Moment Transparency

Brian Sharpe
Weta Digital
Wellington, New Zealand
bsharpe@wetafx.co.nz

(a) weighted-blended OIT **(b)** moment transparency (MT) **(c)** MT with two moments passes **(d)** ground truth

Figure 1: Shown is a scene with a dense pre-lit particle system, and semi-transparent grass and bottles in the foreground. Weighted-blended order-independent transparency (a) gives an overly transparent look, whereas our new moments-based transparency techniques (b and c) compare well to ground truth (d).

ABSTRACT

We introduce moment transparency, a new solution to real-time order-independent transparency. It expands upon existing *approximate transmittance function* techniques by using moments to capture and reconstruct the transmittance function. Because the moment-based transmittance function can be processed analytically using standard hardware blend operations, it is efficient and overcomes limitations of previous techniques.

CCS CONCEPTS

• **Computing methodologies** → **Rasterization**;

KEYWORDS

order-independent transparency, visibility determination, real-time rendering

ACM Reference Format:
Brian Sharpe. 2018. Moment Transparency. In *HPG '18: High-Performance Graphics, August 10–12, 2018, Vancouver, Canada.* ACM, New York, NY, USA, 4 pages. https://doi.org/10.1145/3231578.3231585

HPG '18, August 10–12, 2018, Vancouver, Canada

ACM ISBN 978-1-4503-5896-5/18/08…$15.00
https://doi.org/10.1145/3231578.3231585

1 INTRODUCTION

The handling of transparency is ubiquitous in modern real-time rasterizers. The common way to handle it is using *alpha blending* which requires fragments to be processed in either front-to-back (FTB) or back-to-front (BTF) order. Ideally sorting should be on a per-pixel basis, but this can require potentially unbounded memory. In most cases sorting is handled on a per-object basis which can lead to visual and temporal artefacts.

Moment transparency addresses these problems by using *moments* (as adapted from moment shadow mapping [Peters and Klein 2015]) to construct an approximate per-pixel transmittance function, which it uses to achieve order-independent transparency. Variants on four moments are used for performance reasons, and transmittance is represented in logarithmic form (i.e. *optical depth*) allowing for additive construction, which works naturally with standard GPU blending hardware.

The resulting algorithm is efficient, has strict order independence, works within a small and fixed amount of memory, does not need specialized hardware such as *rasterizer ordered views*, and gives results comparable to ground truth.

2 BACKGROUND AND RELATED WORK

We first define a transparent fragment F_i by its distance to the viewer z_i, its alpha α_i, and its color c_i. We then define a transmittance function $T(z)$ that represents the total transmittance (i.e. visibility) between any depth z and the viewer. Using the transmittance function we can calculate the contribution of many fragments onto the final pixel color, in an order-independent manner

$$\sum_{i=1}^{n} c_i \alpha_i T(z_i). \tag{1}$$

Approximate transmittance function methods [Wyman 2016] precalculate an approximate form of the transmittance function $T(z)$, within a fixed amount of memory, and query it during rasterization to achieve order-independent transparency (OIT).

The current state-of-the-art approximate transmittance function methods have a number of shortcomings which limit their applicability. Adaptive transparency [Salvi et al. 2011] requires specialized hardware (*rasterizer ordered views*) and has temporal artefacts [Salvi and Vaidyanathan 2014]; weighted blended order-independent transparency (WBOIT) [McGuire and Bavoil 2013] has a scene-wide empirical approximation, meaning it suits mostly homogeneous data; K-buffer techniques such as hybrid transparency [Maule et al. 2013] assume the most important information resides in the frontmost k layers, which is not always the case (e.g. particle systems); and Fourier opacity map transparency [Jansen and Bavoil 2010] fails with thin surfaces, given the trigonometric basis functions lack fine-grained precision [Salvi et al. 2011].

Moment shadow mapping (MSM) [Peters and Klein 2015] is a shadowing technique that uses a depth representation based on 4 moments to achieve filtered shadows. In later works, Peters et al. derive sampling functions for 6 moments [Peters et al. 2017], provide a way to store 4 moments which is more compact and efficient to sample [Peters 2017], and show how moments can be used to store transmittance; but construction remains order-dependent [Peters et al. 2017].

