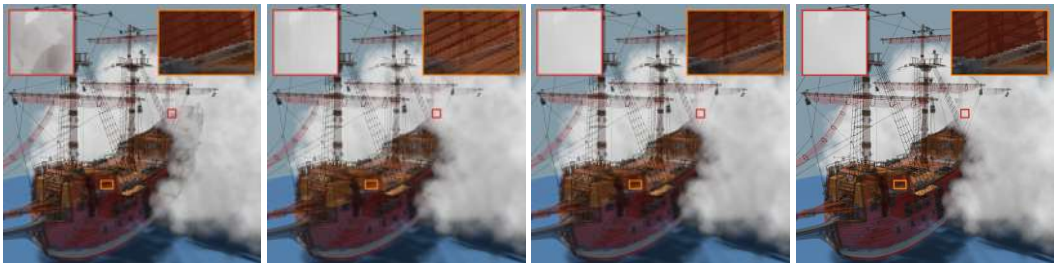
Moment-Based Order-Independent Transparency

CEDRICK MÜNSTERMANN, University of Bonn, Germany
STEFAN KRUMPEN, University of Bonn, Germany
REINHARD KLEIN, University of Bonn, Germany
CHRISTOPH PETERS, Karlsruhe Institute of Technology, Germany and University of Bonn, Germany



Multi-layer alpha blending, 4 layers, 16 ms | Moment-based OIT, ours, 6 power moments, 112 bits, 12 ms | Moment-based OIT, ours, 3 trigonometric moments, 224 bits, 16 ms | Ground truth, depth peeling, 123 ms

Fig. 1. Our novel techniques for order-independent transparency (OIT) use moment-based reconstructions of the transmittance. They can handle complicated participating media and transparent surfaces such as the ship simultaneously. We use the same approach to render shadows for the transparent shadow casters. The timings are full frame times for a resolution of 1024^2 on an NVIDIA GeForce GTX 1080 Ti rendering 49 million transparent fragments.

Compositing transparent surfaces rendered in an arbitrary order requires techniques for order-independent transparency. Each surface color needs to be multiplied by the appropriate transmittance to the eye to incorporate occlusion. Building upon moment shadow mapping, we present a moment-based method for compact storage and fast reconstruction of this depth-dependent function per pixel. We work with the logarithm of the transmittance such that the function may be accumulated additively rather than multiplicatively. Then an additive rendering pass for all transparent surfaces yields moments. Moment-based reconstruction algorithms provide approximations to the original function, which are used for compositing in a second additive pass. We utilize existing algorithms with four or six power moments and develop new algorithms using eight power moments or up to four trigonometric moments. The resulting techniques are completely order-independent, work well for participating media as well as transparent surfaces and come in many variants providing different tradeoffs. We also utilize the same approach for the closely related problem of computing shadows for transparent surfaces.

CCS Concepts: • **Computing methodologies** → **Visibility**; *Rasterization*;

Authors' addresses: Cedrick Münstermann, s6bemuen@uni-bonn.de, University of Bonn, Endenicher Allee 19a, 53115, Bonn, Germany; Stefan Krumpen, krumpen@cs.uni-bonn.de, University of Bonn, Germany; Reinhard Klein, rk@cs.uni-bonn.de, University of Bonn, Germany; Christoph Peters, christoph.peters@kit.edu, Karlsruhe Institute of Technology, Am Fasanengarten 5, 76131, Karlsruhe, Germany, University of Bonn, Germany.

Additional Key Words and Phrases: order-independent transparency, partial coverage, moment-based, power moments, trigonometric moments, moment shadow mapping, transparent shadow casters, shadows, real-time rendering

ACM Reference Format:
Cedrick Münstermann, Stefan Krumpen, Reinhard Klein, and Christoph Peters. 2018. Moment-Based Order-Independent Transparency. *Proc. ACM Comput. Graph. Interact. Tech.* 1, 1, Article 7 (May 2018), 20 pages. https://doi.org/10.1145/3203206

1 INTRODUCTION

Rendering transparent surfaces in real-time is challenging. While over and under blending [Porter and Duff 1984] has had hardware support for decades, it cannot provide correct results unless the geometry is sorted. Sorting all transparent geometry in large dynamic scenes is expensive and in presence of intersections one has to sort fragments per pixel [Carpenter 1984]. The resulting irregular workloads are a poor match for massively parallel graphics hardware.

To avoid this overhead, many heuristics have been proposed. Layer-based approaches accumulate the transparent geometry into a fixed number of layers [Maule et al. 2013; Salvi et al. 2011; Salvi and Vaidyanathan 2014]. This accumulation still depends on the order in which it is performed and thus implementations require hardware that guarantees a deterministic pipeline order. Other heuristics achieve complete order independence by using the same transmittance function independent of the actual transparent geometry [McGuire and Bavoil 2013; McGuire and Mara 2017]. While this approach is fast and easy to implement, results are inaccurate.

We build on recent advances in moment-based reconstructions [Peters and Klein 2015; Peters et al. 2017] to derive a more accurate technique for order-independent transparency (OIT) that relies on programmable shaders and additive blending only. The transmittance, which needs to be accumulated multiplicatively, is represented in terms of its logarithm to enable additive accumulation. An initial additive rendering pass then determines moments of this depth-dependent function (Section 3.1). Once they are available, moment-based reconstructions provide approximations to the transmittance at any depth (Section 3.2). Thus, a second additive rendering pass suffices to composite all transparent surfaces.

When transparent surfaces are plentiful, the transmittance function is complicated. Its adequate representation requires a large number of moments. Prior work in rendering has used up to six power moments, which provide sufficient quality for most use cases (Section 4.1). For more challenging cases, we push to an implementation with eight power moments but find that numerical imprecision lets this approach perform only slightly better (Section 4.2). Therefore, we revisit reconstructions based on trigonometric moments, i.e. Fourier coefficients of a positive signal (Section 5). This leads to a more robust and expressive technique, albeit at a moderate increase in arithmetic operations.

Like OIT, rendering shadows means computing the transmittance. Thus, we apply the exact same approach to render shadows for transparent surfaces through filterable shadow maps (Section 6). Opaque shadow casters are handled separately.

We find that our novel techniques handle complex interactions between participating media and other transparent surfaces faithfully (see Figure 1). The moment-based reconstructions are nearly perfect for few transparent surfaces and smoothly transition to continuous transmittance functions for more complicated situations. Since there are no hard thresholds, the results are entirely free of popping artifacts. Even when no consistent pipeline order is maintained, the results remain unchanged. Intersections of transparent surfaces are more challenging and may be blurred out, dependent on the choice of moments and the number of present surfaces.

2 RELATED WORK

OIT strives to composite $n \in \mathbb{N}$ fragments with colors $L_0, \ldots, L_{n-1}$, opacities $\alpha_0, \ldots, \alpha_{n-1}$ and depths $z_0, \ldots, z_{n-1}$. Application of the over operator [Porter and Duff 1984] in back-to-front order yields the composited color

$$\sum_{l=0}^{n-1} L_l \cdot \alpha_l \cdot \prod_{\substack{k=0 \\ z_k < z_l}}^{n-1} (1 - \alpha_k). \tag{1}$$

The above product describes transmittance. Its dependence on occluding surfaces is the key challenge. Ordered traversal makes the problem simple and may be accomplished by storing all fragments and sorting them explicitly on a per pixel basis [Carpenter 1984]. Hardware-accelerated implementations exist [Yang et al. 2010] but the cost is considerable. Depth peeling uses the depth buffer to achieve rendering in order, one surface at a time [Everitt 2001].

Rather than storing a variable number of fragments per pixel, a k-buffer [Bavoil et al. 2007] only stores $k \in \mathbb{N}$. Where this is insufficient, fragments are merged heuristically. Adaptive volumetric shadow maps [Salvi et al. 2010] and adaptive transparency [Salvi et al. 2011] store an optimal approximation to the transmittance function in a k-buffer. Adaptive transparency then composites in a second pass. Deep shadow maps [Lokovic and Veach 2000] use a similar but more adaptive representation. Multi-layer alpha blending [Salvi and Vaidyanathan 2014] composites into a k-buffer directly, toggling between over- and under blending, dependent on stored depth values. All these heuristics are not order-independent and necessitate a deterministic pipeline order. Hybrid transparency [Maule et al. 2013] works similarly but explicitly uses the layers for the k foremost surfaces. Thus, it is truly order-independent but fails in presence of many surfaces.

