EUROGRAPHICS 2012 / P. Cignoni, T. Ertl
(Guest Editors)

Volume 31 (2012), Number 2

Analytic Anti-Aliasing of Linear Functions on Polytopes

T. Auzinger[1], M. Guthe[2], and S. Jeschke[1]

[1] Vienna University of Technology, Austria [2] Philipps-Universität Marburg, Germany

Abstract

This paper presents an analytic formulation for anti-aliased sampling of 2D polygons and 3D polyhedra. Our framework allows the exact evaluation of the convolution integral with a linear function defined on the polytopes. The filter is a spherically symmetric polynomial of any order, supporting approximations to refined variants such as the Mitchell-Netravali filter family. This enables high-quality rasterization of triangles and tetrahedra with linearly interpolated vertex values to regular and non-regular grids. A closed form solution of the convolution is presented and an efficient implementation on the GPU using DirectX and CUDA C is described.

Categories and Subject Descriptors (according to ACM CCS): I.3.3 [Computer Graphics]: Picture/Image Generation—Antialiasing

1. Introduction

Since the beginning of computer graphics, *rasterization* is the main method to generate images or volumes from shapes. In this process each discrete element of a raster image (a pixel) or volume (a voxel) is assigned a value depending on the shapes' boundary and optional data terms like color, that can vary across the shape. The popularity of this technique led to special purpose hardware (GPUs) that are capable of rasterizing many million triangles per second today. A fundamental problem in discretizing shape data is that practically all polygon-based representations in computer graphics are not band limited: shape boundaries constitute *discontinuities* in the signal, introducing infinitely high frequencies. From the Nyquist-Shannon sampling theorem it follows that such a signal cannot be reconstructed perfectly from discretely sampled data, so that the input signal needs to be *prefiltered* before discretization can take place.

A complete sampling process is shown in Figure 1. First, the original signal is prepared by a *presampling* filter that removes high frequencies. Then discrete samples are produced by multiplying the filter output by the sampling pattern. Finally, the reconversion is accomplished by an interpolation filter (also called *reconstruction filter*). When displaying the image at device resolution, the interpolation filter is defined by the device characteristics and the point-spread function of the human eye.

The mathematically-based argument that leads to the use

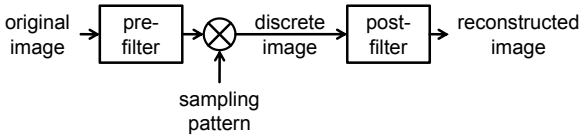


Figure 1: A sampling system.

of ideal low pass prefilters rests on the elimination of overlap in the sampled image spectrum. Aliasing produces Moiré patterns or stair-step renditions of sharp diagonal edges and similar visual defects. As shown by Schreiber and Troxel [ST85], they must be weighed against ringing that is caused by the Gibb's phenomenon, and loss of sharpness that results from removing high frequencies.

When anti-aliasing in two or more dimensions, the one-dimensional filters need to be generalized. This can either be done by separable filtering (i.e. separate filtering in each dimension) or using spherically symmetric filters. While in a numerical setting the first are computationally less expensive, visible artifacts can be caused by angle-dependant behavior or anisotropic effects. These originate from the angle-dependend frequency responses of the filter and need to be considered in addition to ringing and Moiré patterns. In case of a perfect low-pass filter a horizontal or vertical stripe pattern with a frequency slightly above the Nyquist frequency is not visible. If the pattern is rotated it becomes visible since

the Nyquist frequency is by a factor of $\sqrt{2}$ higher in the diagonal. This effect is shown in Figure 2. The anisotropy was also investigated by Mitchell and Netravali [MN88] who constrain the parameter ranges of their separable filter such that it becomes nearly isotropic. Enforcing the isotropy on a separable filter however severely limits the choice of filter functions. Especially filters with negative lobes – like the popular Lanczos filter – become strongly anisotropic when each dimension is filtered separately. We thus use the spherical generalization, which supports a wide range of possible filters.

The mathematical formulation of the problem is as follows: Given a data function $\mathcal{I}$ defined on $\mathbb{R}^n$, e.g. the color values on a triangle or in a tetrahedron, and a filter function $\mathcal{F}$, we can obtain the filtered value $v(\mathbf{x})$ at sample location $\mathbf{x}$ by evaluating the convolution

$$v(\mathbf{x}) = \int_{\mathbb{R}^n} \mathcal{I}(\mathbf{y})\mathcal{F}(\mathbf{x}-\mathbf{y})\,d\mathbf{y}. \tag{1}$$

The spherical generalization is based on the requirement that the multidimensional filter needs to show the same frequency response for one dimensional signals as the original one dimensional filter $\mathcal{F}(x)$. As the convolution is performed in two or three dimensions [Jam95], this leads to the following constraint

$$\int\int \mathcal{F}_{3D}(x,y,z)\,dy\,dz = \int \mathcal{F}_{2D}(x,y)\,dy = \mathcal{F}(x). \tag{2}$$

Consider for example the ideal low pass filter which is the sinc function in one dimension. For two and three dimensions this can be generalized to [Jam95]:

$$\mathcal{F}_{2D}(\mathbf{x}) = \frac{J_1(\pi\|\mathbf{x}\|)}{\pi\|\mathbf{x}\|}, \tag{3}$$

$$\mathcal{F}_{3D}(\mathbf{x}) = \frac{\sin(\pi\|\mathbf{x}\|)}{\pi^3\|\mathbf{x}\|^3} - \frac{\cos(\pi\|\mathbf{x}\|)}{\pi^2\|\mathbf{x}\|^2}, \tag{4}$$

where J_1 is the Bessel function of the first kind.

In this paper we show how to analytically evaluate integral 1 for a given set of 2D or 3D polytopes to produce a high-quality raster image or volume. Our method accounts for a linear data function $\mathcal{I}$ defined on these shapes, i.e., interpolated vertex values (colors, density, etc.) of triangles and tetrahedra. As filter function we assume a spherically symmetric polynomial, which allows modeling a wide range of filters and the approximation of filters which are given by transcendent functions. Our experience shows that such polynomial approximations do not result in any notable differences in the output. It can be argued that filter designs that rely on user studies and perceptional heuristics are inherently associated with an error tolerance. Sampling artifacts are the real concern and can be removed with our method. We provide efficient implementations on a GPU in DirectX and CUDA C. Analytic approaches such as ours make use of the fact that computational performance of current hardware increases much faster than memory bandwidth, which, in the long term, makes it favorable over sampling-based approaches.

