} (1-t)^2 d\vec{x}(h\cdot dt) \\
&= \int_0^1 (1-t)^2 \left[\int_{tP} d\vec{x}\right](h\cdot dt) \\
&= \int_0^1 (1-t)^2 t^{d-1}\mathrm{Vol(P)}h\cdot dt \\
&= \mathrm{Vol}(P)\cdot h\cdot \int_0^1 (1-t)^2 t^{d-1} dt \\
&= \mathrm{Vol}(P)\cdot h\cdot \frac{2}{d(d+1)(d+2)}. \qquad \text{(A.1)}
\end{aligned}$$

Note that the same integral without the weight term $(1-t)^2$ will simply yield the volume of $\mathcal{S}$:

$$\begin{aligned}
\int_0^1 \int_{tP} d\vec{x}(h\cdot dt) &= \mathrm{Vol}(P)\cdot h\cdot \int_0^1 t^{d-1} dt \\
&= \mathrm{Vol}(P)\cdot h\cdot \frac{1}{d}. \qquad \text{(A.2)}
\end{aligned}$$

Dividing (A.1) by (A.2) yields the first of the two cases. When $k\neq j$, we consider applying the same decomposition to P: let P' be the face of P spanned by all vertices of $\mathcal{S}$ except $\vec{s}_k$ and $\vec{s}_j$. Then P is again a pyramid with base P' and height h', and y' the foot of the altitude. Each slice of P parallel to P' cutting the altitude at $(1-r)\cdot\vec{s}_j + r\cdot\vec{y'}$ is again a copy of P' scaled by r, and the barycentric weight $B_j(\cdot)$ is

uniformly $1-r$. Therefore,

$$\begin{aligned}\int_{\mathcal{S}} B_k(\vec{x})B_j(\vec{x})d\vec{x} &= \int_0^1\int_0^1 (1-t)\cdot t(1-r)\\ &\quad\cdot\left(\mathrm{Vol}(P')t^{d-1}r^{d-2}\right)(h'\cdot dr)(h\cdot dt)\\ &= hh'\mathrm{Vol}(P')\int_0^1 t^d - t^{d+1}dt\int_0^1 r^{d-2}-r^{d-1}dr\\ &= hh'\mathrm{Vol}(P')\frac{1}{(d+1)(d+2)(d-1)d}. \qquad \text{(A.3)}\end{aligned}$$

Without the barycentric weights,

$$\begin{aligned}\int_{\mathcal{S}} d\vec{x} &= \int_0^1\int_0^1 \left(\mathrm{Vol}(P')t^{d-1}r^{d-2}\right)(h'\cdot dr)(h\cdot dt)\\ &= hh'\mathrm{Vol}(P')\int_0^1 t^{d-1}dt\int_0^1 r^{d-2}dr\\ &= hh'\mathrm{Vol}(P')\frac{1}{d(d-1)}. \qquad \text{(A.4)}\end{aligned}$$

Dividing (A.3) by (A.4) yields the second of the two cases, as desired. □

Lemma A.2. Let $\vec{x},\vec{y}\in A_d^*$. Then the number of Delaunay cells containing both $\vec{x}$ and $\vec{y}$ is given by

$$\begin{cases}(d+1-k)!k!, & \text{if } \vec{x}-\vec{y} \text{ is some permutation of the remainder-}k \text{ vertex of the canonical simplex,}\\ 0, & \text{otherwise.}\end{cases}$$

Proof. Since every Delaunay cell is given by permuting the coordinates of the canonical simplex and translating the vertices, if $\vec{x}-\vec{y}$ cannot be written as a permutation of a vertex of the canonical simplex, then they must not belong to any common Delaunay cell. So we can assume that $\vec{x}-\vec{y}$ is a permutation of some vertex of the canonical simplex.

WLOG let $\vec{x}=0$. Then any simplex containing $\vec{x}$ has a corresponding permutation in S_{d+1}. Among those, the number of simplices also containing $\vec{y}$ is the number of permutations that sends the remainder-k vertex of the canonical simplex to $\vec{y}$. Because $\vec{y}$ has $d+1-k$ coordinates of the same value, and the remaining k coordinates also have the same value, the number of possible permutations is $(d+1-k)!k!$. □

Lemma A.3. The power of a d-dimensional Gaussian kernel with covariance θI is $(4\pi\theta)^{-d/2}$.

