Sheared Polygonal Texture Filtering

Guowei Lu* Delft University of Technology
Jerry Jinfeng Guo† Delft University of Technology
Petr Kellnhofer‡ Delft University of Technology
Elmar Eisemann§ Delft University of Technology

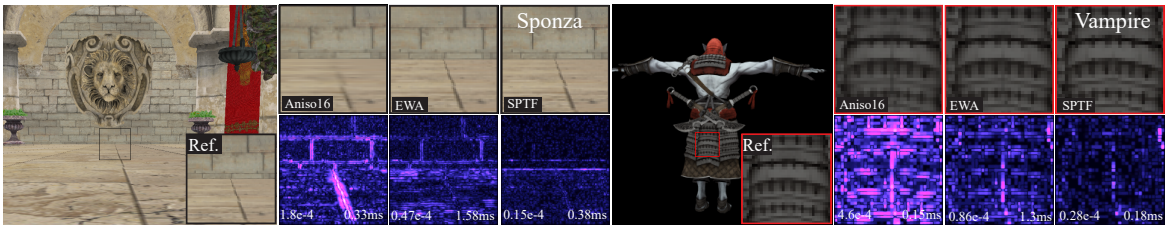


Figure 1: Our Sheared Polygonal Texture Filtering (SPTF) demonstrates superior filtering accuracy compared to hardware-accelerated Anisotropic filtering (ANISO16) at a lower computational cost in comparison to Elliptical Weighted Average (EWA). (Left) In the first scene, the floor exhibits enhanced sharpness and intricate details at grazing viewing angles when rendered using our method. (Right) Additionally, in the second scene, our approach successfully eliminates seams associated with atlas texturing. Notice the barely discernible differences between the SPTF and the EWA outputs, which are more visible in the FLIP error maps [2].

Abstract

Efficient and precise texture filtering is essential in various applications. However, there is often a trade-off between coarse real-time approximations and accurate computationally-expensive supersampling. We introduce a novel efficient texture-filtering method over arbitrary quadrilateral footprints, achieving high accuracy at a low computational cost. We achieve this by pre-computing integration tables that sparsely sample the space of possible footprints. Finally, we compare the qualitative and computational performance of our method to commonly used techniques and demonstrate various applications for high-quality real-time image synthesis, including normal filtering, soft shadow mapping, and glint rendering.

Index Terms: Image sampling, reconstruction, and filtering techniques—Rendering algorithms

1 Introduction

Texture filtering is important for high-quality image generation in computer graphics and common in real-time applications [12]. It can prevent visual artifacts, such as aliasing and Moiré patterns associated with the under-sampling of a high-frequency signal. However, texture filtering implies costly integrations over large and deformed pixel footprints in texture space.

The commonly used *Mipmapping* [25] technique enables real-time performance through the precomputation of multi-scale filter outputs. However, even in its *Anisotropic* variant [16], it remains susceptible to aliasing under extreme viewing angles (Fig. 1a). More complex filters, such as the *Elliptical Weighted Average* (EWA) [10], improve upon that, but are computationally expensive, typically targeting off-line rendering. Our work aims to bridge these two directions and provide state-of-the-art accuracy with real-time performance.

A 2D *Summed Area Table* (SAT) [4] enables fast texture integration over an axis-aligned rectangle. We target general quadrilaterals using multiple tables. One related challenge is numerical errors that could aggregate, which our approach *Sheared Polygonal Texture Filtering* (SPTF) addresses via the design of both our SATs as well as filter footprints. In summary, our key contributions are:

- An algorithm to closely approximate integration over quadrilateral texture regions;
- A study of the filter-shape impact on accuracy and rendering cost;
- Experimental validation of our solution against other methods;
- Various application examples (soft shadow-mapping, normal filtering, and glint rendering).

In the rest of the paper, we first discuss the related prior work (Sec. 2). Next, we explain our method (Sec. 3) and evaluate its performance in several computer graphics tasks (Sec. 4). Finally, we discuss our results and limitations (Sec. 5).

2 Related Work

A rectangular screen pixel projected into the texture space of a locally planar surface follows a homography leading to an arbitrary quadrilateral [14]. Texture filtering integrates over this region, which is computationally challenging in real-time [1]. There are many solutions trading off accuracy for performance, which we cannot cover all. Instead, we refer to two surveys [12, 28].

Given its ubiquitous hardware support, a technique of choice for high-quality real-time rendering is anisotropic filtering [8]. It combines multiple samples from pre-computed MIP maps to approximate the intended filter kernel. However, the fixed number of samples limits the kernel's aspect ratio, leading to excessive blurring or aliasing artifacts in regions with ratios greater than 16. In contrast, our method does not introduce any aspect-ratio limits.

A high-quality approximation relies on EWA [10], assuming circular pixels and a weighted combination of texture samples, corresponding to a pre-integration by elliptical kernels of varying sizes and orientations. Yet, large aspect ratios require many samples for a precise approximation, which makes the method costly. Our method has a bound on the number of texture lookups, irrespective of kernel size, and avoids the circular approximation. For real-time applications, EWA samples can be approximated using hardware-accelerated anisotropic filtering [17]. However, the quality depends on the specific hardware implementation. Our method is software-based and produces device-independent outputs.