Stochastic transparency [Enderton et al. 2011] randomly discards fragments in proportion to their opacity. The expected value is the ground truth but considerable filtering is needed to reduce noise. The same idea has been used for shadows [McGuire and Enderton 2011; McGuire and Mara 2017]. Hashed alpha testing [Wyman and McGuire 2017] makes the noise temporally stable through a deterministic hash.

Various single layer heuristics use an approximation to the transmittance function that is independent of the actual geometry [McGuire and Bavoil 2013; McGuire and Mara 2017; Meshkin 2007]. In weighted blended OIT, transmittance falls of proportional to a user-defined rational function [McGuire and Bavoil 2013]. Phenomenological transparency adds colored transmission and various other effects [McGuire and Mara 2017]. These techniques are very fast, robust, easy to use and do not require a deterministic order. However, the results differ from the ground truth substantially and occlusion cues are lost.

Fourier opacity mapping [Jansen and Bavoil 2010] views transmittance as function of depth and represents it in terms of its logarithm. The resulting absorbance enables additive rather than multiplicative accumulation. As in convolution shadow mapping [Annen et al. 2007], the function is then represented compactly by a Fourier series. The technique is particularly useful for shadows of participating media because the corresponding absorbance functions are low-frequent. Transmittance function mapping [Delalandre et al. 2011] takes a similar approach but represents the transmittance directly. It is computed through ray marching, so order independence is no concern.

Like convolution shadow maps, moment shadow maps [Peters and Klein 2015] offer compact, filterable representations of depth distributions for shadow mapping. Every texel stores four powers of the depth such that filtered samples provide four power moments. An efficient closed form then provides lower and upper bounds to the actual cumulative distribution function. Unlike convolution shadow maps, moment shadow maps are capable of reconstructing sparse signals accurately.

Subsequent work [Peters et al. 2017] has extended the technique to use six power moments and has used it for shadows of transparent surfaces. The approach for transparent surfaces uses a separate technique for OIT to render opaque and transparent surfaces to a single moment shadow map. Another recent work [Peters 2017] investigates more compact storage of four power moments and faster reconstruction.

3 PIPELINE

Our goal is to use an approach similar to Fourier opacity mapping for OIT. Rather than using a Fourier series, we use moments as in moment shadow mapping. In the present section, we introduce our rendering pipeline and regard moment-based reconstructions as a black box. Sections 4 and 5 discuss the various options for the reconstruction in detail.

3.1 Representing Absorbance through Moments

Like most techniques, we handle transparent geometry separately from opaque geometry. All opaque geometry is rendered first using a depth buffer. This depth buffer is kept to render only the visible transparent geometry. Transparent geometry is rendered twice, once to determine the transmittance function per pixel and once to composite all transparent surfaces.

Recall from Equation (1) that the transmittance for a fragment at depth $z_f \in \mathbb{R}$ is given by

$$T(z_f) := \prod_{\substack{l=0 \\ z_l < z_f}}^{n-1} (1 - \alpha_l).$$

Rather than implementing this multiplicative accumulation directly, we work in a logarithmic domain (see Figure 2). We define the absorbance to z_f as

$$A(z_f) := -\ln T(z_f) = \sum_{\substack{l=0 \\ z_l < z_f}}^{n-1} -\ln(1 - \alpha_l).$$

Note that $0 \leq \alpha_l < 1$ and thus $0 \leq -\ln(1 - \alpha_l) < \infty$. Therefore, the absorbance grows monotonically with the depth z_f and can be interpreted as cumulative distribution function of the finite measure

$$Z := \sum_{l=0}^{n-1} -\ln(1 - \alpha_l) \cdot \delta_{z_l}$$

with one Dirac-δ at each surface depth. To store the transmittance, we need a compact representation of this measure. Moment shadow mapping and related techniques provide just that.

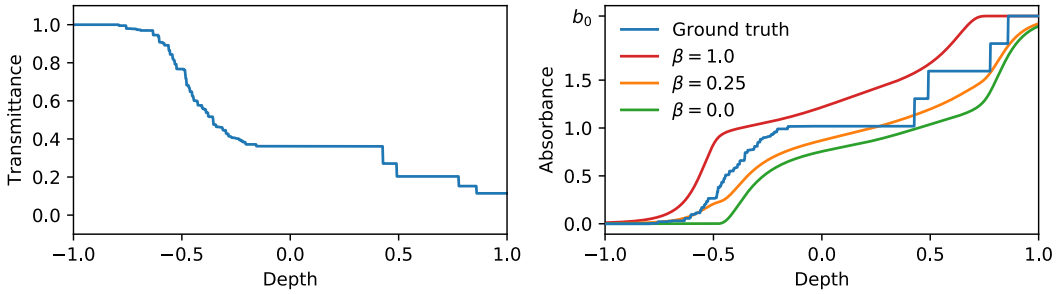
We choose a moment-generating function $\boldsymbol{b} : [-1, 1] \to \mathbb{R}^{m+1}$, e.g. $\boldsymbol{b}(z) = (1, z, z^2, z^3, z^4)^\mathsf{T}$ as in moment shadow mapping [Peters and Klein 2015]. Then the transmittance is stored as

$$b := \mathcal{E}_Z(\boldsymbol{b}) := \sum_{l=0}^{n-1} -\ln(1 - \alpha_l) \cdot \boldsymbol{b}(z_l).$$

A simple rendering pass with additive blending suffices to write this vector into render targets with a total of $m + 1$ channels.

The zeroth moment stores the total absorbance

$$b_0 = \sum_{l=0}^{n-1} -\ln(1 - \alpha_l).$$



(a) Transmittance (b) Absorbance with reconstructions

Fig. 2. Transmittance and absorbance for a view ray that passes through a particle cloud and four transparent surfaces. The cumulative distribution function of the absorbance is reconstructed from six power moments using different amounts of overestimation.

Thus, $\exp(-b_0)$ is the total transmittance which has to be applied as factor to the radiance of opaque surfaces during final compositing. Since it may take any non-negative value, we need to store it explicitly. This is different from moment shadow mapping where Z is a probability distribution and $b_0 = 1$ needs not be stored.

3.2 Reconstructing the Transmittance

The reconstruction algorithm at the heart of moment shadow mapping offers a sharp lower bound to the cumulative distribution function of Z based on b [Peters and Klein 2015, Algorithm 2]. Strictly speaking, this algorithm assumes $b_0 = 1$ but we can always divide b by b_0 and then multiply the end result by b_0. We provide more details in Sections 4 and 5.

With the same methods, an upper bound to the cumulative distribution function can be computed [Peters et al. 2017, Section 5.3]. Using the lower bound guarantees that surfaces do not occlude themselves but at the cost of systematic overestimation of visibility. We found that the overall result is more compelling when interpolating towards the upper bound using a weight around $\beta = 0.25$ (see Figure 2b).

Hence, we obtain an approximation $A(z_f, b, \beta)$ to the absorbance at any depth $z_f \in [-1, 1]$. From that, we compute the transmittance

$$T(z_f, b, \beta) := \exp(-A(z_f, b, \beta)).$$

In our second additive rendering pass, we render and shade all transparent surfaces, compute the absorbance and composite them into an off-screen render target in accordance with Equation (1):

$$\sum_{l=0}^{n-1} L_l \cdot \alpha_l \cdot T(z_f, b, \beta)$$

For pixels with $b_0 = 0$, the fragments have no opacity at all and it is best to do an early return.

3.3 Warping Depth

While it does not make a difference in theory, numerical errors degrade the quality of moment-based reconstructions in practice when the depth range is unreasonably large [Peters and Klein 2015]. Therefore, we compute a conservative bounding sphere around all transparent geometry in the scene, derive a reasonably sharp depth range $[z_{\min}, z_{\max}]$ from this sphere and map it to $[-1, 1]$.

Note that $z_{\min} > 0$ thanks to the near clipping plane. This way, the bounds never change abruptly and results are temporally coherent.