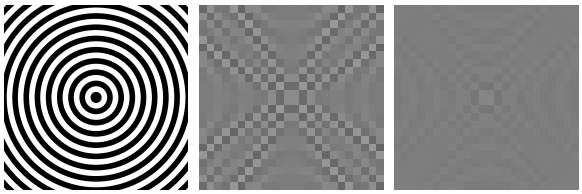


Figure 2: Circular pattern with a frequency slightly above the Nyquist frequency filtered with a Lanczos filter. Separable filtering (middle) causes anisotropy effects in the diagonals which are removed with spherical filtering (right).

2. Related Work

A considerable amount of work has been published about 2D analytic anti-aliasing. Early methods by Catmull started of with box filtering [Cat78] and were extended to spherically symmetric filters using look-up tables [Cat84]. The latter work builds on a filtering method by Feibush et al. which uses domain decomposition to evaluate the integrals [FLC80]. Pioneering works treated constant color polygons [KU81], [Cat78] and smoothly shaded polygons are mentioned as a possible extension in [Cat84]. It should be noted that [KU81] also treats the more general problem of finding an optimal filter for the given raster display. Grant [Gra85] uses a 4D formulation of this problem to model spatial and temporal anti-aliasing of translating and scaling polygons. The de-facto state of the art is still an analytic 2D filtering method by Duff [Duf89]. It supports general filter models with polynomial approximations and supports linear functions defined on the polytopes. Unfortunately, the separable 2D formulation seems not easily generalizable to three dimensions as the integral complexity grows beyond a manageable level (see section 3 and the additional material). Later, McCool [McC95] proposed simplicial decompositions of shapes for a faster filtering compared to Duff at that time, but an extension to 3D space seems not straightforward as well. Furthermore, Guenther and Tumblin [GT96] used quadrature prefiltering to further speed up the process, with analytic aliasing in one direction and sampling in the other. More recently, Lin et al. [LCSW05] proposed an analytic evaluation approach for radially symmetric filters that is in spirit similar to this work, but they do not support linear functions nor negative filter lobes as needed for most practical applications [MN88]. In addition, their implementation runs on the CPU while we describe a fully integrated GPU implementation. Note that while the filters in [MN88] were originally designed for reconstruction purposes, they are also used in sampling due to their favorable comparison to other variants.

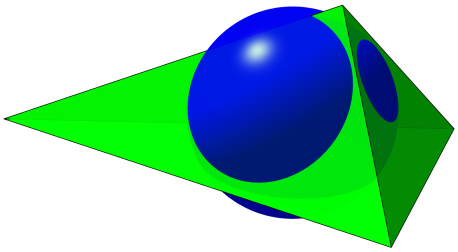


Figure 3: The intersection of a polytope with the filter support's sphere can generate complicated shapes, that have to be subdivided in order to obtain well behaved domains of integration.

Concerning three dimensional rasterization, i.e., *voxelization*, a distinction has to be made between surface voxelizations [Kau87, Jon96] and solid object voxelizations [WK93]. In addition, *binary* voxelization techniques [HYFK98] do only apply a binary value to each voxel which omits filtering issues. Several solid voxelization papers apply filtering, either based on integral lookup tables [WK93] and/or on the closest distance of each voxel center to the shape boundary [vK98, SK99] and applying so-called *oriented box filtering*. Note that all these methods do not compute the correct volumetric filter integral and do not allow for varying object functions as we do. Probably the most closely related approach to this work is a recent paper called *wavelet rasterization* [MS11]. It uses a Haar wavelet-based representation to rasterize 2D shapes (with boundaries defined by splines) to images and 3D polyhedra to volumes. Unfortunately, in its current form it is inherently restricted to (separable) box filtering and only allows for binary attributes. In contrast, this paper considers radially symmetric filters modeled by polynomial functions of varying degree (as opposed to simple box filtering), and supports linear functions defined over the polytopes to be integrated. To the best of our knowledge, an analytic formulation and evaluation of this general setup has not been attempted before.

3. Analytic Integration

In general it is not possible to derive a closed form solution for the convolution given in formula (1). It was already shown that with a set of preconditions on the data term and its support as well as on the filter term, such a solution can be obtained [Duf89]. The work of Duff [Duf89] covers analytic anti-aliasing in 2D with separable polynomial filters by arguing that the integration of a polynomial can be evaluated analytically, but they give no explicit results. In theory the method can also be extended to three dimensions, where the domains of integration can be determined by clipping the input polygons against 3D grid cells. Thus the integration boundaries of the convolution are linear functions.

While these integrals can be solved analytically, the resulting formulas become unmanageable for higher dimensions. In our work with a radially symmetrical filter each filter order can be integrated separately and produces a single expression; hence, for m filter orders we have m summands *irrespective of the dimension*. A similar procedure for a separable filter would require the evaluation of all possible monomials, i.e. $x_1^{m_1} x_2^{m_2} \cdots x_n^{m_n}$, and would yield m^n summands for m filter orders and dimension n. Obviously, this becomes unpractical and one could try to obtain a single formula for the whole integration over all filter orders. After clipping against a nD grid cell the worst case integration domain would be a polytope that is bounded by non-axis aligned hyperplanes in each dimension, i.e. the integration boundaries for coordinate x_i is of the form $a_0 + \sum_{j=1}^{i-1} a_j x_j$. The result would thus depend on $\frac{1}{2}\left(n^2+n+1\right)$ variables for the integration boundaries alone. Together with the integration of all filter orders in *each* dimension and the linear interpolation term this amounts to formulas of unmanageable sizes for three or more dimensions. See the additional material for details.

With spherically symmetric filters, our problem statement is formally as follows: Let $\mathcal{P}$ be a orientable non-self-intersecting polytope in $\mathbb{R}^n$, $\mathcal{I}_{\mathbf{a},c}(\mathbf{x}) = \mathbf{a}\cdot\mathbf{x}+c$ a linear function and $\mathcal{F}_R(\mathbf{x}) = \sum_{i=0}^{N} c_i \|\mathbf{x}\|^i \chi_{\|\mathbf{x}\|\leq R}$ a filter function with a cutoff radius R. For dimensions $n = 2, 3$ we will give a closed form solution for the convolution term

$$v(\mathbf{x}) = \int_{\mathcal{P}} \mathcal{I}_{\mathbf{a},c}(\mathbf{y})\mathcal{F}_R(\mathbf{x}-\mathbf{y})\,d\mathbf{y} \tag{5}$$