*e-mail: G.Lu-1@tudelft.nl
†e-mail: J.Guo-3@tudelft.nl
‡e-mail: P.Kellnhofer@tudelft.nl
§e-mail: E.Eisemann@tudelft.nl

SATs allow the effective integration for axis-aligned rectangular kernels [15], which has limited utility for texture filtering using a projected pixel footprint. Novosad et al. [18] splits the quadrilateral filter kernel into rectangular segments, which is typically inaccurate.

In this paper, we integrate directly over the area of the projected quadrilaterals without simplifying the region or limiting the kernel aspect ratio. Our objective is to produce results that closely match the reference while maintaining efficiency.

3 Our Approach

We extend the concept of SAT to *Sheared Summed Area Tables* (SSAT) for a fast close-to-accurate integration over arbitrary polygonal pixel footprints. In the following, we formalize the filtering (Sec. 3.1), then outline our solution (Sec. 3.2) and introduce our precomputation step (Sec. 3.3). Finally, we apply our solution to texture filtering in two variants (Sec. 3.4).

3.1 Integral over a quadrilateral

We aim at integrating over a pixel's footprint projected to texture space. Its projection forms a general quadrilateral resulting from the homography transformation of square pixels assuming a locally planar scene surface.

For the quadrilateral Q representing a projected pixel (Fig. 2(a)), we obtain:

$$I = \iint_Q \mathscr{P}(u,v)\mathrm{d}u\mathrm{d}v, \tag{1}$$

where $\mathscr{P}$ is the texture value at (u,v) and I is the area integral of $\mathscr{P}$ over the region Q.

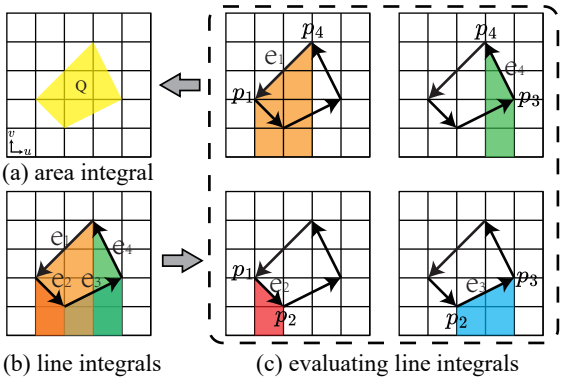


Figure 2: Integral over quadrilateral. We transform the area integral over a quadrilateral (a) into line integrals over its boundary (b). Then, we evaluate these 4 line integrals numerically (c). This method can be extended to arbitrary polygons.

As shown in Fig. 2(b), we can divide the boundary C of the quadrilateral $Q = \{\mathbf{p}_i\}_{i=1}^4$ into four directed line segments $\{e_i = (\mathbf{p}_a, \mathbf{p}_b)\}_{i=1}^4$ where $\mathbf{p}_a$ and $\mathbf{p}_b$ are neighboring vertices enumerated in a consistent counter-clockwise order. Finally, we refer to a line slope of e_i as λ_i.

Next, Green's theorem [23, p. 989, Eq. 6] relates a line integral around the boundary C to an area integral over the region Q bounded by C. By defining a vertical line integral $M(u,v) = \int_0^v \mathscr{P}(u,v')\mathrm{d}v'$, where (u,v) is a point on the boundary C, we obtain:

$$I = \iint_Q \underbrace{\mathscr{P}(u,v)}_{\frac{\partial M}{\partial v}} \mathrm{d}u\mathrm{d}v = -\oint_C M(u,v)\mathrm{d}u$$
$$= -\sum_{i=1}^{4} \int_{e_i} M(u,v)\,\mathrm{d}u = -\sum_{i=1}^{4} \mathscr{L}(e_i), \tag{2}$$

where $\mathscr{L}(e_i)$ represents an area integral of all values under edge e_i. Note that the boundary C is divided into four directed line segments and the sign of $\mathscr{L}(e_i)$ depends on the edge orientation.

The computational cost of the integral in Eq. 1 is proportional to the number of texels inside the quadrilateral. Yet, if the cost of $\mathscr{L}(e_i)$ was constant, it would follow that the cost of the sum in Eq. 2 is constant for a quadrilateral. In the following section, we will see how to precompute these terms to approximate the integral.