Naturally, nearby transparent surfaces tend to be more visible than distant surfaces because transmittance always decreases monotonically. In consequence, we additionally allocate more precision for nearby geometry by warping depth logarithmically. Doing so consistently improves the visual quality. Thus, the depth values z that enter all moment-based computations are computed from linear view-space depth values $z_v \in \mathbb{R}$ by

$$z := \frac{\ln z_v - \ln z_{\min}}{\ln z_{\max} - \ln z_{\min}} \cdot 2 - 1 \in [-1, 1].$$

3.4 Final Compositing

After the opaque pass and the two additive passes for transparent geometry, we have two render targets showing opaque and transparent surfaces, respectively. The obvious way to composite them is to multiply the opaque render target by the total transmittance $\exp(-b_0)$ and to add the transparent render target. If the outgoing radiance of the opaque surface is L_n, this leads to the end result

$$\exp(-b_0) \cdot L_n + \sum_{l=0}^{n-1} L_l \cdot \alpha_l \cdot T(z_f, b, \beta).$$

However, there is a problem with this approach; it is not energy conserving. The total weight of all radiances combined is not necessarily one. While the total transmittance is accurate, the reconstructed transmittance for the individual surfaces may be off and these errors need not cancel out. To get a less biased result, we explicitly renormalize as in weighted blended OIT [McGuire and Bavoil 2013]:

$$\exp(-b_0) \cdot L_n + \frac{1 - \exp(-b_0)}{\sum_{l=0}^{n-1} \alpha_l \cdot T(z_f, b, \beta)} \cdot \sum_{l=0}^{n-1} L_l \cdot \alpha_l \cdot T(z_f, b, \beta) \tag{2}$$

The normalization value in the denominator is simply written to the alpha channel during the second additive rendering pass.

This renormalization imposes almost no additional cost. In some cases, the estimate for the transmittance on individual surfaces becomes less accurate but overall artifacts are diminished (see Figure 7).

4 POWER MOMENTS

Thus far, we have viewed reconstruction of the cumulative distribution function of the absorbance Z from the vector of moments $b \in \mathbb{R}^{m+1}$ as a black box. In the following, we will provide more details on this step and discuss a broad range of alternatives offering different tradeoffs.

4.1 Four or Six Power Moments

When we use m power moments, i.e.

$$\boldsymbol{b}(z) = (1, z, z^2, \ldots, z^m)^\mathsf{T}$$

with m even, Algorithm 1 provides the desired reconstruction [Peters et al. 2017]. Conceptually, it deals with a specific depth distribution S, which consists of Dirac-δ distributions at the computed depths $z_0, \ldots, z_{\frac{m}{2}} \in \mathbb{R}$. This depth distribution realizes the given power moments exactly, i.e. $\mathcal{E}_S(\boldsymbol{b}) = b$ [Peters and Klein 2015, Proposition 10]. At the same time, the values of its cumulative distribution function around $z_f = z_0$ are extremal among all such depth distributions [Peters and Klein 2015, Proposition 8]. Therefore, it provides access to the desired lower and upper bounds for

Algorithm 1 Reconstruction from power moments [Peters et al. 2017].
Input: Moment count $m \in \mathbb{N}$ even, power moments $b_j = \mathcal{E}_Z(z^j) \in \mathbb{R}$ for all $j \in \{0, \dots, m\}$ and $z(z) := z$, fragment depth $z_f \in [-1, 1]$, overestimation weight $\beta \in [0, 1]$.
Output: The approximation $A(z_f, b, \beta)$.

- Set $z_0 := z_f$ and $v_0 := \beta$,
- Use a Cholesky decomposition to solve for $q \in \mathbb{R}^{\frac{m}{2}+1}$:

$$\begin{pmatrix} b_0 & b_1 & \cdots & b_{\frac{m}{2}} \\ b_1 & b_2 & \iddots & b_{\frac{m}{2}+1} \\ \vdots & \iddots & \iddots & \vdots \\ b_{\frac{m}{2}} & b_{\frac{m}{2}+1} & \cdots & b_m \end{pmatrix} \cdot q = \begin{pmatrix} 1 \\ z_0^1 \\ \vdots \\ z_0^{\frac{m}{2}} \end{pmatrix}$$

- Solve $q_{\frac{m}{2}} \cdot z^{\frac{m}{2}} + \cdots + q_0 \cdot z^0 = 0$ for z to obtain roots $z_1, \dots, z_{\frac{m}{2}} \in \mathbb{R}$,
- For $1 \le l \le \frac{m}{2}$ set $v_l := 1$ if $z_l < z_0$ and $v_l := 0$ otherwise,
- Use divided differences to solve a Vandermonde system for $u \in \mathbb{R}^{\frac{m}{2}+1}$:

$$\begin{pmatrix} 1 & z_0^1 & \cdots & z_0^{\frac{m}{2}} \\ \vdots & \vdots & \ddots & \vdots \\ 1 & z_{\frac{m}{2}}^1 & \cdots & z_{\frac{m}{2}}^{\frac{m}{2}} \end{pmatrix} \cdot u = \begin{pmatrix} v_0 \\ \vdots \\ v_{\frac{m}{2}} \end{pmatrix},$$

- Return $\sum_{j=0}^{\frac{m}{2}} b_j \cdot u_j$.

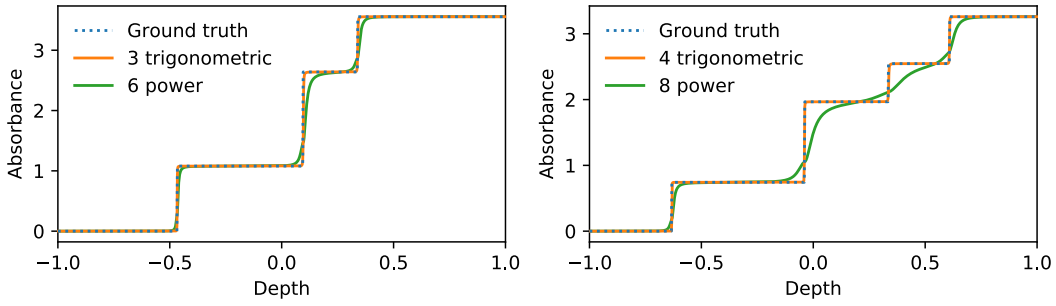
$A(z_f)$. Since our available knowledge does not preclude the case $S = Z$, these bounds are the best attainable bounds.

In practice, numerical stability is paramount to avoid artifacts. Thus, it is important to use a robust polynomial solver and the stated linear solvers. For $m = 4$ and $m = 6$, we rely on available robust implementations from prior work [Peters et al. 2017].

Application of these implementations in the pipeline outlined above is most straightforward when using one single-precision float per moment. Since Algorithm 1 is usually implemented for the special case $b_0 = 1$, we input $\frac{b}{b_0}$ and multiply the end result by b_0. The biasing, which compensates for rounding errors, is also applied to this normalized vector. For six power moments stored in single precision, we need to redo the optimization of the biasing strategy [Peters et al. 2017] since this variant has not been used before. Details on appropriate biasing strategies for all discussed techniques are given in the supplementary.

As with moment shadow mapping, it is worthwhile to use a quantization with only 16 bits per moment to reduce bandwidth requirements. However, the existing quantization schemes exploit $b_0 = 1$. Since this assumption fails in the present application, values may end up being out of range. Our solution is to store the normalized moments $\frac{b_1}{b_0}, \dots, \frac{b_m}{b_0}$ using the existing quantization schemes [Peters et al. 2017] while storing b_0 separately. Due to the greater dynamic range of b_0, a half-precision float works best.

An implementation of the additive blending then requires that the stored vector is multiplied by b_0 before addition and divided by the updated b_0 before storage. This read-modify-write operation still does not require a deterministic pipeline order but on current hardware corresponding features offer the most practical implementation.



(a) Three surfaces (b) Four surfaces

Fig. 3. Moment-based reconstructions in ideal cases with $\beta = 0.25$ and the recommended biasing for single-precision floats. In theory, the techniques could produce perfect reconstructions here. In practice, the biasing that is needed to compensate rounding errors degrades the reconstruction quality. For power moments, this problem is much greater than for trigonometric moments.

A problem with such an additive render pass to a 16-bit render target is that rounding errors accumulate as the number of fragments increases. The usual biasing strategies are still an effective countermeasure but the required strength of the bias is no longer scene independent. Nonetheless, appropriate bias values are easy to find. Even for the complicated scene in Figure 1, the usual bias values did not need to be increased by more than one order of magnitude.

We also tried using non-linear quantization [Peters 2017] to benefit from reduced rounding errors without greater bandwidth requirements. However, the cost added to the read-modify-write operation through non-linear dequantization and quantization turned out to be too big to be justified in this setting.