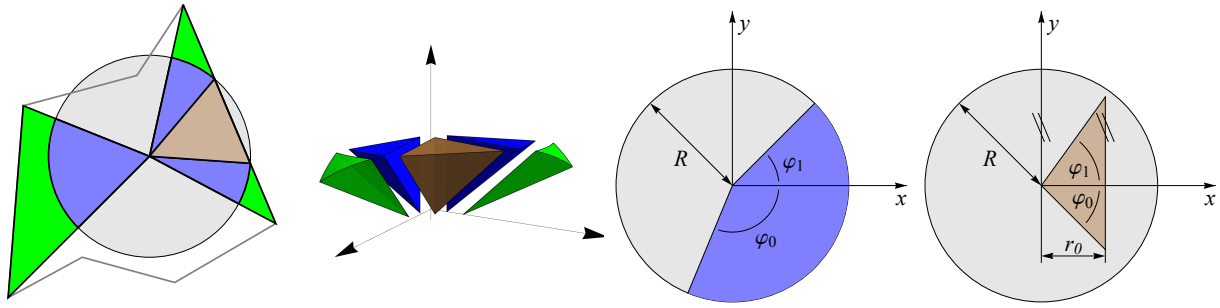
$$= \int_{\mathcal{P}\cap\mathcal{S}_R} \mathcal{I}_{\mathbf{a},c}(\mathbf{y})\mathcal{F}_\infty(\mathbf{x}-\mathbf{y})\,d\mathbf{y} \tag{6}$$

at the sample location $\mathbf{x}$. $\mathcal{S}_R$ denotes the filter function's support - a sphere with radius R. Although we restrict ourself to dimensions 2 and 3 it is possible to extend this result to higher dimensions as can be seen in the additional material.

The main difficulty in analytically computing this integral lies in its potentially complicated domain of integration, i.e. $\mathcal{P}\cap\mathcal{S}_R$. Especially in three or more dimensions the intersection of the filter's spherical support with a general polytope can produce a complicated shape on which the integrand has to be evaluated, as shown in Figure 3. Therefore, an important component of our solution is the partition of the intersection domain into a set of simple regions. Each of these regions falls into one of a small number of geometric categories for which the integral (6) can be evaluated. We furthermore use the fact that by substitution the convolution can be rewritten as

$$v(\mathbf{x}) = \int_{\mathcal{P}} \mathcal{I}_{\mathbf{a},c}(\mathbf{y})\mathcal{F}_R(\mathbf{x}-\mathbf{y})\,d\mathbf{y} = \int_{\hat{\mathcal{P}}} \mathcal{I}_{\hat{\mathbf{a}},\hat{c}}(\mathbf{y})\mathcal{F}_R(\mathbf{y})\,d\mathbf{y} \tag{7}$$

with $\mathbf{R}$ being a rotation matrix, $\hat{\mathbf{a}} = \mathbf{R}^{-1}\mathbf{a}$ and $\hat{c} = c + \mathbf{a}\cdot\mathbf{x}$. We denote by $\hat{\mathcal{P}} = \mathbf{R}(\mathcal{P}-\mathbf{x})$ the shifted and rotated version of the polytope $\mathcal{P}$. This allows us to assume the filter to be centered at the origin and enables us to align a chosen face of the shape $\mathcal{P}$ with a given (hyper)plane by rotation around the origin. In comparison to the separable case this



(a) (b) (c) (d)

Figure 4: Intersecting a polygon with the circular filter support gives three qualitatively different domains for subsequent integration (a). These domains can be immediately used for integration in the 2D case. In 3D these planar regions have to be extruded to the origin to yield the desired integration domains. The exploded view of such an extrusion of the right highlighted region of (a) is shown in (b). The marked variables for the sector (c) (resp. segment (d)) coincide with the 2D integration result in (8) (resp. in (9)).

gives us two main advantages. The filter is a function of just one coordinate, i.e. the radial direction, and the integration domains can be rotated to be 'as axis aligned as possible' thus yielding simple integration boundaries.

It should be noted that equation (7) also holds under anisotropic scaling of the space. Together with the invariance under rotation, this framework covers *elliptic* filters as well. In addition this framework accommodates *piecewise polynomial* filter functions by evaluating the convolution for different filter radii, i.e. the convolution with two polynoms $a(\mathbf{x})$ and $b(\mathbf{x})$ which are defined on $[0, r_a]$ and $[r_a, r_b]$ can be treated by the sum of the convolutions with $a(\mathbf{x}) - b(\mathbf{x})$ on $[0, r_a]$ and $b(\mathbf{x})$ on $[0, r_b]$. Hence, for n piecewise polynomials the algorithm has to be executed n times.

3.1. Integration in 2D

We first outline the integration in 2D as the 3D case builds on it. Without loss of generality we assume the polygon boundary to be oriented counterclockwise and we determine the integral contributions of all polygon edges separately (see figure 4a). This property is guaranteed by the non-self-intersection precondition on the polytope. Each edge together with the filter center at the origin spans a triangle, and thus it is possible to evaluate the integral simply by summing up the individual triangle contributions. The symbolic evaluations of the these integrals can be found in appendix A. Preserving the boundary orientation in the following intersection procedure, leads to the correct signs in the final summation, similar to [GT96].

In this setting the intersection of the filter support $\mathcal{S}_R$ with such a triangle results in a simple line-circle intersection of the triangle edge that stems from the original polygon, and a centered circle with the cutoff radius R of $\mathcal{S}_R$. Depending on the edge geometry this leads to a varying set of *segments* and *sectors*. We denote with a *segment* a triangle that has one vertex at the origin and is fully contained in $\mathcal{S}_R$ (figure 4d). In contrast, a *sector* is a circular region of $\mathcal{S}_R$ (figure 4c). As can be seen in figure 4a, boundary parts outside $\mathcal{S}_R$ result in sectors while the parts inside result in segments. Note that the same holds if the origin is not contained in the polygon as all contributions cancel correctly.

The contribution of a sector can be evaluated analytically as given in (8). Prior to integration we rotate a segment such that it lies in the positive half plane $x \geq 0$ and such that the polygon boundary is parallel to the y-axis. This is shown in figure 4d and the closed form solution is given in (9). Of course, the data function always has to be rotated together with the segment.

3.2. Integration in 3D

The 3D case is a natural extension of the 2D case. Here we assume a polyhedron with outward facing normals and all polygon faces oriented accordingly. Together with the filter center at the origin each face spans a pyramid. As before, the integration over the whole polyhedron is the sum of the contributions of all individual pyramids. Each pyramid is again rotated around the origin such that it lies in the positive half space $z \geq 0$ and its base lies in the $z = d$ plane where d denotes the pyramid height. Due to (7) this does not alter the value of the integration, as long as the data term is rotated the same way just like in the 2D case. Similarly, the orientation of the polyhedron faces are preserved up to the actual integration and the final summation.