3.2 Sheared Summed Area Table

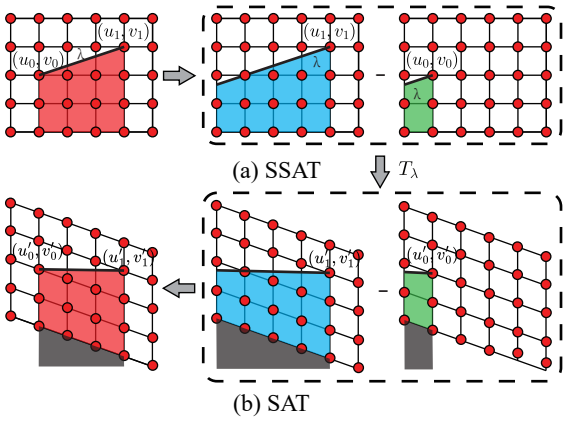


Figure 3: Our SSAT represents vertical integration under a line segment with a slope λ (a). We use shear transformation, to convert this problem to integration under a horizontal line segment which can be solved using SAT (b).

In a classical summed area table, $SAT(\mathbf{p})$ represents the integral over a rectangular area between the origin and point $\mathbf{p} = (u,v)$. Our SSAT generalizes this to a sheared coordinate system described by a shear slope λ (Fig. 3(a) in blue). Instead of integrating along the slope directly, we linearly transform the texture using a shear operator T_λ such that the slope becomes zero and the SSAT becomes equivalent to a classical SAT (Fig. 3(b)):

$$SSAT(\lambda,\mathbf{p}) = \iint_{Z(\lambda,\mathbf{p})} \mathscr{P}(u,v)\,\mathrm{d}u\,\mathrm{d}v$$
$$= \iint_{R(u'_p,v'_p)} \mathscr{P}(u',v') \underbrace{\left|T_\lambda^{-1}(u',v')\right|}_{1} \mathrm{d}u'\mathrm{d}v'$$
$$= SAT(\mathbf{p}'), \tag{3}$$

where (u,v) and (u',v') are the coordinates in the original and sheared texture spaces respectively, and T_λ is the corresponding transformation $\mathbf{p}' = T_\lambda(\mathbf{p})$. Z denotes the area of a trapezoid under a line with a slope λ between the origin and $\mathbf{p}$ (see Fig. 3(a)), while R denotes an area of a rectangle corresponding to Z transformed to the sheared space.

While this formulation does not avoid interpolation errors in the texture transformation, it ensures that these errors are consistent for

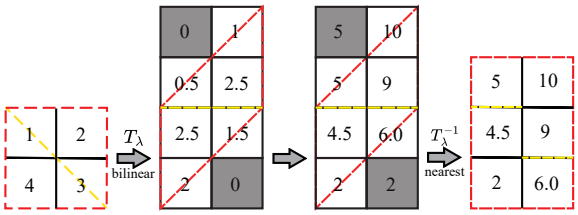
all SSAT values computed for the same λ and, hence, they cancel out when computing segment integrals.

3.3 Lookup table construction

We now present a three-step algorithm for producing a lookup table for our SSAT (see Fig. 4): a forward shear transformation T_λ, an SAT, and an inverse transformation T_λ^{-1}.

We use a linear filter when resampling the original texture following the shear operator T_λ in Eq. 3. An output texture is vertically expanded to accommodate the transformed texels. After computing a standard SAT, the output contains many empty texels (shown in gray in Fig. 4(b,c)). To reduce the memory footprint, we do one additional remapping from the (u',v') sheared coordinates to the (u'',v'') coordinates used for SSAT readout. This is done using the inverse shear operator T_λ^{-1} with nearest sampling to avoid additional interpolation errors (Fig. 4(d)). The composed coordinate transformation $\mathscr{T}$ can then be simplified as:

$$(u'',v'') = \mathscr{T}(u,v,\lambda) = (u, v+\lambda u - \text{floor}(\lambda u)). \tag{4}$$



(a) Original texture (b) Sheared texture (c) SAT (d) Sheared SAT

Figure 4: SSAT construction. (a) Original texture, (b) Vertical shear transformation with a bilinear sampler creates a sheared texture, (c) classical SAT, and (d) Inverse shear operation with a nearest sampler removes unused texels and produces our final SSAT texture.

Lookup table parameterization

A shear transformation with $\lambda \in (-\infty, +\infty)$ leads to an unbounded expansion of the image. We avoid this by rotating the original texture before the SSAT computation such that $\lambda \notin [-1,1]$ corresponds to a $\lambda \in [-1,1]$ for the rotated texture. This reduces the worst case memory overhead to a factor of 2 per table. We define a corresponding transformation operator T_λ for $\lambda \in (-\infty, +\infty)$ as:

$$T_\lambda = \begin{cases} S_\lambda & \lambda \in [-1,1] \\ S_{-1/\lambda} R_{\pi/2} & \lambda \notin [-1,1], \end{cases} \tag{5}$$

where $R_{\pi/2} \in \mathbb{R}^{2\times 2}$ is a rotation and $S_\lambda \in \mathbb{R}^{2\times 2}$ a vertical shear.