Proof.

$$\begin{aligned}\int_{\mathbb{R}^d} N(\vec{y};\,\theta)^2 dy &= \left[\int_{-\infty}^{\infty} N(y;\,\theta)^2 dy\right]^d\\ &= \left[\int_{-\infty}^{\infty}\frac{\exp\left(-y^2/(2\theta)\right)^2}{2\pi\theta}dy\right]^d\\ &= \left[\int_{-\infty}^{\infty}\frac{\exp\left(-y^2/\left(2\cdot\frac{\theta}{2}\right)\right)}{\sqrt{2\pi\frac{\theta}{2}}\sqrt{4\pi\theta}}dy\right]^d\\ &= \left[\frac{1}{\sqrt{4\pi\theta}}\int_{-\infty}^{\infty} N(y;\,\theta/2)dy\right]^d\\ &= (4\pi\theta)^{-d/2},\end{aligned}$$

as desired. □

The next pair of lemmas illustrate how to integrate arbitrary functions over the canonical simplex with particular weighting schemes.

Lemma A.4. Let X be the union of Delaunay cells containing the origin in A_d^*, and let $f: X\to\mathbb{R}$ be a function. Then,

$$\int_X f(\vec{x})\cdot K_s(\vec{x})d\vec{x} = \frac{1}{(d+1)^{3/2}}\int_{[0,1]^{d+1}} f\left(T((d+1)\vec{x})\right)d\vec{x}.$$

Proof. This is a direct consequence of Proposition 4.4. In short, weighted sampling corresponding to K_s can be achieved by uniformly sampling in the unit cube and projecting the sample onto H_d. □

Lemma A.5. Let $f: A_d^*\to\mathbb{R}$ be a function on A_d^*. Let $b: A_d^*\to\mathbb{R}$ and the vectors $\vec{w}_k$ be as in Section 4.2. Then,

$$\sum_{\vec{q}\in A_d^*} f(\vec{q})b(\vec{q}) = \sum_{\vec{t}\in\{-1,0,1\}^{d+1}} f\left(\sum_k t_k\cdot\vec{w}_k\right)\prod_k\frac{1}{2^{1+|t_k|}}.$$

Proof. The blur kernel $b(\vec{x})$ is given by the convolution of $K_b^0,\ldots,K_b^d$. Therefore, it is the accumulation of the weights corresponding to offsets in the $d+1$ axes that sum to $\vec{x}$. The weight equals $\frac{1}{2}$ if the offset t_k in a given axis is zero, and $\frac{1}{4}$ if it is $t_k=\pm1$. Hence we can write it succinctly as $\frac{1}{2^{1+|t_k|}}$. The claim follows. □

Appendix B: Evaluation of SNR for $\mathbb{Z}^d$

The kernel $Z_{\vec{x}}$ repeats periodically because of the translational symmetry of $\mathbb{Z}^d$. In order to evaluate the expected value of an expression involving $Z_{\vec{x}}$, it suffices to consider $\vec{x}\in[0,1]^d$. In other words,

$$\eta = \frac{\int_{\mathbb{R}^d} N(\vec{y};\,5/6)^2 d\vec{y}}{\int_{[0,1]^d}\int_{\mathbb{R}^d}\left(Z_{\vec{x}}(\vec{y}-\vec{x})-N(\vec{y}-\vec{x};\,5/6)\right)^2 d\vec{y}d\vec{x}}.$$

Expanding the quadratic term in the integrand, we obtain $\eta = \frac{(B.2)}{(B.1)+(B.2)-2(B.3)}$, containing the following three expressions:

$$\int_{[0,1]^d}\int_{\mathbb{R}^d} Z_{\vec{x}}(\vec{y}-\vec{x})^2 d\vec{y}d\vec{x}, \qquad \text{(B.1)}$$

$$\int_{\mathbb{R}^d} N(\vec{y};\,5/6)^2 d\vec{y}, \qquad \text{(B.2)}$$

$$\int_{[0,1]^d}\int_{\mathbb{R}^d} Z_{\vec{x}}(\vec{y}-\vec{x})N(\vec{y}-\vec{x};\,5/6)d\vec{y}d\vec{x}. \qquad \text{(B.3)}$$

It follows from Lemma A.3 that (B.2) is $(10\pi/3)^{-d/2}$.