Depending on the pyramid height d and the filter cutoff radius R, we have to consider two cases. If $d \leq R$, the pyramid's base polygon intersects the filter support $\mathcal{S}_R$, otherwise

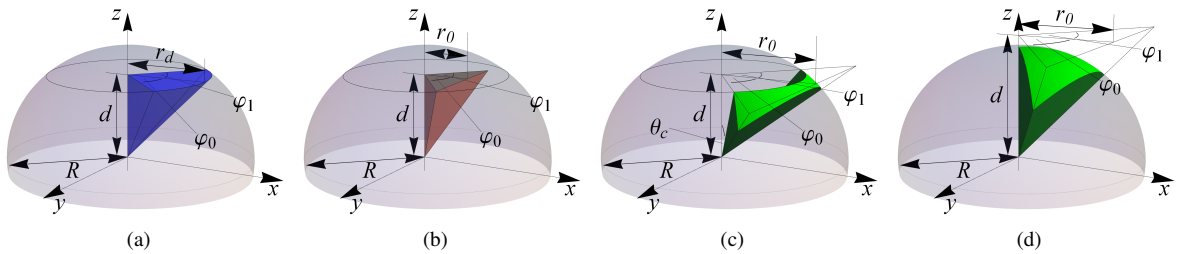


Figure 5: In 3D the domains of integration are obtained by extruding the 2D intersection domains to the origin. The extrusion of a sector (see figure 4c) gives the wedge of a cone (a) while a segment (see figure 4d) yields the tetrahedron (b). The green appendage (see figure 4a) has to be projected onto the filter support's sphere and gives a pyramid with a curved base (c). A special case of it is given in (d). Note that the color scheme is the same as in figure 4 and that domains (a) and (c) always occur in pairs.

it does not. We treat the former case first as the latter one can be treated as a special subcase of it. The intersection of the pyramid's base polygon with $\mathcal{S}_R$ is done as in the 2D case above, with the difference that we use the cutoff radius r_d at height d which amounts to $\sqrt{R^2-d^2}$. Apart from sectors and segments, a third kind of integration domain appears. In 3D the area between the line segment that lies outside the circle and the resulting sector has to be treated as well and we refer to it as *appendage* (see the green areas in figure 4a).

Since the base polygon lies at $z = d$, we have to incorporate the third dimension into our integration domain. A sector becomes the wedge of a cone, as shown in figure 5a, and its value is given in (10). Both the segment and the appendage are rotated around the z-axis such that their boundary edge opposite the vertex in $(0,0,d)$ is parallel to the y-axis. The segment together with the origin spans a tetrahedron as can be seen in figure 5b and its integration is given in (12). The appendage produces a rather different shape (figure 5c). Projecting the area of the appendage onto the boundary sphere of $\mathcal{S}_R$ towards the origin results in an area that is confined from above by the circle with center $(0,0,d)$ and radius r_d and from the sides by the great circles of the sphere with constant angle φ. The lower boundary is given by the projection of the relevant part of the polygon's edge onto the sphere. The final 3D shape given by the appendage is the 'pyramid' with apex in the origin and a curved base section given by this projection. Its integration formula in (11).

We are left with the aforementioned case where $d > R$ and the pyramid base does not intersect $\mathcal{S}_R$. Here we have no sectors nor segments but only an appendage (figure 5d). In this case we project it onto the filter support sphere $\mathcal{S}_R$ in the same way as done before and, by setting $\theta_C = 0$, we get the same result (formula (11)) as in the intersecting case.

4. Implementation

It is obvious that the evaluation of the convolution integral (6) can be done in parallel for each sampling location and each polytope, whereas the final summation can be done independently for each sampling location. This maps the problem very well to highly parallel hardware such as a GPU. We implemented the 2D case in DirectX 10 and the 3D case in CUDA C. The algorithm can be naturally divided into three stages: a setup stage, the intersection routine and a final integration. Due to the finite extent of the filter it is clear that if a sampling point $\mathbf{x}$ is placed outside the Minkowski sum of the current polytope and the filter support as shown in figure 6, the convolution integral evaluates to zero. This fact localizes computations and is used in the setup stage to generate bounding regions for each input polytope to mask irrelevant output locations.

While the mathematical framework developed in section 3 works for general polygonal and polyhedral input, we implemented the most common case of input triangles and tetrahedra. In this case a linear interpolation of vertex values coincides with the linear data function.

The DirectX implementation for the 2D case performs the setup stage in the geometry shader. Here, each incoming triangle is 'thickened' so that all pixels in the relevant region (figure 6) get rasterized. The intersection and integration stages are then computed in the pixel shader. All operations are performed in a single render pass, making the technique easy to implement in existing rendering systems.

In the CUDA implementation for the 3D case the setup stage takes each incoming tetrahedron and displaces all four faces of it outwards by the filter support radius. Similarly, the axis aligned bounding box of the tetrahedron is enlarged by the same amount. The intersection of these two figures tightly encloses the relevant region and is used as a mask in the following stages. The intersection stage takes as input a tetrahedron, a sampling location, and the spherical filter support. It generates the integration domain regions (sector, seg-

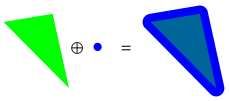


Figure 6: A tringle (left), a filter support (middle) and the Minkowski sum of the two (right).

ments and appendages) as output. Every translation and rotation that is applied to the shape has to be applied similarly to the linear function defined on that shape, as noted above (see equation (7)). The generated domains are then processed in a separate integration stage which calculates the contributions to the final sum at each sampling location. Note that since the intersection of a tetrahedron and the filter support can result in a highly varying number (9-45) of integration domains, the intersection stage can incur a significant amount of warp divergence. Depending on the hardware capabilities, warp divergence can significantly impact the overall performance if several time consuming integrations are serialized for a warp. Consequently, we buffered the domain data and decoupled the intersection and integration stages in our CUDA implemention. The final integration is then done with a separate kernel efficiently using the segment and sector buffer. We observed that the amount of computations needed for each stage hides the memory latency of the GPU.