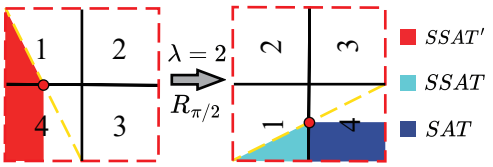


Figure 5: Sampling of SSAT (Eq. 7) for slope $\lambda \in (1,\infty)$ (yellow dashed line). A SAT (blue rectangle) is combined with a rotated SSAT (cyan triangle) lookup to obtain the intended integral (red area). $\lambda \in (-\infty,-1)$ is analogous.

A consistent integration direction is required in Eq. 2, but the rotation $R_{\pi/2}$ in Eq. 5 violates this. Therefore, in the respective cases, we compute a complementary vertical integral with an additional SAT lookup for the bounding rectangle (see Fig. 5) and, hence, the line segment integral $\mathscr{L}(e_i)$ for $e_i = (\mathbf{p}_a, \mathbf{p}_b)$ is:

$$\mathscr{L}(e_i) = SSAT'(\lambda, \mathbf{p}_b) - SSAT'(\lambda, \mathbf{p}_a), \tag{6}$$

where

$$SSAT'(\lambda,\mathbf{p}) = \begin{cases} SAT(\mathbf{p}) - SSAT(\lambda, \mathscr{T}(R_{\frac{\pi}{2}}\mathbf{p}, \frac{-1}{\lambda})) & \lambda \in (-\infty,-1) \\ SSAT(\lambda, \mathscr{T}(\mathbf{p},\lambda)) & \lambda \in [-1,1] \\ SAT(\mathbf{p}) + SSAT(\lambda, \mathscr{T}(R_{\frac{\pi}{2}}\mathbf{p}, \frac{-1}{\lambda}) & \lambda \in (1,\infty). \end{cases} \tag{7}$$

Since $SAT(\mathbf{p}) = SSAT(0, \mathscr{T}(\mathbf{p},0)) = SSAT(0,\mathbf{p})$, this does not yield an additional pre-computation cost.

In combination with Eq. 2, this allows for fast integration over an area of an arbitrary polygon assuming the existence of SSAT tables for all necessary slopes λ_i.

Step size

An enumeration of all endpoint pairs necessary to ensure the availability of all possible slopes λ in a texture with size $n \times n$ yields $O(n^4)$ complexity (Fig. 3(a)). This is intractable even after accounting for duplicate slope values. Instead, we limit the memory footprint of SSAT to $O(kn^2)$ where k is a fixed slope count. We uniformly sample the slope λ with a step $s = 4/k$ to cover range $[0,1]$ for the four possible combinations of a rotation and a mirroring that together fill the entire space of $\lambda \in (-\infty, +\infty)$ and define our slope set Γ. In practice, we find $s = 0.5$ (i.e., 8 tables) sufficient for accuracy comparable to EWA [10]. Please refer to Sec. 4 for a more in-depth analysis.

3.4 Texture filtering

We model screen-space pixels as squares [3] and their projection to locally planar surfaces as a homography [13]. Filtering the resulting general quadrilateral polygons is integrated using our tables derived from the discrete slope sampling. We will present two different approximations - a nearest quadrilateral (Fig. 6(b)) and a semi-axis-aligned parallelogram (semi-parallelogram for short, see Fig. 6(c)).

Given a quadrilateral Q with arbitrary edges $\{e_i\}_{i=1}^4$ each with its slope λ_i, we search for the nearest slopes $\lambda_i' = \arg\min_\lambda |\lambda_i - \lambda|$ subject to $\lambda_i' \in \Gamma$. A new quadrilateral Q_Q, which can be evaluated using our SSAT, is then constructed by intersecting lines with slopes $\{\lambda_i'\}_{i=1}^4$. We integrate over the area of Q_Q following Eq. 2 and refer to this filtering process as SPTF_Q.

To reduce the number of required SSAT samples further, we can approximate Q with a semi-axis-aligned parallelogram Q_S. To this extent, from Q, we derive a 2×2 covariance matrix $C = TT^T$, where $T = [\partial\mathbf{uv}/\partial x, \partial\mathbf{uv}/\partial y]$ is formed by two screen space gradients of the texturing coordinates $\mathbf{uv}$ along the x and y axes (see Fig. 6(c) left). In practice, we obtain the necessary gradients from the built-in OpenGL functions `dFdx` and `dFdy`. Next, we apply Cholesky decomposition $C = LL^T$ to obtain $L \in \mathbb{R}^{2\times 2}$ with one axis-aligned vector v_L and one general vector u_L that jointly define the semi-parallelogram Q_S (see Fig. 6(c) right). Finally, we refer to integration over the area of Q_S as SPTF_S.

Since SPTF_Q allows for up to 4 different slopes, integration over each of the edges requires 2 or 4 unique texture samples depending on λ (see Eq. 6). In contrast, SPTF_S avoids integration over the vertical edges since they cancel out in the final sum. Also, a combination of the two opposing diagonal edges avoids additional lookups when $\lambda \notin [-1,1]$. Therefore, SPTF_S always requires only 4 unique SSAT samples similar to standard SATs for a rectangular area. Refer to Sec. 4 for analysis of the impact on accuracy.