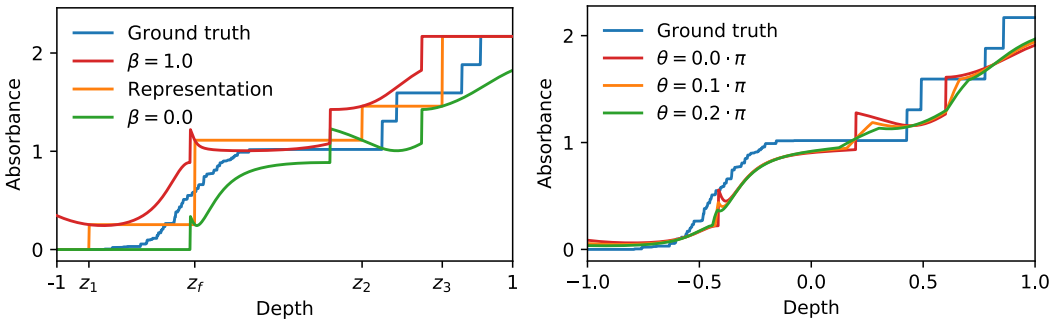
4.2 Eight Power Moments

Prior work in real-time graphics has never used more than six power moments. Since transparent surfaces may occur in very complicated configurations, we now investigate the extension to $m = 8$ for the sake of a more expensive but also more powerful technique. Concerning the linear solvers, the recommendations in Algorithm 1 still work well for $m = 8$.

The remaining challenge is to find a robust polynomial solver for quartic polynomials with four real roots. After trying a variety of iterative, closed-form and hybrid solvers, we settled for a closed-form solution [Neumark 1965] which gave the most robust results efficiently. Internally, this solver uses the root of smallest magnitude of the resolvent cubic, which we compute using another robust solver [Blinn 2007]. It then factorizes the quartic into two quadratic polynomials. Their roots are the end result. To avoid numerical cancellations, two different approaches to this factorization are used adaptively [Herbison-Evans 1995].

Finally, we need to derive new quantization transforms for 16-bit quantization and optimal biasing strategies. We compute these exactly as in the case $m = 6$ [Peters et al. 2017] and provide the resulting matrices and vectors in the supplementary among with complete code for the reconstruction algorithm.

In theory, eight power moments capture complicated distributions significantly better than six. Distributions with up to four transparent surfaces can be reconstructed perfectly. In practice, rounding errors are a major concern. Even with single precision floats, these errors necessitate a biasing that diminishes the benefit of the greater number of power moments (see Figure 3).



(a) Construction of the approximations

(b) Effect of the wrapping zone

Fig. 4. Various reconstructions of a depth distribution using three trigonometric moments. To the left, we show how the upper and lower approximation are formed from the representation S. To the right, we show how a larger wrapping zone smoothes the reconstruction for $\beta = 0.25$.

5 TRIGONOMETRIC MOMENTS

Moment shadow mapping has been introduced alongside an alternative called trigonometric moment shadow mapping [Peters and Klein 2015], which uses Fourier basis functions as moment-generating function $\boldsymbol{b}$. It has proven to be much less vulnerable to rounding errors. However, the arithmetic overhead is prohibitive because computing the sharp lower bound is far more complicated.

5.1 Approximate Bounds

For OIT, having a lower bound is not crucial. In fact, we explicitly interpolate towards the upper bound to reduce the systematic error. We exploit this for a far more efficient algorithm. The idea is to use the same principles as in Algorithm 1 to find a sparse depth distribution with the given trigonometric moments and a Dirac-δ at z_f. The cumulative distribution function of this reconstruction at z_f then provides approximations to the original cumulative distribution function (see Figure 4a).

Algorithm 2 implements this approach. Every single step is analogous to Algorithm 1. Rather than dealing with real power moments, it takes complex trigonometric moments. Our depth range is associated with the complex unit circle through

$$\boldsymbol{x}(z) := \exp\left(2 \cdot \pi \cdot i \cdot \frac{z+1}{2}\right) \in \mathbb{C}. \tag{3}$$

Our moment generating function consists of powers of $\boldsymbol{x}$, i.e. $\boldsymbol{b}(z) = (1, \boldsymbol{x}^1(z), \ldots, \boldsymbol{x}^m(z))^\mathsf{T}$. In other words, $\boldsymbol{b}(z)$ is a vector of complex Fourier basis functions.

Implicitly, Algorithm 2 computes the unique depth distribution $S := \sum_{l=0}^{m} w_l \cdot \delta_{z_l}$ with $z_l := \boldsymbol{x}^{-1}(x_l)$ and $w_0, \ldots, w_m > 0$ such that all given trigonometric moments match exactly [Peters and Klein 2015, Propositions 15 and 16], i.e. for all $j \in \{0, \ldots, m\}$

$$\mathcal{E}_S(\boldsymbol{x}^j) = \sum_{l=0}^{m} w_l \cdot x_l^j = b_j.$$

This is one of infinitely many distributions representing these trigonometric moments. By construction, it has a non-zero weight w_0 at z_f. The last three steps of Algorithm 2 evaluate $w^\mathsf{T} \cdot v$,

Algorithm 2 Reconstruction from trigonometric moments.
Input: Moment count $m \in \mathbb{N}$, trigonometric moments $b_j = \mathcal{E}_Z(\boldsymbol{x}^j) \in \mathbb{C}$ for all $j \in \{0, \dots, m\}$ (see Equation (3) or (5)), fragment depth $z_f \in [-1, 1]$, overestimation weight $\beta \in [0, 1]$.
Output: The approximation $A(z_f, b, \beta)$.

- Set $x_0 := \boldsymbol{x}(z_f) \in \mathbb{C}$ (see Equation (3) or (5)) and $v_0 := \beta$,
- Use a Cholesky decomposition to solve for $q \in \mathbb{C}^{m+1}$:

$$\begin{pmatrix} b_0 & \overline{b_1} & \cdots & \overline{b_m} \\ b_1 & b_0 & \ddots & \vdots \\ \vdots & \ddots & \ddots & \overline{b_1} \\ b_m & \cdots & b_1 & b_0 \end{pmatrix} \cdot q = \begin{pmatrix} 1 \\ x_0^1 \\ \vdots \\ x_0^{\frac{m}{2}} \end{pmatrix},$$

- Solve $q_m \cdot x^m + \cdots + q_0 \cdot x^0 = 0$ for x to obtain roots $x_1, \dots, x_m \in \mathbb{C}$,
- For $1 \leq l \leq m$ set $v_l := \boldsymbol{v}(x_l, x_0)$ (see Equation (4) or (6)),
- Use divided differences to solve a Vandermonde system for $u \in \mathbb{C}^{m+1}$:

$$\begin{pmatrix} 1 & x_0^1 & \cdots & x_0^m \\ \vdots & \vdots & \ddots & \vdots \\ 1 & x_m^1 & \cdots & x_m^m \end{pmatrix} \cdot u = \begin{pmatrix} v_0 \\ \vdots \\ v_m \end{pmatrix},$$

- Return $b^{\mathsf{T}} \cdot u$.

which is the cumulative distribution function at z_f [Peters et al. 2017, Section 5.4.2]. To this end, the contribution of each point needs to be determined, which is given by

$$v_l = \boldsymbol{v}(x_l, x_0) := \begin{cases} 1 & \text{if } \boldsymbol{x}^{-1}(x_l) < \boldsymbol{x}^{-1}(x_0), \\ 0 & \text{otherwise.} \end{cases} \tag{4}$$

The parameter $v_0 = \beta$ determines the contribution of the weight w_0 at z_f. When it is zero, the result will be closer to a lower bound, when it is one, it will be closer to an upper bound. Either way, we do not get proper bounds but at least we have some control over whether we under- or overestimate (see Figure 4a).

5.2 The Wrapping Zone

While the lack of proper bounds is unproblematic by itself, it does have some negative consequences. Most notably, the reconstruction is not continuous. This problem stems from a discontinuity in $\boldsymbol{v}(x_l, x_0)$ for $x_l = 1$ where depth values wrap around. It also has a discontinuity for $x_l = x_0$ but this case cannot occur [Kreĭn and Nudel'man 1977, Theorem IV.4.2]. Since the discontinuities are easy to spot in the results, we address the issue by modifying $\boldsymbol{v}(x_l, x_0)$.