(B.1) is similarly separable:

$$\begin{aligned}&\int_{[0,1]^d}\int_{\mathbb{R}^d} Z_{\vec{x}}(\vec{y}-\vec{x})^2 d\vec{y}d\vec{x}\\ &= \int_{[0,1]^d}\int_{\mathbb{R}^d}\left[\prod_{i=1}^d\sum_{p,q\in\mathbb{Z}} B_p(x_i)B_q(y_i)b(p-q)\right]^2 d\vec{y}d\vec{x}\\ &= \left[\int_{[0,1]}\int_{\mathbb{R}}\left[\sum_{p,q\in\mathbb{Z}} B_p(x)B_q(y)b(p-q)\right]^2 dydx\right]^d\end{aligned}$$

$$= \left[\int_{[0,1]} \int_{\mathbb{R}} \sum_{p,p',q,q' \in \mathbb{Z}} B_p(x) B_{p'}(x) B_q(y) B_{q'}(y) \cdot b(p-q) b(p'-q') dy dx \right]^d$$

$$= \left[\sum_{p,p',q,q' \in \mathbb{Z}} b(p-q) b(p'-q') \cdot \int_{\mathbb{R}} B_q(y) B_{q'}(y) dy \int_{[0,1]} B_p(x) B_{p'}(x) dx \right]^d.$$

Note that $B_p(x)$ and $B_{p'}(x)$ are both nonzero only if $p = p'$ or $|p - p'| = 1$. In the former case, the integral of $B_p(x)B_{p'}(x)$ with respect to x over $\mathbb{R}$ is 2/3; in the latter case, 1/6. Simiarly, $B_q(y)$ and $B_{q'}(y)$ are both nonzero over $y \in [0,1]$ iff $q, q' = 0$ or 1. Setting $q = q'$ yields 1/3 for the value of the integral, and $q \neq q'$ yields 1/6. Summing over all possible values of p', q, q' with nonzero summands, we obtain that (B.1) equals

$$\left[\sum_{p \in \mathbb{Z}} \frac{b(p)^2}{2} + \frac{4b(p)b(p-1)}{9} + \frac{b(p)b(p-2)}{18} \right]^d = \left[\frac{29}{96} \right]^d.$$

The cross term (B.3) remains. We can similarly expand the expression and collect product terms into the product of sums, and exploit the fact that only few values of p, q yield nonzero summands. Unfortunately, because of the Gaussian term, there is no succinct analytic expression.

$$\begin{aligned}
\text{(B.3)} &= \int_{[0,1]^d} \int_{\mathbb{R}^d} Z_{\vec{x}}(\vec{y} - \vec{x}) N(\vec{y} - \vec{x};\ 5/6) d\vec{y} d\vec{x} \\
&= \int_0^1 \int_{\mathbb{R}^d} \prod_{i=1}^{d} \sum_{p,q \in \mathbb{Z}} B_p(x_i) B_q(y_i) \cdot b(p-q) N(y_i - x_i;\ 5/6) d\vec{y} d\vec{x} \\
&= \left[\sum_{p,q \in \mathbb{Z}} \int_0^1 \int_{\mathbb{R}} B_p(x) B_q(y) \cdot b(p-q) N(y-x;\ 5/6) dy dx \right]^d \\
&= \left[\sum_{q \in \{-1,0,1,2\}} \int_0^1 \int_{\mathbb{R}} \{(1-x)b(q) + xb(q-1)\} \cdot B_q(y) N(y-x;\ 5/6) dy dx \right]^d \\
&\simeq 0.30456.
\end{aligned}$$

Corollary B.1. The SNR term for $\mathbb{Z}^d$ is

$$\eta = \frac{\sqrt{3/(10\pi)}^d}{(29/96)^d + \sqrt{3/(10\pi)}^d - 2 \cdot 0.30456^d}$$

Appendix C: Evaluation of SNR for A_d^*

The error analysis for A_d^* is analogous to that of $\mathbb{Z}^d$, with a few subtle changes. First, $K_{\vec{x}}$ does not sum to 1 over its domain, as seen in Lemma C.1 and should be normalized before being compared to a Gaussian.