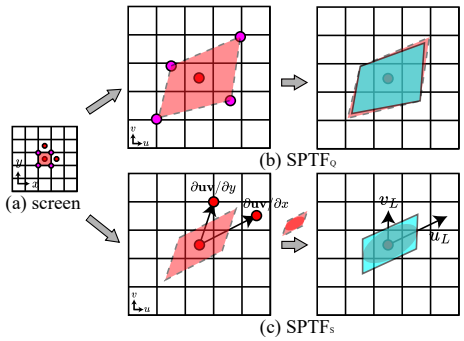


Figure 6: Pixel footprint approximation for SPTF. (a) A square screen-space pixel is projected to a quadrilateral Q in the texture space of the surface. (b) SPTF$_Q$: A new quadrilateral Q_Q is constructed with edges following the nearest slopes λ' within our SSAT sampling space γ. (c) SPTF$_S$: A semi-parallelogram Q_S is constructed by factorization of a covariance matrix.

4 Evaluation

In this section, we compare our algorithm to various texture filtering techniques and assess the accuracy and computational efficiency for game-like assets and a checkerboard (Sec. 4.1). Next, we demonstrate its versatility in various computer graphics applications (Sec. 4.2) before discussing its properties and limitations (Sec. 4.3).

4.1 Rendering Results

Reference. We obtain reference images via the PBRT ray tracer [19] configured to render pure albedo. Each pixel is sampled 2048 times for anti-aliasing. All experiments are conducted on an NVIDIA GeForce RTX 3090 GPU.

Baselines We compare to hardware anisotropic filtering with 16 samples (ANISO16) and a high-quality offline implementation of Elliptical Weighted Average filtering (EWA) with a one-pixel width. We use the visual difference predictor *FLIP* [2] as well as Mean Squared Error (MSE) to gauge resulting differences.

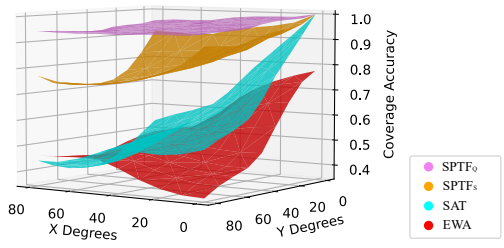


Figure 7: Filter footprint accuracy is measured as a mean Intersection Over Union between the filter kernels and the true pixel footprints Q. Vertical and horizontal viewing angles relative to the rendered surface are shown on horizontal axes with $[0,0]$ representing orthogonal viewpoint. Both SPTF$_Q$ and SPTF$_S$ achieve consistently higher accuracy than the nearest rectangle filter implemented by a standard SAT and filtering by ellipse kernels in EWA. The accuracy of EWA is notably poor even for the orthogonal viewing angle due to the shape mismatch between a square pixel and an ellipse. A step size of $s = 0.05$ was used for our methods.

Texturing In Table 1, we provide a quantitative assessment of our method's variants and the ANISO16 and EWA baselines for three scenes; *Checkerboard* - a classical challenging pattern with high frequency signal, *Sponza* - a real-time asset showcasing the capabilities of anisotropic filtering, *Vampire* - a model utilizing a

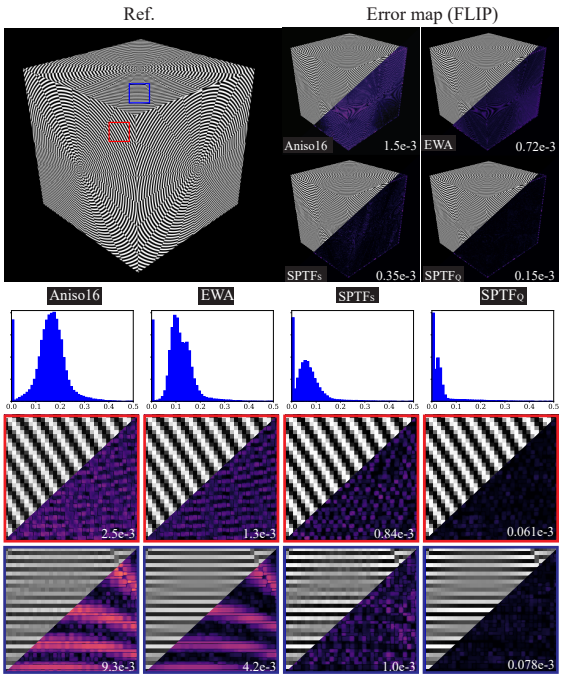


Figure 8: *Checkerboard* analysis. **Top:** FLIP error maps [2] show that both our method variants produce clearly lower errors than ANISO16, and EWAwhen compared to the ground truth reference (left top) in this texture minification task. **Middle:** Histograms of the full FLIP maps weighted by the error magnitudes as proposed by the FLIP authors [2]. **Bottom:** Two zoomed-in regions and their FLIP maps.

texture atlas. See Fig. 8 and Fig. 1 qualitative comparisons. We use a smaller step size $s = 0.1$ for our higher-accuracy method variant SPTF$_Q$ and a higher step size $s = 0.5$ for our lower-weight method variant SPTF$_S$ to further emphasize their distinct strengths.