We introduce the wrapping zone angle $0 < \theta \ll 2 \cdot \pi$ controlling the size of a region that does not correspond to any depth values. It is used solely to wrap depth values around in a continuous fashion. Hence, we redefine

$$\boldsymbol{x}(z) := \exp\left((2 \cdot \pi - \theta) \cdot i \cdot \frac{z + 1}{2}\right). \tag{5}$$

As we change $\boldsymbol{v}(x_l, x_0)$, we note that it is inefficient to evaluate $\boldsymbol{x}^{-1}$ because it involves inverse trigonometric functions. We replace it by a piecewise linear function that is monotonic in the same

sense, namely

$$\boldsymbol{\psi}(x) := \begin{cases} -\Re x + \Im x & \text{if } \Re x \geq 0,\ \Im x \geq 0, \\ 2 - \Re x - \Im x & \text{if } \Re x < 0,\ \Im x \geq 0, \\ 4 + \Re x - \Im x & \text{if } \Re x < 0,\ \Im x < 0, \\ 6 + \Re x + \Im x & \text{if } \Re x \geq 0,\ \Im x < 0. \end{cases}$$

Then continuous factors for our weights can be computed as

$$\boldsymbol{v}(x_l, x_0) := \begin{cases} 1 & \text{if } \boldsymbol{\psi}(x_l) < \boldsymbol{\psi}(x_0), \\ 0 & \text{if } \boldsymbol{\psi}(x_0) \leq \boldsymbol{\psi}(x_l) \leq \boldsymbol{\psi}(\boldsymbol{x}(1)), \\ \frac{\boldsymbol{\psi}(x_l) - \boldsymbol{\psi}(\boldsymbol{x}(1))}{7 - \boldsymbol{\psi}(\boldsymbol{x}(1))} & \text{otherwise.} \end{cases} \tag{6}$$

Increasing the size of the wrapping zone θ trades smoothness for depth precision. The reconstructions are continuous for any $\theta > 0$ but may still change rapidly (see Figure 4b). In our experiments, we use $\theta = \frac{\pi}{10}$.

5.3 Three or Four Trigonometric Moments

For the most part, the implementation of Algorithm 2 is very similar to that of Algorithm 1. The major difference is that complex numbers replace most real numbers. Therefore, m trigonometric moments take as much space as $2 \cdot m$ power moments but carry a similar amount of information. In fact, they provide better reconstructions at higher order due to greater stability. However, the complex operations require more arithmetic instructions. With two trigonometric moments, the benefit from the improved stability is too small to justify this cost.

Once more, the remaining challenge for stable implementations at higher order is the polynomial solver. Since all roots $x_1, \ldots, x_m$ are known to lie on the unit circle, the problem is quite well-behaved. For three trigonometric moments, we started from a naive generalization of Blinn's solver [Blinn 2007] to the complex case without any branches and got only few artifacts. Internally, this solver computes a root of a quadratic polynomial. To eliminate the remaining artifacts, it suffices to avoid cancellation by picking the root of the quadratic with greater magnitude. A similar generalization of Neumark's solver [Neumark 1965] to the complex case works robustly if the chosen root of the depressed cubic is not the one with least magnitude.

Biasing is implemented as in trigonometric moment shadow mapping [Peters and Klein 2015] by multiplying $b_1, \ldots, b_m$ by an appropriate constant $1 - \alpha$ (see supplementary). Since trigonometric moments are less vulnerable to rounding errors, there is no need for a quantization transform when storing them in 16 bits each.

6 SHADOWS

While OIT deals with primary visibility, rendering shadows corresponds to secondary visibility. In both cases, the core problem is computation of the transmittance. Thus, most techniques for OIT may be used for shadows and vice versa. For example, stochastic transparency [Enderton et al. 2011] and colored stochastic shadow maps [McGuire and Enderton 2011] use the same basic principle. We will now use our moment-based OIT for shadows. This leads us to the moment-based analog of Fourier opacity mapping [Jansen and Bavoil 2010].

As with OIT, we handle opaque and transparent shadow casters separately. For opaque shadow casters, any shadow technique may be used. We choose moment shadow mapping [Peters and Klein 2015]. Then an additive pass renders transparent shadow casters to a separate moment shadow map, thus accumulating a representation of absorbance as described in Section 3.1. Except for the different coordinate transform, this works in exactly the same manner and any choice of moments

in Sections 4 and 5 is valid. Our implementation still determines the depth range from a bounding sphere but depth values are defined linearly. In any case, the zeroth moment needs to be stored. It represents the transmittance to infinite depth. Storing it explicitly is benefitial to the quality of shadows on distant receivers such as a ground plane.

To diminish aliasing, our implementation filters both moment shadow maps with a 9×9 Gaussian and uses bilinear interpolation. This works fine when the moments are stored in single-precision floats but causes problems at 16 bits per moment. Since division is non-linear, the normalized vectors of moments $\frac{b_1}{b_0}, \ldots, \frac{b_m}{b_0}$ should not be filtered linearly. Doing so may give acceptable results in some cases but leads to obvious artifacts when the filter region includes texels without transparent surfaces where b_0 vanishes. For proper filtering, the normalization must be undone before each filtering operation. Our implementation ended up being slower than the one using single-precision floats. Hence, we discard 16-bit quantization in this context but note that specialized compute shaders should make it practical [Peters 2017].

To shade a fragment, we first compute the filtered transmittance through opaque shadow casters as usual. Then we perform a reconstruction of the transmittance through transparent shadow casters as explained in Sections 3.2, 4 and 5. To combine both results, we assume that the latter transmittance is constant within the filter region. Under this assumption, the appropriate combination operator is simple multiplication. Finally, we multiply the incoming irradiance from the light by this transmittance product and proceed with shading.

The assumption of constant transmittance fails quickly when transparent shadow casters are discarded using the depth buffer of opaque shadow casters. This leads to leaking whenever the silhouette of an opaque shadow caster shadows transparent shadow casters. This artifact is easily avoided by not using the depth buffer for transparent shadow casters.

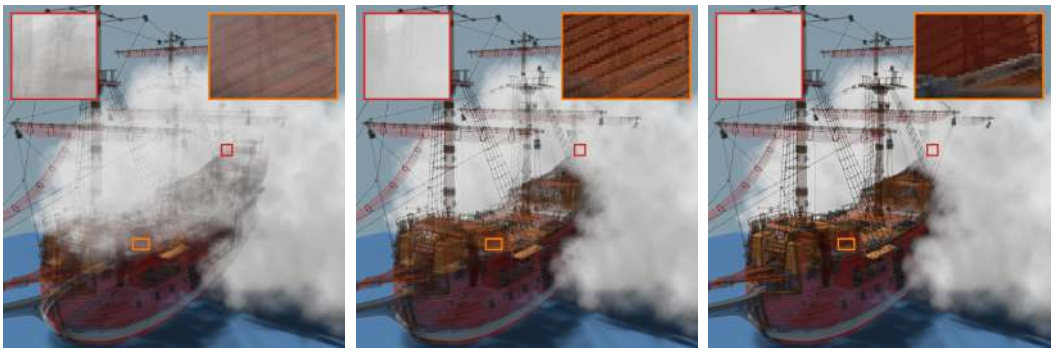
7 RESULTS

We evaluate with a forward renderer using Direct3D 11.3 and compare against adaptive transparency [Salvi et al. 2011], multi-layer alpha blending [Salvi and Vaidyanathan 2014] and weighted blended OIT [McGuire and Bavoil 2013]. Whenever needed for a deterministic pipeline order, we use rasterizer ordered views. All techniques, including multi-layer alpha blending, render colors to render targets with 16 bits per channel. Our implementation of weighted blended OIT uses the same weighting function as phenomenological transparency [McGuire and Mara 2017].

We compare our shadows to Fourier opacity mapping [Jansen and Bavoil 2010] but do not use this technique for primary visibility since prior work has shown that the quality would be poor [Salvi et al. 2010]. For moment-based OIT, we always use an overestimation weight of $\beta = 0.25$. Unless stated otherwise, we render opaque shadows using a 64-bit moment shadow map and transparent shadows using six power moments stored in 224 bits. The shadow map resolution is 1024^2.