Lemma C.1.

$$\int_{H_d} K_{\vec{x}}(\vec{y} - \vec{x}) d\vec{y} = (d+1)^{d-1/2}.$$

Proof.

$$\begin{aligned}
\int_{H_d} K_{\vec{x}}(\vec{y} - \vec{x}) d\vec{y} &= \int_{H_d} \sum_{\vec{p},\vec{q} \in A_d^*} B_{\vec{p}}(\vec{x}) B_{\vec{q}}(\vec{y}) b(\vec{p} - \vec{q}) d\vec{y} \\
&\quad \text{by Proposition 5.1,} \\
&= \sum_{\vec{p},\vec{q} \in A_d^*} B_{\vec{p}}(\vec{x}) \left[\int_{H_d} B_{\vec{q}}(\vec{y}) d\vec{y} \right] b(\vec{p} - \vec{q}) \\
&= \sum_{\vec{p},\vec{q} \in A_d^*} B_{\vec{p}}(\vec{x}) (d+1)^{d-1/2} b(\vec{p} - \vec{q}) \\
&\quad \text{by Corollary 4.5,} \\
&= \sum_{\vec{p} \in A_d^*} B_{\vec{p}}(\vec{x}) (d+1)^{d-1/2} \\
&\quad \text{since } \sum_{\vec{q} \in A_d^*} b(\vec{p} - \vec{q}) = 1, \\
&= (d+1)^{d-1/2}.
\end{aligned}$$

The last line follows because the sum of splatting weights for $\vec{x}$ over all lattice points is also 1. □

$K_{\vec{x}}$, like the lattice, is translation-invariant and symmetric with respect to permutation, so it suffices to iterate $\vec{x}$ over the canonical simplex, denoted by C^*, in order to compute the expected value of the L_2 distance. Recall from Proposition 2.11 that the volume of C^* is given by $(d+1)^{d-1/2}/d!$. Therefore,

$$\eta = \frac{(d+1)^{d-1/2}/d! \int_{H_d} N(\vec{y};\ 2(d+1)^2/3)^2 d\vec{y}}{\int_{C^*} \int_{H_d} \left(\frac{K_{\vec{x}}(\vec{y} - \vec{x})}{(d+1)^{d-1/2}} - N(\vec{y} - \vec{x};\ 2(d+1)^2/3) \right)^2 d\vec{y} d\vec{x}}.$$

As before, η can be expressed as $\frac{\text{(C.2)}}{\text{(C.1)} + \text{(C.2)} - 2\text{(C.3)}}$, where the components are as follows:

$$\frac{d!}{(d+1)^{3d-3/2}} \int_{C^*} \int_{H_d} K_{\vec{x}}(\vec{y} - \vec{x})^2 d\vec{y} d\vec{x}, \tag{C.1}$$

$$\int_{H_d} N(\vec{y};\ 2(d+1)^2/3)^2 d\vec{y}, \tag{C.2}$$

$$\frac{d!}{(d+1)^{2d-1}} \int_{C^*} \int_{H_d} K_{\vec{x}}(\vec{y} - \vec{x}) \cdot N(\vec{y} - \vec{x};\ 2(d+1)^2/3) d\vec{y} d\vec{x}. \tag{C.3}$$

By Lemma A.3, (C.2) equals $\left(\frac{8(d+1)^2 \pi}{3} \right)^{-d/2}$.

Let us compute (C.1) without the constant term. The integrand may be expanded and be simplified as follows:

$$\int_{C^*} \int_{H_d} K_{\vec{x}}(\vec{y} - \vec{x})^2 d\vec{y} d\vec{x} = \int_{C^*} \int_{H_d} \left[\sum_{\vec{p},\vec{q} \in A_d^*} B_{\vec{p}}(\vec{x}) B_{\vec{q}}(\vec{y}) b(\vec{p} - \vec{q}) \right]^2 d\vec{y} d\vec{x}$$