Our lighter method SPTF$_S$ achieves a consistently lower MSE than both the real-time ANISO16 and the costly EWA, while maintaining at least 2/3 of the best case framerate in each scene. ANISO16 tends to favor over-blurring to avoid aliasing. Our more accurate method SPTF$_Q$ then provides further improvement of accuracy measured by MSE, while maintaining computational cost comparable to EWA.

The differences are smaller for the *Vampire* scene, where grazing surface angles testing the anisotropic performance are not as prominent. Finally, we further analyze the residual filtering artifacts for the *Checkerboard* scene in Fig. 8.

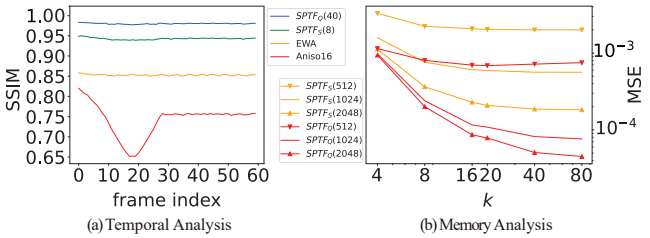
Footprint Analysis The accuracy depends on the match between the approximated kernel and the intended integration area. We illustrate the favorable characteristic of both our kernel approximations in Fig. 7. An accuracy of one means a perfect reproduction of the intended pixel footprints as measured for a realistic pixel grid projected to a planar surface under different viewing angles. The graph shows that our two variants maintain a high footprint accuracy even for extreme viewing angles and beat both analyzed baselines.

Temporal Analysis Inspired by [20], we evaluate the temporal stability by measuring the SSIM reference metric variation during smooth camera motion (Fig. 9(a)). We observe that the temporal variation is lower for our SPTF and the costly EWA than for the commonly used ANISO16 filtering. We further demonstrate this stability qualitatively in the supplemental video.

		Ref.	ANISO16	EWA	SPTF_Q	SPTF_S
	Num. lookups	adaptive	1	adaptive	8-16	4
	Memory (relative)	1	1*	1	40†	8†
Checkerboard	Frame rate (rel.)	-	100%	25%	38%	**66%**
	MSE	0	1.5e-3	0.72e-3	**0.15e-3**	0.35e-3
Sponza	Frame rate (rel.)	-	100%	20%	36%	**90%**
	MSE	0	7.4e-5	3.8e-5	**0.6e-5**	3.2e-5
Vampire	Frame rate (rel.)	-	100%	11%	9%	**81%**
	MSE	0	7.5e-5	6.8e-5	**6.7e-5**	**6.7e-5**

Table 1: A comparison between our method variants and commonly used baselines. *The memory footprint of ANISO16 does not include the implementation-specific MIP tables. †The additional pixel row per table is not included for SPTF (asymptotically negligible for resolution $\to \infty$).

Memory analysis We conducted an analysis of image quality by varying both step size s and texture size (Fig. 9(b)). It shows that a relatively low number of 8 or 16 lookup tables, corresponding to $s = 0.5$ and $s = 0.25$ respectively, is sufficient for high quality results. Additional tables are only beneficial for high resolution textures.



(a) Temporal Analysis (b) Memory Analysis

Figure 9: **(a)** SSIM relative to the reference measured during a camera motion (corresponds to the number of lookup tables, in parentheses). **(b)** Impact of the lookup table resolution (corresponds to the texture resolution, in parentheses) and their count k on MSE. Both are measured in the *Checkerboard* scene.

4.2 Applications

We demonstrate three additional applications: *Normal Mapping*, *Soft Shadows*, and *Real-time Glints*.

Normal Mapping Normal maps are commonplace inputs for algorithms, such as bump mapping, but accurate filtering remains a challenging problem. Due to the non-linearity of surface shading, we cannot pre-filter normal maps directly [11]. Instead, we replace the original anisotropic filter with our SPTF_S and use Toksvig's model [24] to calculate the specular contribution. As illustrated for normal mapping in Fig. 10, our filter preserves fine-grained details better than the baseline.

Soft shadows Eisemann et al. [6, 7] introduced a plausible soft-shadow method based on occlusion texture pre-filtering. They utilize a box filter with a rectangular kernel and propose warping the scene to handle elongated light sources, which lowers the resolution. Our method can instead rely on anisotropic filtering.

With the baseline method, we also observe distinct aliasing for narrow penumbra radii (Fig. 11(a)). To address this, we convolve the original shadow filter kernel with projected kernels of individual image pixels. This results in semi-parallelogram kernels [21,22] suitable for our SPTF_S method. Finally, we obtain smoother and better-quality soft shadows for the grazing view displayed in Fig. 11(b).