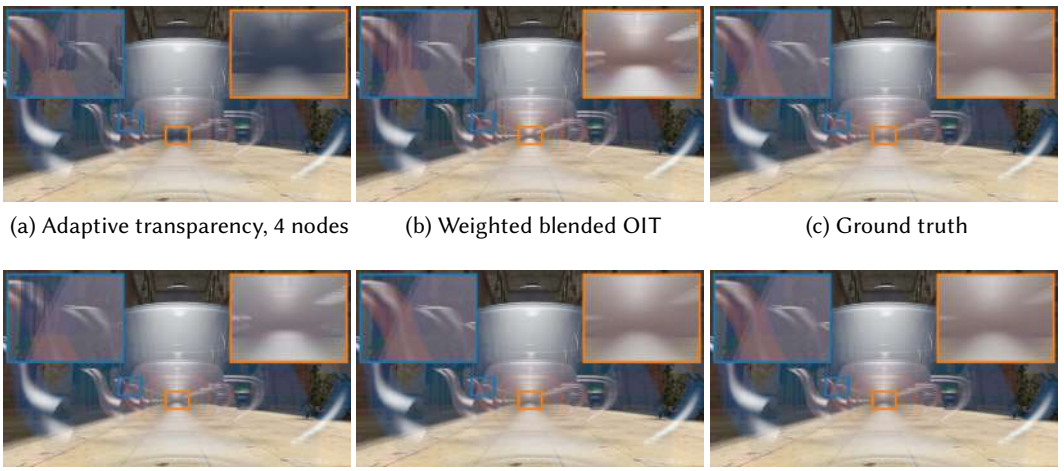
7.1 Quality of OIT

Figures 1 and 5 compare a selection of techniques on a complex scene that combines large particle clouds with a complicated transparent model of a ship. The particles are simply camera-oriented billboards. While multi-layer alpha blending [Salvi and Vaidyanathan 2014] works relatively well for the ship, it fails to accomplish proper occlusion through the particles and produces discontinuities along particle boundaries. It can be made to work well with six layers by sorting all particles and rendering them after the ship. Results of adaptive transparency [Salvi et al. 2011] are worse and not shown here. Weighted blended OIT [McGuire and Bavoil 2013] yields smooth results but does not convey the mutual occlusion of the fog and the ship correctly.



(a) Weighted blended OIT, 5.5 ms (b) Moment-based OIT, ours, 4 power moments, 80 bits, 10 ms (c) Ground truth, depth peeling, 123 ms

Fig. 5. Two additional results and the ground truth for the scene in Figure 1.



(a) Adaptive transparency, 4 nodes (b) Weighted blended OIT (c) Ground truth

(d) Multi-layer alpha blending, 4 layers (e) Moment-based OIT, ours, 4 power moments, 80 bits (f) Moment-based OIT, ours, 6 power moments, 112 bits

Fig. 6. A scene with 17 transparent teapots standing in a row on the ground in Sponza. The curtains are transparent as well. Note how the overdraw increases towards the center.

Our moment-based OIT produces a smooth result as well while capturing occlusions far more accurately. With four power moments, results are plausible overall but some surfaces appear to be occluded by nearby surfaces in the background. This is particularly true around the main mast. Using six power moments, results are close to the ground truth almost everywhere. Only in places where surfaces with substantially different colors are close, there is some leaking. These errors vanish almost entirely when using three trigonometric moments stored in single precision. To demonstrate the excellent temporal stability of our continuously defined reconstruction, we show an animated view of this scene in the supplementary video.

(a) Without renormalization (b) With renormalization

Fig. 7. A red teapot occluding alpha mapped foliage using moment-based OIT with six power moments. Without renormalization, the transparent parts of the foliage are too dark. Renormalization removes this artifact but propagates part of the error to the teapot.

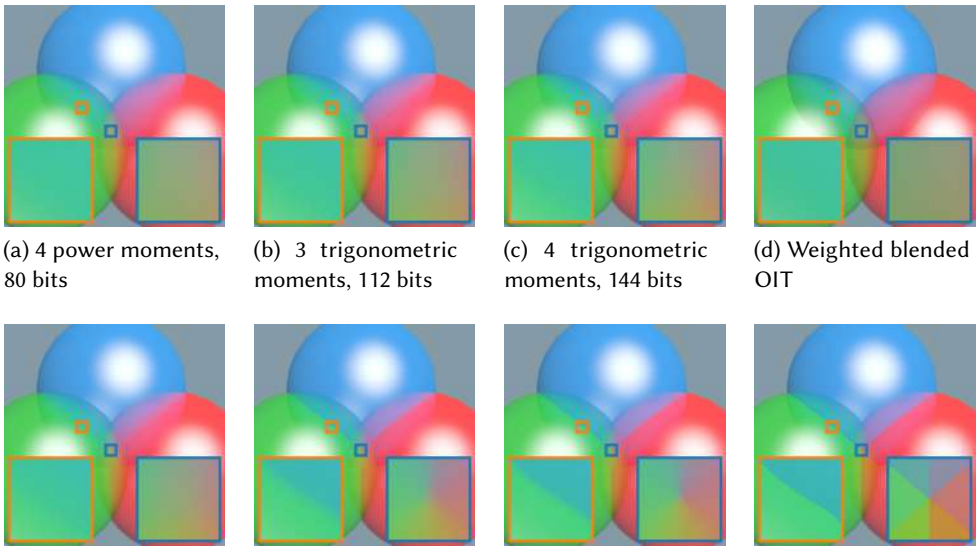
Figure 6 shows a less challenging test case. Multi-layer alpha blending performs well except for some artifacts where a curtain is improperly occluded by three teapots. Adaptive transparency shares similar artifacts but additionally underestimates the transmittance in the center region where many teapots overlap. Weighted blended OIT generally lets the curtains and the hindmost teapots appear too visible. With our moment-based OIT using four power moments, the visibility on some of the curtains is overestimated slightly but the result is plausible. Using six power moments improves on this situation further.

In Figure 6, we use an alpha test without blending for the foliage. Using our moment-based OIT triggers a failure case that we show in Figure 7. Through the overestimation of $\beta = 0.25$, the transparent parts of the foliage end up being too dark. Next to the opaque parts, this is an obvious artifact. By default we use the renormalization described in Equation (2), which removes this artifact. However, it may end up propagating part of the error to another transparent surface. Weighted blended OIT performs a similar renormalization leading to similar artifacts. Overall, our technique is not ideal for rapidly changing alpha maps. Hashed alpha testing is specifically designed for this case and can step in at little cost [Wyman and McGuire 2017].

Unlike most other techniques, our approach is free of depth comparisons that lead to discontinuities in the transmittance function. In many cases, this is a benefit because it leads to smooth and stable results. However, it turns into a problem when dealing with intersecting geometry as shown in Figure 8. All techniques storing moments in 16 bits fail to produce a sharp line where the spheres intersect. Though, using trigonometric moments improves sharpness. With four trigonometric moments stored in single precision floats, the intersection of two spheres becomes sharp but even this variant blurs the intersection of three spheres. Weighted blended OIT shares this artifact but multi-layer alpha blending with two layers or adaptive transparency with four nodes obtain a perfect result. Note that we artificially enlarged the depth range for this example to be more than ten times the sphere radius.

7.2 Quality of Shadows

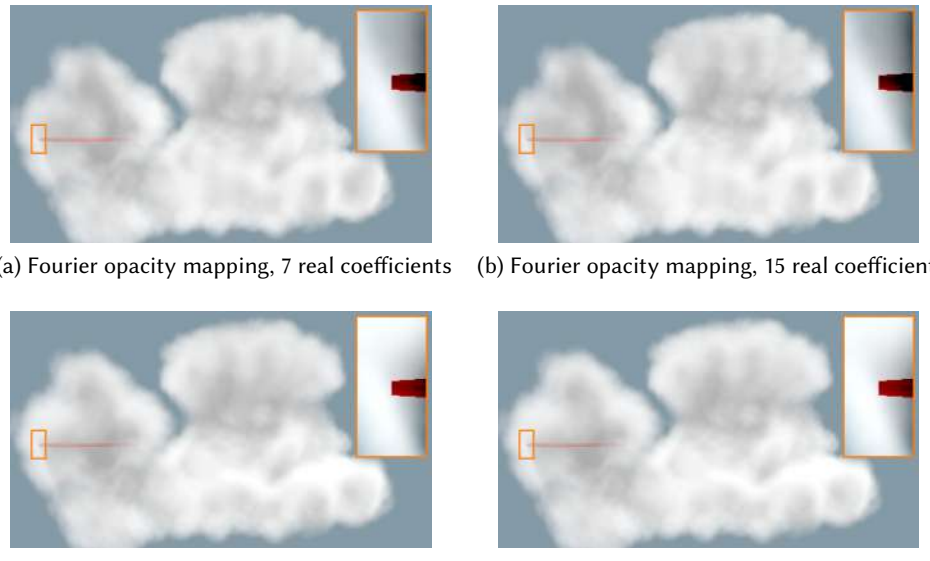
Like all results shown above, Figure 1 uses moment-based OIT shadows with six power moments stored in 112 bits for the transparent shadows. The combination of particle clouds and shadows leads to single scattering, the particles shadow themselves and the ground receives partial shadow as expected. Some light leaking does occur but for transparent shadows it is less objectionable than for opaque shadows [McGuire and Mara 2017].