5. Results

We show that our analytic anti-aliasing method performs reasonably well on current GPUs with an informal comparison to supersampling. Figure 7 shows a zone plane pattern consisting of 14400 colored triangles that form 80 rings with 4 degrees angular resolution. It is displayed with a Gauss filter with a radius of 2.3 pixels at a resolution of 400x400 pixels. The left image was computed using our analytic solution. It took 7.4 ms to compute on a GeForce GTX 580 graphics card using a single DirectX shader pass. The right image was computed with jittered supersampling and some effort to make this process efficient. A Halton sequence efficiently places samples and each sample contributes to all pixels were it lies in the filter range. However, in our implementation the common GPU rasterizer was used. It spaces samples regularly so that pixel samples are somewhat correlated. While this might be overcome with a CUDA implementation in theory, the induced additional bandwidth and computation might or might not compensate the benefit from sample decorrelation. Figure 7 right took a similar time to compute (7.9ms) as the left image by accumulating 576 samples in each pixel. Some aliasing clearly remains. However, this informal comparison must be taken with a grain of salt as typical scenes in entertainment applications do not show such strong aliasing as a zone plate. This makes the theoretically quite bad convergence of sampling less problematic in practice. But one general observation is that analytic approaches processes each triangle only once, so bandwidth

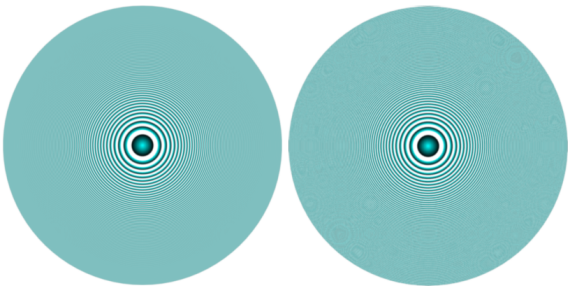


Figure 7: A zone plate image, computed with a 2.3 pixel radius Gauss filter. Left: the analytic solution. Right: the sampled solution. Both took about the same time to compute.

is traded with processing power. As the latter appears less costly compared to the former, analytic solutions might in general outperform sampling in the long run. We provide an implementation in the additional material.

Table 1 shows the time required to compute anti-aliased 3D rasterizations. We note that our approach is limited by the diverging warps in the intersection stage and not by the actual integration stage. In summary, complex tetrahedral models can be filtered and rasterized into Cartesian grids within few seconds up to few minutes.

Figure 8 shows a 3D rasterization of an extruded zone plate using area or Gaussian anti-aliasing. It is clearly visible that the more complex Gaussian filter is able to remove aliasing at a smaller filter radius while also preserving larger features much better. This emphasizes the need for sophisticated filters that were not possible with previous approaches.

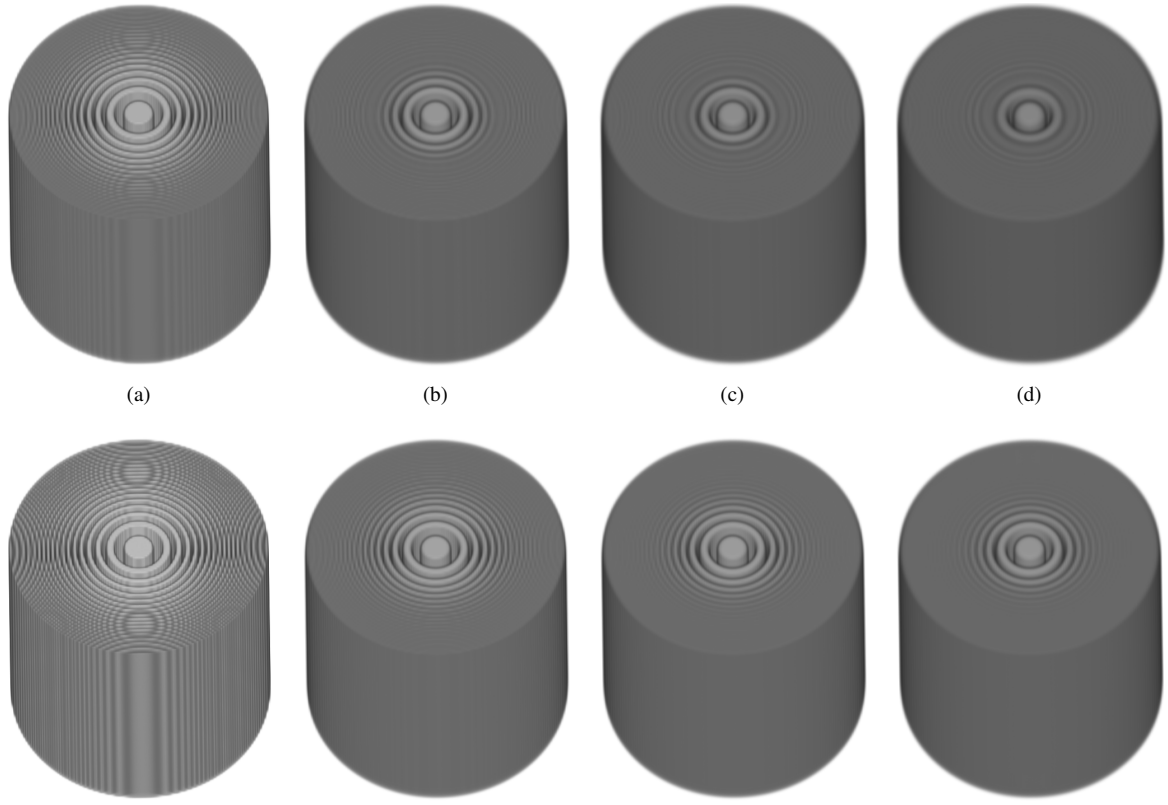
Finally, Figure 9 shows a sea urchin sampled at different spatial resolutions. The direct volume rendering shows how our analytic filtering smoothly filters the solid volume at different spatial resolutions.

6. Summary and Future Work

This paper presented an analytic formulation for anti-aliased sampling of polytopes with a linear function defined on them. This includes the practically highly relevant cases of triangles and tetrahedra with linearly interpolated vertex values. Together with a spherically symmetric polynomial filter, this allows the analytical evaluation of the convolution integrals. Using the closed form solution formulas we described an efficient GPU implementation in the 2D and 3D case.

The linear volumetric approach might be used to render time-varying data, like for example the helicopter blades in Dachille and Kaufman [DK00]. Elliptical filtering is made possible straightforwardly by scaling the polyhedra in the opposite direction to render effects like motion blur and support non-square pixels.