Real-time Glints Rendering The distribution of tiny specular particles and their orientation with respect to the object surface can be described by the Normal Distribution Function (NDF). However, high-frequency representations can lead to aliasing [26, 27]. Instead

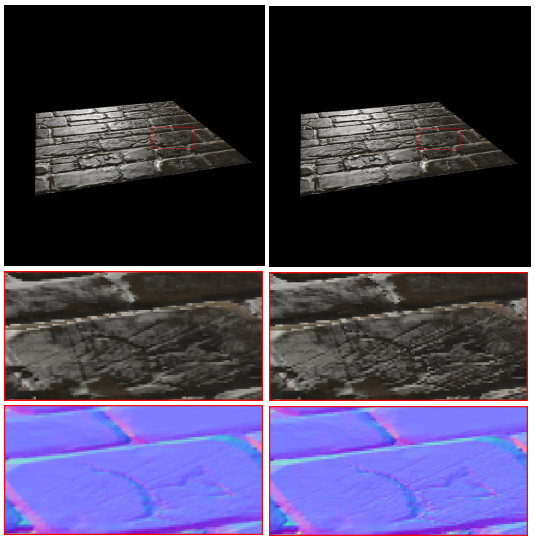
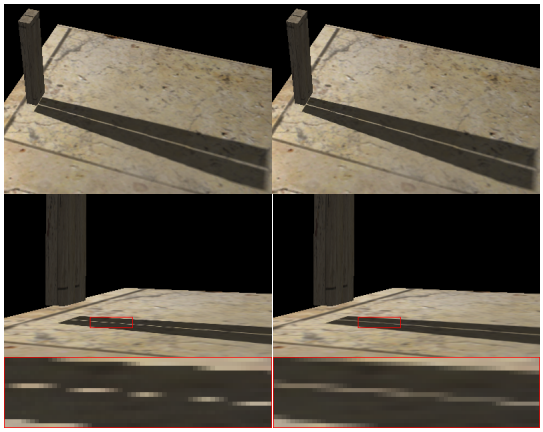

(a) Aniso16 (b) Ours

Figure 10: A bump mapping effect with a normal map integration using ANISO16 and SPTF filters. With ANISO16 normal filtering, these regions are averaged to a nearly flat surface (a). By contrast, our method keeps more details in distant regions (b).

of filtering the NDF [9], we propose to directly encode the spatial distribution of particles as a surface texture containing particle count and their mean normal in each texel. During rendering, we integrate this map over pixel footprints using our SPTF_S before computing the specular contributions. For the normal component, we use the same approach as in the Normal Mapping application. We observe that our simple representation produces visually plausible results and avoids unwanted spatial and temporal aliasing (Fig. 12). Please see our video for a comparison to the method of Deliot et al. [5] which we modified to use the same particle distribution texture. We observe that our method removes aliasing visible in the baseline results during camera motion.

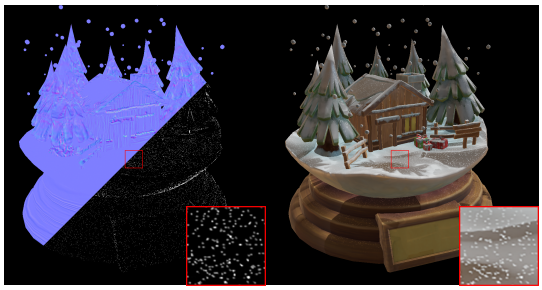
4.3 Limitations

While the results show the accuracy of our method, it remains approximate due to the discretely sampled slope space (Sec. 3.4). Furthermore, the linear interpolation required for the forward shear operation (Fig. 4(b)) does not accurately describe the non-linear contribution of partially included pixels when integrating along non-horizontal slopes. In practice, we observe that this leads to acceptable errors for common textures, although it could have an impact on data with highly unnatural distributions.



(a) Baseline (b) Ours

Figure 11: Soft shadows computed using the baseline method [7] and our SPTF-based modification. **Top:** A scene overview illustrating the narrow slit between the two beams. **Bottom:** Our method reconstructs the lit gap between the two penumbra even at a grazing view angle.



(a) particle normals and counts (b) glint rendering

Figure 12: Our real-time glint rendering. We use a texture to represent per-texel particle counts and mean normals (a). We integrate this texture to obtain specular contributions for the glint effect (b).

Our method supports texture atlases (Fig. 1). However, texture repetition for tiling can only be partially resolved. If the maximum number of wraps per filtering query can be bounded, we can expand the original texture accordingly before SSAT construction. Beyond this limit, we could transition to a standard filtering method.

5 Conclusion

We showed that high accuracy can be obtained at low cost using our SPTF method that builds upon a simple mathematical insight. We proposed an efficient lookup table encoding, making our approach well-suited for real-time rendering. The solution is general and we illustrated its usefulness in various applications. Our work improves standard texture filtering, normal map filtering, soft shadow mapping, and the rendering of glints, which illustrates SPTF's versatility.

Acknowledgments

The work is part of VR-Renovate (project number 403.19.222), financed by the Dutch Research Council (NWO).