(a) 4 power moments, 80 bits (b) 3 trigonometric moments, 112 bits (c) 4 trigonometric moments, 144 bits (d) Weighted blended OIT

(e) 6 power moments, 224 bits (f) 3 trigonometric moments, 224 bits (g) 4 trigonometric moments, 288 bits (h) Ground truth

Fig. 8. Three intersecting spheres rendered with various techniques. Note how different choices of moments and quantization schemes yield different amounts of blur.



(a) Fourier opacity mapping, 7 real coefficients (b) Fourier opacity mapping, 15 real coefficients

(c) Moment-based OIT shadows, ours, 4 power moments, 160 bits (d) Moment-based OIT shadows, ours, 6 power moments, 224 bits

Fig. 9. A transparent particle cloud rendered with shadows. The left part contains a transparent, red disk to obtain a sharp, volumetric shadow. Note how Fourier opacity mapping fails to produce a hard onset for this shadow.

(a) Teapots

(b) Ship

Fig. 10. The scenes used for OIT run time measurements. They consist of $3.6 \cdot 10^5$ and $9.5 \cdot 10^5$ transparent triangles, respectively. At a resolution of 1920×1080, these views show $9.4 \cdot 10^6$ and $18.7 \cdot 10^6$ transparent fragments, respectively.

Figure 9 compares our approach for moment-based OIT shadows to Fourier opacity mapping [Jansen and Bavoil 2010]. While the other results of our technique use $\beta = 0$ to avoid wrong self-shadowing on transparent surfaces, this scene uses $\beta = 0.5$ for a more meaningful comparison. Overall, this scene plays to the advantage of Fourier opacity mapping because the depth range is small and the transmittance function is changing smoothly within the particle cloud. Nonetheless, the shadows are smeared out along the light direction. The directly lit top of the cloud is already slightly darkened. To make this effect more visible, we placed a red disk with 94% opacity in the left part of the cloud. With moment-based OIT shadows, we obtain a reasonably hard onset for the shadow while the shadow is clearly visible above the disk when using Fourier opacity mapping.

Except for a slight reduction in wrong self-shadowing with six power moments, results for the two variants of our technique are similar in this example. Though with $\beta = 0$, the complicated depth distributions arising from transparent surfaces cause a lot of leaking when using four power moments. In static images, these artifacts are hard to spot but in animated sequences they can be obvious as demonstrated in the supplementary video. We recommend using six power moments for improved robustness. The extra cost for the use of trigonometric moments is hard to justify because with monochromatic shadows artifacts are less noticeable than with OIT on colored surfaces.

7.3 Run Time

We measure run times on a computer with an NVIDIA GeForce GTX 1080 Ti and an Intel Core i7-8700K running Windows 10. To evaluate OIT techniques, we use the teapots and a version of the ship with smaller particle clouds (see Figure 10). With adaptive transparency and multi-layer alpha blending, the memory layout is critical for a good run time. After some experimentation, we decided to index the data within 16×16 tiles through Morton codes to improve cache locality. Depth values for these techniques are stored in 16-bit fixed-point numbers using the warped depth from Section 3.3. Table 1 lists the key results. Additional timings are provided in the supplementary document.

Adaptive transparency with four nodes takes slightly more time than moment-based OIT with six power moments in 112 bits. Timings for multi-layer alpha blending with four layers are slightly longer than for moment-based OIT with three trigonometric moments in 224 bits. With two layers, the run time is inbetween the two variants with four power moments. In all three cases, our techniques clearly offer the higher quality, although this would not be true for all scenes (see e.g. Figure 8). Weighted blended OIT is significantly faster than all of our techniques but much less accurate.

Table 1. Run times in ms for the two scenes in Figure 10. Differential timings are full frame times where the frame time for rendering the scenes through alpha blending without any sorting has been subtracted. Adaptive transparency and moment-based OIT use a third pass for compositing that takes 0.06 ms. We also list the required additional memory in bits per pixel, not counting color buffers for accumulation.

OIT technique		Alpha blend	Adaptive transparency	Multi-layer alpha blend		Weighted blended
Nodes/layers		-	4	2	4	-
Memory		0	128	160	320	8
Teapots	Pass 1	0.79	1.6	2.0	3.4	0.85
	Pass 2	-	0.94	0.13	0.22	0.07
	Diff.	0.0	1.8	1.4	2.8	0.10
Ship	Pass 1	1.4	3.0	3.9	6.6	1.6
	Pass 2	-	1.8	0.11	0.20	0.06
	Diff	0.0	3.3	2.6	5.3	0.16

OIT technique		Moment-based OIT (ours)							
Moment type		Power				Trigonometric			
Moment count		4		6		3		4	
Memory		80	160	112	224	112	224	144	288
Teapots	Pass 1	0.94	1.2	1.2	1.6	1.2	1.6	1.3	1.9
	Pass 2	1.0	1.0	1.1	1.2	1.7	1.7	2.4	2.4
	Diff.	1.2	1.5	1.6	2.1	2.2	2.6	3.0	3.6
Ship	Pass 1	2.0	2.3	2.3	3.1	2.3	3.1	2.9	3.6
	Pass 2	1.8	1.8	2.2	2.2	3.2	3.2	4.5	4.8
	Diff.	2.3	2.7	3.1	3.8	4.2	4.9	5.7	6.8

Note that, unlike multi-layer alpha blending and weighted blended OIT, our techniques require two transparent passes. In our implementation, the entire geometry pipeline runs twice. This overhead could be reduced by reusing transformed vertex data.

Comparing the variants of our technique, we note that their run times are perfectly sorted the way they are listed. This progression nicely matches the increases in quality, with the exception of four trigonometric moments stored in 144 bits. Four or six power moments or three trigonometric moments offer sensible tradeoffs. The quality gains from using four trigonometric moments are relatively small, so the additional cost might be hard to justify. In the supplementary document, we demonstrate that results with eight power moments are slightly better than with six power moments at a cost close to that of three trigonometric moments.

The timings for the second pass of our techniques appear to be limited by compute since they are independent of the used quantization. This cost rises substantially if we use more moments or trigonometric moments. On the other hand, timings for the first pass depend strongly on the quantization, which suggests that it is bandwidth limited. For adaptive transparency, the second pass is relatively inexpensive whereas the first pass has a greater cost due to more arithmetic work.

Table 2 lists frame times for rendering shadows of transparent surfaces. While Fourier opacity mapping is always faster with 7 real coefficients, the variant with 15 coefficients often falls inbetween our techniques with four or six power moments. In spite of the lower bandwidth usage, generating a shadow map with six power moments takes more time than a Fourier opacity map with 15 coefficients. This appears to be due to a lower throughput for single-precision floats compared to

Table 2. Run times in ms at two shadow map resolutions for the ship in Figure 10 and the cloud in Figure 9. We list the cost of rendering the shadow map, filtering it and rendering the shaded scene using moment-based OIT with six power moments in 112 bits for primary visibility. Additionally, we provide a differential frame time where the cost for rendering without shadows of transparent surfaces has been subtracted (3.9 ms for the ship and 18.1 ms for the cloud). We also list the required additional memory in bits per shadow map texel.

Shadow technique		Fourier opacity mapping				Moment-based OIT shadows (ours)			
Scene		Ship		Cloud		Ship		Cloud	
Coefficients		7	15	7	15	1+4	1+6	1+4	1+6
Memory		112	240	112	240	160	224	160	224
1024^2	Render	0.42	0.46	1.6	3.1	0.80	0.83	2.0	2.8
	Filter	0.07	0.16	0.08	0.15	0.15	0.18	0.16	0.21
	Shade	1.8	1.9	8.6	9.5	1.9	2.3	9.1	11.4
	Diff.	1.0	1.3	4.7	7.4	1.5	2.0	5.8	8.8
2048^2	Render	0.54	0.71	6.4	11.4	0.90	1.0	9.6	13.1
	Filter	0.28	0.53	0.30	0.55	0.56	0.69	0.62	0.80
	Shade	1.8	1.9	8.7	11.3	1.9	2.2	9.2	11.9
	Diff.	1.4	1.9	9.4	17.9	2.0	2.7	13.7	20.0

half-precision floats on our test hardware. The cost for shading tends to be slightly higher with our moment-based approach, which is to be expected given the greater arithmetic workload. However, the quality improvements observed above justify this greater cost.