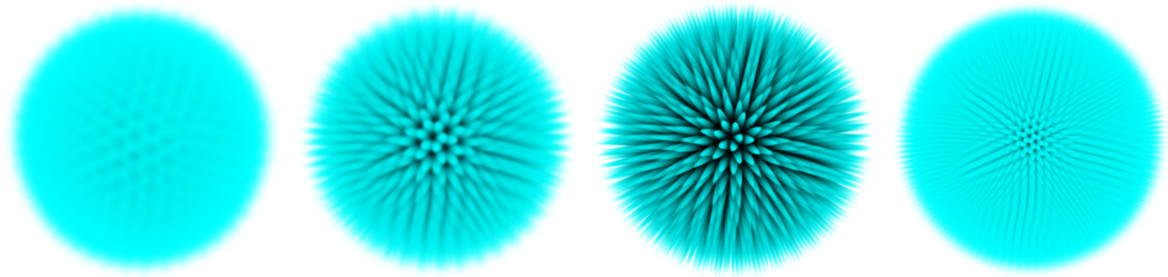
In the future we plan to extend our approach to also sup-



(a) (b) (c) (d)

(e) (f) (g) (h)

Figure 8: Direct volume renderings of an extruded zone plate pattern. The input data consists of nearly two million tetrahedra and was analytically filtered with an area filter (a)-(d) and a Gaussian filter (e)-(h) to a Cartesian grid with dimensions 128^3. Each filter was evaluated with support radii 1, 2, 2.5 and 3 (left to right). For too narrow radii both filters show severe aliasing (a)&(e). While the most glaring artifacts disappear with increasing radii, the area filter still exhibits defects at low frequencies which are not present when using the Gaussian filter. The rendering technique uses linear interpolation of the source data and a linear transfer function.



(a) 64^3 (b) 128^3 (c) 256^3 (d) 256^3

Figure 9: Direct volume renderings of a colored sea urchin with different complexities at various spatial resolutions (given as captions) using a Gaussian filter with 2.3 voxels filter radius. The tetrahedron count for (d) is 12652 and 2470 for (a)-(c). The base of the spikes is colored black while the tip vertex is in cyan. The color value is linearly interpolated inside the spike tetrahedra. Note that for decreasing spatial resolution (a)-(c) or higher frequencies in the source data (d) the output shows no aliasing effect.

	64^3			128^3			256^3		
tets	$\cap$	$\int$	$\sum$	$\cap$	$\int$	$\sum$	$\cap$	$\int$	$\sum$
19.0k	2.0	0.5	2.5	11.7	2.0	13.8	78.6	12.0	90.5
190.0k	4.2	1.1	5.2	18.9	4.1	22.9	111.7	19.0	130.6
1.9M	12.1	3.1	15.3	38.1	9.8	47.8	179.9	37.0	216.8

Table 1: Time taken (in seconds) to sample a cube – composed of a varying number of tetrahedra (tets) – to Cartesian grids of increasing complexity. The timings for the intersection stage ($\cap$), the integration stage ($\int$), and their sum ($\sum$) is given. A Gaussian filter function with five even polynomial orders and a radius of 2 grid cell lengths was used.

port temporal filtering [GBAM11]. In addition we will investigate wavelet approaches for spherical filters [MS11]. Finally, we plan to incorporate analytic visibility methods and extend our filtering to perspective interpolation. This way the method could be used for general analytic 3D rendering.

7. Acknowledgments

We want to thank the reviewers for their insightful and helpful remarks, Hang Si for making available *TetGen* and Stefan Bruckner for *VolumeShop*. Funding was provided by the FWF grant P20768-N13.

References

[Cat78] CATMULL E.: A hidden-surface algorithm with anti-aliasing. In *Proceedings of the 5th annual conference on Computer graphics and interactive techniques* (New York, NY, USA, 1978), SIGGRAPH '78, ACM, pp. 6–11. 2

[Cat84] CATMULL E.: An analytic visible surface algorithm for independent pixel processing. In *Proceedings of the 11th annual conference on Computer graphics and interactive techniques* (New York, NY, USA, 1984), SIGGRAPH '84, ACM, pp. 109–115. 2

[DK00] DACHILLE F., KAUFMAN A. E.: Incremental triangle voxelization. In *Graphics Interface* (2000), pp. 205–212. 6

[Duf89] DUFF T.: Polygon scan conversion by exact convolution. *Raster Imaging and Digital Typography '89* (1989), 154–168. 2, 3

[FLC80] FEIBUSH E. A., LEVOY M., COOK R. L.: Synthetic texturing using digital filters. In *Proceedings of the 7th annual conference on Computer graphics and interactive techniques* (New York, NY, USA, 1980), SIGGRAPH '80, ACM, pp. 294–301. 2

[GBAM11] GRIBEL C. J., BARRINGER R., AKENINE-MÖLLER T.: High-quality spatio-temporal rendering using semi-analytical visibility. In *ACM SIGGRAPH 2011 papers* (New York, NY, USA, 2011), SIGGRAPH '11, ACM, pp. 54:1–54:12. 8

[Gra85] GRANT C. W.: Integrated analytic spatial and temporal anti-aliasing for polyhedra in 4-space. In *Proceedings of the 12th annual conference on Computer graphics and interactive techniques* (New York, NY, USA, 1985), SIGGRAPH '85, ACM, pp. 79–84. 2

[GT96] GUENTER B., TUMBLIN J.: Quadrature prefiltering for high quality antialiasing. *ACM Trans. Graph. 15* (October 1996), 332–353. 2, 4

[HYFK98] HUANG J., YAGEL R., FILIPPOV V., KURZION Y.: An accurate method for voxelizing polygon meshes. In *Proceedings of the 1998 IEEE symposium on Volume visualization* (New York, NY, USA, 1998), VVS '98, ACM, pp. 119–126. 3

[Jam95] JAMES J.: *A student's guide to Fourier transforms: with applications in physics and engineering*. Cambridge University Press, New York, 1995. 2

[Jon96] JONES M. W.: The production of volume data from triangular meshes using voxelisation. *Computer Graphics Forum 15* (1996), 311–318. 3

[Kau87] KAUFMAN A.: Efficient algorithms for 3d scan-conversion of parametric curves, surfaces, and volumes. In *Proceedings of the 14th annual conference on Computer graphics and interactive techniques* (New York, NY, USA, 1987), SIGGRAPH '87, ACM, pp. 171–179. 3

[KU81] KAJIYA J., ULLNER M.: Filtering high quality text for display on raster scan devices. In *ACM Trans. Graph.* (1981), SIGGRAPH '81, ACM, pp. 7–15. 2

[LCSW05] LIN Z., CHEN H.-T., SHUM H.-Y., WANG J.: Pre-filtering two-dimensional polygons without clipping. *journal of graphics, gpu, and game tools 10*, 1 (2005), 17–26. 2