References

[1] T. Akenine-Mo, E. Haines, N. Hoffman, et al. Real-time rendering. 2018.

[2] P. Andersson, J. Nilsson, T. Akenine-Möller, M. Oskarsson, K. Åström, and M. D. Fairchild. Flip: A difference evaluator for alternating images. *Proc. ACM Comput. Graph. Interact. Tech.*, 3(2):15–1, 2020.

[3] J. F. Blinn. What is a pixel? *IEEE computer graphics and applications*, 25(5):82–87, 2005.

[4] F. C. Crow. Summed-area tables for texture mapping. In *Proceedings of the 11th annual conference on Computer graphics and interactive techniques*, pp. 207–212, 1984.

[5] T. Deliot and L. Belcour. Real-time rendering of glinty appearances using distributed binomial laws on anisotropic grids. 2023.

[6] E. Eisemann and X. Décoret. Plausible image based soft shadows using occlusion textures. In *Proc. of the Brazilian Symposium on Computer Graphics and Image Processing, 19 (SIBGRAPI)*, 2006.

[7] E. Eisemann and X. Décoret. Occlusion textures for plausible soft shadows. In *Computer Graphics Forum*, vol. 27, pp. 13–23. Wiley Online Library, 2008.

[8] J. P. Ewins, M. D. Waller, M. White, and P. F. Lister. Implementing an anisotropic texture filter. *Computers & Graphics*, 24(2):253–267, 2000.

[9] L. E. Gamboa, J.-P. Guertin, and D. Nowrouzezahrai. Scalable appearance filtering for complex lighting effects. *ACM Trans. Graph.*, 37(6):277–1, 2018.

[10] N. Greene and P. S. Heckbert. Creating raster omnimax images from multiple perspective views using the elliptical weighted average filter. *IEEE Computer Graphics and Applications*, 6(6):21–27, 1986.

[11] C. Han, B. Sun, R. Ramamoorthi, and E. Grinspun. Frequency domain normal map filtering. In *ACM SIGGRAPH 2007 papers*, pp. 28–es. 2007.

[12] P. S. Heckbert. Survey of texture mapping. *IEEE computer graphics and applications*, 6(11):56–67, 1986.

[13] P. S. Heckbert. Fundamentals of texture mapping and image warping. 1989.

[14] P. S. Heckbert et al. Texture mapping polygons in perspective. Technical report, Citeseer, 1983.

[15] J. Hensley, T. Scheuermann, G. Coombe, M. Singh, and A. Lastra. Fast summed-area table generation and its applications. In *Computer Graphics Forum*, vol. 24, pp. 547–556. Citeseer, 2005.

[16] R. D. Larson and M. S. Shah. Method for generating addresses to textured graphics primitives stored in rip maps, June 22 1993. US Patent 5,222,205.

[17] P. Mavridis and G. Papaioannou. High quality elliptical texture filtering on gpu. In *Symposium on Interactive 3D Graphics and Games*, pp. 23–30, 2011.

[18] J. Novosad. Advanced high-quality filtering. *GPU gems*, 2:417–435, 2005.

[19] M. Pharr, W. Jakob, and G. Humphreys. *Physically based rendering: From theory to implementation*. MIT Press, 2023.

[20] C. Schied, C. Peters, and C. Dachsbacher. Gradient estimation for real-time adaptive temporal filtering. *Proceedings of the ACM on Computer Graphics and Interactive Techniques*, 1(2):1–16, 2018.

[21] L. Shen, J. Feng, and B. Yang. Exponential soft shadow mapping. In *Computer graphics forum*, vol. 32, pp. 107–116. Wiley Online Library, 2013.

[22] L. Shen, G. Guennebaud, B. Yang, and J. Feng. Predicted virtual soft shadow maps with high quality filtering. In *Computer graphics forum*, vol. 30, pp. 493–502. Wiley Online Library, 2011.

[23] G. B. Thomas Jr, M. D. Weir, J. Hass, C. Heil, and T. Edition. Early transcendentals, 2014.

[24] M. Toksvig. Mipmapping normal maps. *journal of graphics tools*, 10(3):65–71, 2005.

[25] L. Williams. Pyramidal parametrics. In *Proceedings of the 10th annual conference on Computer graphics and interactive techniques*, pp. 1–11, 1983.

[26] L.-Q. Yan, M. Hašan, W. Jakob, J. Lawrence, S. Marschner, and R. Ramamoorthi. Rendering glints on high-resolution normal-mapped specular surfaces. *ACM Transactions on Graphics (TOG)*, 33(4):1–9, 2014.

[27] L.-Q. Yan, M. Hašan, S. Marschner, and R. Ramamoorthi. Position-normal distributions for efficient rendering of specular microstructure. *ACM Transactions on Graphics (TOG)*, 35(4):1–9, 2016.

[28] C. Yuksel, S. Lefebvre, and M. Tarini. Rethinking texture mapping. In *Computer Graphics Forum*, vol. 38, pp. 535–551. Wiley Online Library, 2019.