8 CONCLUSIONS

Moment-based OIT complements existing OIT techniques in an interesting and useful way. Like weighted blended OIT, it is fully order-independent and provides smooth results when the opacities are smooth. However, it uses actual data to represent the per-pixel transmittance, thus providing a far more accurate result. Its ability to deal with complicated transparent models and participating media simultaneously is unmatched by earlier techniques. At the same time, it can compete with these techniques in terms of run time. In times where the compute capability of GPUs outruns their bandwidth [Olsson et al. 2015], the greater number of arithmetic instructions is a small concern.

With six possible choices for the moments and two quantization schemes for each of them, there is a host of techniques to choose from. While four power moments provide an unobjectionable and inexpensive approximation in moderately challenging cases, three trigonometric moments stored in single precision handle even the most challenging cases faithfully (see Figure 1). Six power moments offer a good tradeoff between quality and cost.

The guaranteed smoothness of the results stems from the continuous representation of the transmittance. This property is very desirable but harms our ability to render intersecting geometry and might pose a problem for scenes with extremely large depth range. The logarithmically warped depth is an effective countermeasure for the latter issue.

Our new techniques for shadows surpass the quality of Fourier opacity mapping but at increased cost. Future work may push for further optimizations.

ACKNOWLEDGMENTS

We would like to thank Michael Weinmann for his feedback and for help with the evaluation. The ship model is kindly provided by Greg Zaal and Chris Kuhn. The teapots scene uses the Atrium Sponza Palace modeled by Marko Dabrovic and improved by Frank Meinl.

REFERENCES

Thomas Annen, Tom Mertens, Philippe Bekaert, Hans-Peter Seidel, and Jan Kautz. 2007. Convolution Shadow Maps. In *EGSR07: 18th Eurographics Symposium on Rendering*. Eurographics Association, 51–60. https://doi.org/10.2312/EGWR/EGSR07/051-060

Louis Bavoil, Steven P. Callahan, Aaron Lefohn, João L. D. Comba, and Cláudio T. Silva. 2007. Multi-fragment Effects on the GPU Using the K-buffer. In *Proceedings of the 2007 Symposium on Interactive 3D Graphics and Games (I3D '07)*. ACM, New York, NY, USA, 97–104. https://doi.org/10.1145/1230100.1230117

James F. Blinn. 2007. How to Solve a Cubic Equation, Part 5: Back to Numerics. *IEEE Computer Graphics and Applications* 27, 3 (May 2007), 78–89. https://doi.org/10.1109/MCG.2007.60

Loren Carpenter. 1984. The A -buffer, an Antialiased Hidden Surface Method. *SIGGRAPH Comput. Graph.* 18, 3 (Jan. 1984), 103–108. https://doi.org/10.1145/964965.808585

Cyril Delalandre, Pascal Gautron, Jean-Eudes Marvie, and Guillaume François. 2011. Transmittance Function Mapping. In *Symposium on Interactive 3D Graphics and Games (I3D '11)*. ACM, 31–38. https://doi.org/10.1145/1944745.1944751

Eric Enderton, Erik Sintorn, Peter Shirley, and David Luebke. 2011. Stochastic Transparency. *IEEE Transactions on Visualization and Computer Graphics* 17, 8 (Aug 2011), 1036–1047. https://doi.org/10.1109/TVCG.2010.123

Cass Everitt. 2001. Interactive order-independent transparency. (2001). http://www.nvidia.com/object/Interactive_Order_Transparency.html NVIDIA whitepaper.

Don Herbison-Evans. 1995. Solving quartics and cubics for graphics. In *Graphics Gems V*, Alan W. Paeth (Ed.). Academic Press, Inc., Chapter I.1, 3–15.

Jon Jansen and Louis Bavoil. 2010. Fourier Opacity Mapping. In *Proceedings of the 2010 ACM SIGGRAPH Symposium on Interactive 3D Graphics and Games (i3D '10)*. ACM, 165–172. https://doi.org/10.1145/1730804.1730831

Mark Grigorievich Kreĭn and Adol'f Abramovich Nudel'man. 1977. *The Markov Moment Problem and Extremal Problems*. Translations of Mathematical Monographs, Vol. 50. American Mathematical Society.

Tom Lokovic and Eric Veach. 2000. Deep Shadow Maps. In *Proceedings of the 27th Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH '00)*. ACM Press/Addison-Wesley Publishing Co., New York, NY, USA, 385–392. https://doi.org/10.1145/344779.344958

Marilena Maule, João Comba, Rafael Torchelsen, and Rui Bastos. 2013. Hybrid Transparency. In *Proceedings of the ACM SIGGRAPH Symposium on Interactive 3D Graphics and Games (I3D '13)*. ACM, New York, NY, USA, 103–118. https://doi.org/10.1145/2448196.2448212

Morgan McGuire and Louis Bavoil. 2013. Weighted Blended Order-Independent Transparency. *Journal of Computer Graphics Techniques (JCGT)* 2, 2 (18 December 2013), 122–141. http://jcgt.org/published/0002/02/09/

Morgan McGuire and Eric Enderton. 2011. Colored Stochastic Shadow Maps. In *Symposium on Interactive 3D Graphics and Games (I3D '11)*. ACM, New York, NY, USA, 89–96. https://doi.org/10.1145/1944745.1944760

Morgan McGuire and Michael Mara. 2017. Phenomenological Transparency. *IEEE Transactions on Visualization and Computer Graphics* 23, 5 (May 2017), 1465–1478. https://doi.org/10.1109/TVCG.2017.2656082

Houman Meshkin. 2007. Sort-independent alpha blending. (2007). GDC Session.

Stefan Neumark. 1965. Chapter 3 - Quartic Equation. In *Solution of Cubic and Quartic Equations*. Pergamon Press, 12–24. https://doi.org/10.1016/B978-0-08-011220-6.50006-8

Ola Olsson, Emil Persson, and Markus Billeter. 2015. Real-time Many-light Management and Shadows with Clustered Shading. In *ACM SIGGRAPH 2015 Courses (SIGGRAPH '15)*. ACM, Article 12, 398 pages. https://doi.org/10.1145/2776880.2792712

Christoph Peters. 2017. Non-linearly Quantized Moment Shadow Maps. In *Proceedings of High Performance Graphics (HPG '17)*. ACM, New York, NY, USA, Article 15, 11 pages. https://doi.org/10.1145/3105762.3105775

Christoph Peters and Reinhard Klein. 2015. Moment Shadow Mapping. In *Proceedings of the 19th ACM SIGGRAPH Symposium on Interactive 3D Graphics and Games (i3D '15)*. ACM, 7–14. https://doi.org/10.1145/2699276.2699277

Christoph Peters, Cedrick Münstermann, Nico Wetzstein, and Reinhard Klein. 2017. Improved Moment Shadow Maps for Translucent Occluders, Soft Shadows and Single Scattering. *Journal of Computer Graphics Techniques (JCGT)* 6, 1 (2017), 17–67. http://jcgt.org/published/0006/01/03/

Thomas Porter and Tom Duff. 1984. Compositing Digital Images. *SIGGRAPH Comput. Graph.* 18, 3 (Jan. 1984), 253–259. https://doi.org/10.1145/964965.808606

Marco Salvi, Jefferson Montgomery, and Aaron Lefohn. 2011. Adaptive Transparency. In *Proceedings of the ACM SIGGRAPH Symposium on High Performance Graphics (HPG '11)*. ACM, New York, NY, USA, 119–126. https://doi.org/10.1145/2018323.

2018342

Marco Salvi and Karthik Vaidyanathan. 2014. Multi-layer Alpha Blending. In *Proceedings of the 18th Meeting of the ACM SIGGRAPH Symposium on Interactive 3D Graphics and Games (I3D '14)*. ACM, New York, NY, USA, 151–158. https://doi.org/10.1145/2556700.2556705

Marco Salvi, Kiril Vidimče, Andrew Lauritzen, and Aaron Lefohn. 2010. Adaptive Volumetric Shadow Maps. *Computer Graphics Forum* 29, 4 (2010), 1289–1296. https://doi.org/10.1111/j.1467-8659.2010.01724.x

Chris Wyman and Morgan McGuire. 2017. Improved Alpha Testing Using Hashed Sampling. *IEEE Transactions on Visualization and Computer Graphics* PP, 99 (2017), 1–1. https://doi.org/10.1109/TVCG.2017.2739149

Jason C. Yang, Justin Hensley, Holger Grün, and Nicolas Thibieroz. 2010. Real-Time Concurrent Linked List Construction on the GPU. *Computer Graphics Forum* 29, 4 (2010), 1297–1304. https://doi.org/10.1111/j.1467-8659.2010.01725.x

Received December 2017; revised March 2018; accepted April 2018