[McC95] MCCOOL M. D.: Analytic antialiasing with prism splines. In *ACM Trans. Graph.* (1995), SIGGRAPH '95, ACM, pp. 429–436. 2

[MN88] MITCHELL D. P., NETRAVALI A. N.: Reconstruction filters in computer-graphics. In *ACM Trans. Graph.* (1988), SIGGRAPH '88, ACM, pp. 221–228. 2

[MS11] MANSON J., SCHAEFER S.: Wavelet rasterization. *Computer Graphics Forum 30*, 2 (2011), 1–10. 3, 8

[SK99] SRAMEK M., KAUFMAN A. E.: Alias-free voxelization of geometric objects. *IEEE Transactions on Visualization and Computer Graphics 5* (July 1999), 251–267. 3

[ST85] SCHREIBER W. F., TROXEL D. E.: Transformation between continuous and discrete representations of images: A perceptual approach. *IEEE Transactions on Pattern Analysis and Machine Intelligence 7* (March 1985), 178–186. 1

[vK98] ŠRÁMEK M., KAUFMAN A.: Object voxeliztion by filtering. In *Proceedings of the 1998 IEEE symposium on Volume visualization* (New York, NY, USA, 1998), VVS '98, ACM, pp. 111–118. 3

[WK93] WANG S. W., KAUFMAN A. E.: Volume sampled voxelization of geometric primitives. In *Proceedings of the 4th conference on Visualization '93* (Washington, DC, USA, 1993), VIS '93, IEEE Computer Society, pp. 78–84. 3

Appendix A: Closed form solutions

The following integration results are given for a fixed filter order n. The final value is obtained by summation of the desired filter orders with their respective weights. It should be noted that although the results contain non-elementary functions such as the Gauss hypergeometric function ${}_2F_1$ and the Appell hypergeometric function F_1, the terms for a fixed order n can be expressed with elementary functions. We use the abbreviation $f(x)|_{x_0}^{x_1}$ for $f(x_1)-f(x_0)$.

The formulas were obtained with the help of a symbolic mathematical software and manually corrected, checked and optimized. The variable names used in the formulas are consistent with figures 4 and 5. Further explanations can be found in the additional material.

2D Integrals

Assuming that $0 < r_0 < R$, $n \in \mathbb{N}$ and interpolation coefficients $\gamma, \lambda, \mu \in \mathbb{R}$ we get with polar coordinates $x = r\cos\varphi, y = r\sin\varphi$

$$\int_0^R r \int_{\varphi_0}^{\varphi_1} (\gamma + \lambda x + \mu y) r^n \, d\varphi \, dr = R^{n+2} \left(\frac{\gamma}{n+2} (\varphi_1 - \varphi_0) + \frac{R}{n+3} (\lambda (\sin\varphi_1 - \sin\varphi_0) - \mu (\cos\varphi_1 - \cos\varphi_0)) \right), \tag{8}$$

and with the further assumption that $-\frac{\pi}{2} < \varphi_0, \varphi_1 < \frac{\pi}{2}$ we obtain

$$\int_{\varphi_0}^{\varphi_1} \int_0^{\frac{r_0}{\cos\varphi}} r(\gamma + \lambda x + \mu y) r^n \, dr \, d\varphi = r_0^{n+2} \left(\left(\frac{\gamma}{n+2} + \frac{r_0 \lambda}{n+3} \right) {}_2F_1 \left(\frac{1}{2}, \frac{n+3}{2}; \frac{3}{2}; \sin^2\varphi \right) \sin\varphi + \frac{r_0 \mu}{(n+2)(n+3)} \sec^{n+2}\varphi \right) \Bigg|_{\varphi_0}^{\varphi_1}. \tag{9}$$

3D Integrals

Assuming that $0 < d < R$, $n \in \mathbb{N}$, $0 < \theta < \frac{\pi}{2}$ and $\gamma, \lambda, \mu, \tau \in \mathbb{R}$ with spherical coordinates $x = r\cos\varphi\sin\theta, y = r\sin\varphi\sin\theta, z = r\cos\theta$ we get

$$\begin{aligned} \int_{\varphi_0}^{\varphi_1} \int_0^{\arccos\left(\frac{d}{R}\right)} \sin\theta \int_0^{\frac{d}{\cos\theta}} r^2 (\gamma + \lambda x + \mu y + \tau z) r^n \, dr \, d\theta \, d\varphi = \frac{d}{n+4} \Bigg(& \frac{(n+4)\gamma + (n+3) d\tau}{(n+2)(n+3)} \left(R^{n+2} - d^{n+2} \right) (\varphi_1 - \varphi_0) \\ & + 8 d^{n+3} \sqrt{1 - \frac{2d}{d+R}} \left(F_1 \left(\frac{1}{2}; n+2, -n; \frac{3}{2}; 1 - \frac{2d}{R+d}, \frac{2d}{R+d} - 1 \right) - 3F_1 \left(\frac{1}{2}; n+3, -n; \frac{3}{2}; 1 - \frac{2d}{R+d}, \frac{2d}{R+d} - 1 \right) \right. \\ & \left. + 2F_1 \left(\frac{1}{2}; n+4, -n; \frac{3}{2}; 1 - \frac{2d}{R+d}, \frac{2d}{R+d} - 1 \right) \right) (\lambda (\sin\varphi_1 - \sin\varphi_0) - \mu (\cos\varphi_1 - \cos\varphi_0)) \Bigg) \end{aligned} \tag{10}$$

whereas with the additional conditions $-\frac{\pi}{2} < \varphi_0, \varphi_1 < \frac{\pi}{2}$, $0 < r_0 < R$ and $0 \geq \theta_C = \arccos\left(\frac{d}{R}\right) < \frac{\pi}{2}$ we obtain

$$\begin{aligned} \int_{\varphi_0}^{\varphi_1} \int_{\theta_C}^{\arctan\left(\frac{r_0}{d\cos\varphi}\right)} \sin\theta \int_0^R r^2 (\gamma + \lambda x + \mu y + \tau z) r^n \, dr \, d\theta \, d\varphi = \frac{R^{n+3}}{4} \Bigg(& -\frac{4\gamma}{n+3} \arctan \left(\frac{\sqrt{2} d \sin\varphi}{\sqrt{2r_0^2 + d^2 + d^2\cos 2\varphi}} \right) \\ & + \frac{2R(r_0\tau - d\lambda)}{(n+4)\sqrt{r_0^2 + d^2}} \arctan \left(\frac{r_0 \tan\varphi}{\sqrt{r_0^2 + d^2}} \right) + \frac{2R}{n+4} \arctan\left(\frac{r_0 \sec\varphi}{d} \right) (\lambda \sin\varphi - \mu\cos\varphi) + \frac{4\gamma\cos\theta_C}{n+3}\varphi \\ & + \frac{R}{n+4} (\tau (\cos 2\theta_C - 1)\varphi + 2(\theta_C - \cos\theta_C \sin\theta_C)(\mu\cos\varphi - \lambda\sin\varphi)) \Bigg) \Bigg|_{\varphi_0}^{\varphi_1}. \end{aligned} \tag{11}$$