$$
\begin{aligned}
&= \int_{C^*} \int_{H_d} \sum_{\substack{\vec{p}_1,\vec{p}_2,\\ \vec{q}_1,\vec{q}_2 \in A_d^*}} B_{\vec{p}_1}(\vec{x}) B_{\vec{p}_2}(\vec{x}) B_{\vec{q}_1}(\vec{y}) B_{\vec{q}_2}(\vec{y}) \\
&\qquad \cdot b(\vec{p}_1 - \vec{q}_1) b(\vec{p}_2 - \vec{q}_2) d\vec{y} d\vec{x} \\
&= \sum_{\ldots} b(\vec{q}_1) b(\vec{q}_2) \left[\int_{C^*} B_{\vec{p}_1}(\vec{x}) B_{\vec{p}_2}(\vec{x}) d\vec{x} \right] \\
&\qquad \cdot \left[\int_{H_d} B_{\vec{p}_1 - \vec{q}_1}(\vec{y}) B_{\vec{p}_2 - \vec{q}_2}(\vec{y}) d\vec{y} \right] d\vec{y} d\vec{x}, \\
&\qquad \text{via change of variable } \vec{q}_i \leftarrow \vec{p}_i - \vec{q}_i.
\end{aligned}
$$

Our expression is a weighted sum with weights $b(\vec{q}_1)b(\vec{q}_2)$. By Lemma A.5, it is equivalent to

$$
\sum_{\vec{p}_1,\vec{p}_2 \in C^*} \sum_{\substack{\vec{t} \in \{-1,0,1\}^{d+1} \\ \vec{s} \in \{-1,0,1\}^{d+1}}} [\cdots][\cdots] \prod_k \frac{1}{2^{1+|t_k|}} \frac{1}{2^{1+|s_k|}}, \quad \text{(C.4)}
$$

where $\vec{q}_1 = \sum t_k \cdot \vec{w}_k$ and $\vec{q}_2 = \sum s_k \cdot \vec{w}_k$.

Note that the first bracketed term is effectively given by Lemma A.1. The second term is similar, but the integral is over H_d, so we must additively consider each simplex whose vertex contains both $\vec{p}_1 - \vec{q}_1$ and $\vec{p}_2 - \vec{q}_2$. The number of such simplices is given by Lemma A.2.

(C.4) is more attractive than its original form because its summands are now simple numerical quantities. There is no succinct analytic expression for the sum, but it can be computed numerically in a straightforward manner. The number of summands is $(d+1)^2 \cdot 3^{2(d+1)}$.

That leaves us with (C.3). Once again, the integral is expanded and manipulated:

$$
\begin{aligned}
&\int_{C^*} \int_{H_d} K_{\vec{x}}(\vec{y} - \vec{x}) N(\vec{y} - \vec{x};\, 2(d+1)^2/3) d\vec{y} d\vec{x} \\
&\qquad = \sum_{\vec{p},\vec{q} \in A_d^*} \int_{C^*} \int_{H_d} B_{\vec{p}}(\vec{x}) B_{\vec{q}}(\vec{y}) \\
&\qquad\qquad \cdot b(\vec{p} - \vec{q}) N(\vec{y} - \vec{x}; \ldots) d\vec{y} d\vec{x}
\end{aligned}
$$

Using Lemma A.5, we obtain

$$
\begin{aligned}
&\sum_{\vec{p} \in C^*} \sum_{\vec{t} \in \{-1,0,1\}^{d+1}} \int_{C^*} \int_{H_d} B_{\vec{p}}(\vec{x}) B_{\vec{p}+\vec{q}}(\vec{y}) \\
&\qquad \cdot \prod_k \frac{1}{2^{1+|t_k|}} N(\vec{y} - \vec{x}; \ldots) d\vec{y} d\vec{x},
\end{aligned}
$$

where $\vec{q} = \sum t_k \cdot \vec{w}_k$. By symmetry, we can swap out C^* with the union of Delaunay cells containing the origin. Then we have an integral weighted by $B_{\vec{p}}(\vec{x})$ and $B_{\vec{p}+\vec{q}}(\vec{y})$, which can be computed by Monte-Carlo sampling the $(d+1)$-dimensional unit cube, by Lemma A.4. In summary, we iterate over $\vec{p} \in C^*$ and $\vec{t} \in \{-1,0,1\}^{d+1}$, and for each pair of vectors, sample $\vec{x}$ and $\vec{y}$ via Lemma A.4, and evaluate the Gaussian of $\vec{y} - \vec{x}$, weighted by $\prod_k \frac{1}{2^{1+|t_k|}}$. Multiplying the result by an appropriate constant yields (C.4).