The evaluation of the integrand on a tetrahedral integration domain proves much harder than the previous cases. This can be attributed to the fact that while the integration domain can be very easily expressed in Cartesian coordinates, the filter monomial r^n is easily described in spherical coordinates. Thus we have not found a result formula that permits the filter order n to be expressed as a variable. Nevertheless it is possible to evaluate the integral for each filter order separately. We show the results for the first few even orders as the odd orders produce lengthy expressions.

$$\int_{\varphi_0}^{\varphi_1}\int_0^{\arctan\left(\frac{r_0}{d\cos\varphi}\right)}\sin\theta\int_0^{\frac{d}{\cos\theta}} r^2(\gamma+\lambda x+\mu y+\tau z)r^n\,dr\,d\theta\,d\varphi =$$

$$\begin{cases}
\frac{dr_0^2}{24}\left((4\gamma+2r_0\lambda+3d\tau)\tan\varphi+r_0\sec^2\phi\mu\right)\Big|_{\varphi_0}^{\varphi_1} & n=0\\
\frac{dr_0^2}{360}\Big(\Big(12\left(3d^2+r_0^2\right)\gamma\ +r_0\left(20d^2+8r_0^2\right)\lambda+10d\left(3d^2+r_0^2\right)\tau\Big)\tan\varphi & \\
\qquad +r_0\sec^2\varphi\left(10d^2\mu+r_0\left(6\gamma+4r_0\lambda+5d\tau\right)\tan\varphi\right)+3r_0^3\sec^4\varphi\mu\Big)\Big|_{\varphi_0}^{\varphi_1} & n=2\\
\frac{dr_0^2}{5040}\Big(\Big(8\left(45d^4+30d^2r_0^2+8r_0^4\right)\gamma+6r_0\left(35d^4+28d^2r_0^2+8r_0^4\right)\lambda+7d\left(45d^4+30d^2r_0^2+8r_0^4\right)\tau\Big)\tan\varphi & \\
\qquad +r_0\sec^2\varphi\Big(105d^4\mu+r_0\Big(8\left(15d^2+4r_0^2\right)\gamma+r_0\left(84d^2+24r_0^2\right)\lambda+7d\left(15d^2+4r_0^2\right)\tau\Big)\tan\varphi\Big) & n=4\\
\qquad +3r_0^3\sec^4\varphi\left(21d^2\mu+r_0\left(8\gamma 6r_0\lambda+7d\tau\right)\tan\varphi\right)+15r_0^5\sec^6\varphi\mu\Big)\Big|_{\varphi_0}^{\varphi_1} &
\end{cases} \tag{12}$$

Appendix B: Polynomial filter coefficients

Table 2 and 3 show the coefficients for polynomial fits of various common anti-aliasing filters together with the approximation error. All filters were normalized to the interval $[0,1]$. The numbers refer to the original filter width, i.e. the Gaussian ($\sigma = 2^{-1/2}$) is cut off at 2.3 and the Lanczos filter is of radius 2. Note that for 3D filters we use only coefficients for *even* polynomials while the 2D filters can also use odd and thus have a lower degree. The coefficients are computed using a least squares fit with the pre-integrated polynomials shown in table 4. The pre-integrations ensure that our filters satisfy equation (2).

filter (radius)	c_0	c_1	c_2	c_3	c_4	ε_{max}
Gaussian (2.3)	1.30321	0.119155	-9.69022	14.2894	-6.02528	0.00579162
Lanczos (2)	1.68971	0.71905	-19.8035	31.601	-14.2415	0.00892035
Mitchell-Netravali	2.03111	-2.16092	-15.102	31.24	-16.13328	0.0347118
B	-1.16265	2.328	8.46852	-21.7564	12.2412	0.0332397
C	-0.521127	3.9474	-4.20988	-4.72515	5.66	0.0445042
Blackman-Harris	1.36055	0.33004	-11.613	17.2908	-7.386368	0.00457847

Table 2: Coefficients for polynomials fits of various 2D filters. All filters are normalized to the range $[0,1]$.

filter (radius)	c_0	c_2	c_4	c_6	c_8	ε_{max}
Gaussian (2.3)	1.65951	-8.09529	16.6202	-16.1818	6.05909	0.00504176
Lanczos (2)	2.58949	-14.7969	30.5864	-27.9904	9.64687	0.00298654
Mitchell-Netravali	3.25801	-24.7342	65.8731	-73.6141	29.366	0.0131654
B	-2.16456	19.7691	-56.6125	65.4189	-26.5495	0.013814
C	-1.1118	16.1821	-56.1939	71.2608	-30.3158	0.0201162
Blackman-Harris	1.79798	-8.86188	17.5986	-16.2609	5.75158	0.00211148

Table 3: Coefficients for polynomials fits of various 3D filters. All filters are normalized to the range $[0,1]$.

monomial (2D / 3D)	2D pre-integration	3D pre-integration
1	$2\sqrt{1-x^2}$	$\left(1-x^2\right)\pi$
x / x^2	$\sqrt{1-x^2}+x^2\log\left(\frac{1+\sqrt{1-x^2}}{x}\right)$	$\left(1-x^4\right)\frac{\pi}{2}$
x^2 / x^4	$\frac{2}{3}\sqrt{1-x^2}\left(1+2x^2\right)$	$\left(1-x^6\right)\frac{\pi}{3}$
x^3 / x^6	$\frac{1}{4}\left(\sqrt{1-x^2}\left(2+3x^2\right)+3x^4\log\left(\frac{1+\sqrt{1-x^2}}{x}\right)\right)$	$\left(1-x^8\right)\frac{\pi}{4}$
x^4 / x^8	$\frac{2}{5}\sqrt{1-x^2}\left(3+4x^2+8x^4\right)$	$\left(1-x^{10}\right)\frac{\pi}{5}$

Table 4: Pre-integrated polynomials for the range $[0,1]$.