Moment transparency (MT) builds upon the foundation provided by WBOIT, replacing the empirically determined transmittance function with a 4-moment version adapted from MSM.

Concurrent to our work, [Münstermann et al. 2018] recently introduced moment-based order-independent transparency (MBOIT) which also uses moments to approximate a per-pixel transmittance function for order-independent transparency. Derivations based on 4, 6, and 8 *power-moments* and 2, 3, and 4 *trigonometric-moments* are presented, each with increasing degrees of quality and cost. MBOIT shows how low-precision buffers can be used to improve performance and reduce memory, but at the expense of requiring *rasterizer ordered views* compatible hardware and a potentially scene-dependent bias. MBOIT also explores volumetric shadows.

We compare MT to WBOIT, and a ground-truth reference.

3 ALGORITHM OVERVIEW

To define moment transparency (MT) we first adapt the transmittance problem to one which is compatible with moments and OIT (Section 3.1). We then integrate the new moment-based transmittance function into WBOIT to achieve OIT (Section 3.2). To finish we present a way to warp any depth z to the unit range needed for precision purposes (Section 3.3).

3.1 Moments for Order-Independent Transmittance

Before adapting the transmittance problem to moments and OIT, we briefly outline how moments are used for shadowing in MSM.

For each pixel in a shadowmap, MSM stores the corresponding *4 moments* $(z, z^2, z^3, z^4)^\top$ for a single depth $z \in [0, 1]$ [1]. A filtered sample over a set of these pixels with corresponding weights w (assuming weights add to 1) is defined as

$$b = \sum_{i=1}^{n} (z_i, z_i^2, z_i^3, z_i^4)^\top \cdot w_i. \tag{2}$$

From b the MSM *Hamburger* function is used to calculate an approximate fraction of depth samples which are closer to the light than a given input depth z, achieving filtered shadows. Hamburger is formulated to give a *lower-bound* of the distribution, to keep self-shadowing artefacts to a minimum.

To adapt transmittance for use with moments and OIT, we move to log space, which allows alpha composition to be treated as an additive process. It is made easier using terminology from volumetric rendering. We can interpret a fragment with alpha $\alpha \in [0, 1)$ as a volume with *optical depth* d

$$d = -\log(1 - \alpha). \tag{3}$$

Optical depth is additive, meaning composition of multiple fragments is a sum

$$d' = \sum_{i=1}^{n} d_i. \tag{4}$$

Using optical depth d_i as weight w_i, Equations 2 and 3 can be used to construct a moment sample b from any number of transparent fragments. Noting that *total optical depth* d' from Equation 4 is required for normalization, we now have a compressed representation of our transmittance function made up of b and d' (i.e. 5 floats in total), that can be constructed in an order-independent manner.

By using Hamburger and inverting Equation 3 we can evaluate the moments-based transmittance function for any input depth z

$$T(z) = \exp(-\text{Hamburger}(\frac{b}{d'}, z) \cdot d'). \tag{5}$$

Results of this reconstruction can be seen in Figure 2. Note that for two samples the reconstruction is exact, but for anything more it becomes approximate.

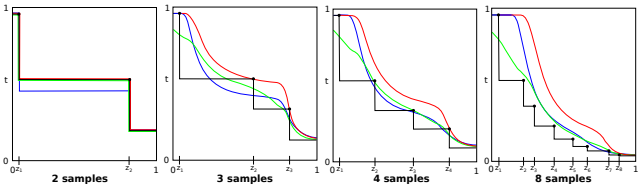

Figure 2: A transmittance function is shown (in black). Moment transparency (red) is a good match for 2 and 3 samples only. Overestimation of 25% (green) is an improvement, although it overshadows in the front. Two moments passes (blue) overshadows with 2 and 3 samples, but is an improvement for anything higher.

3.2 Moment Transparency

To extend WBOIT we replace its empirically defined $T(z)$ with the moments-based version defined by Equation 5, and call the new algorithm *Moment Transparency*. It has the following render stages:

(1) Render opaque surfaces to primary framebuffer.
   - depthwrite enabled

[1] In later works Peters et al. show that $z \in [-1, 1]$ is more optimal [Peters et al. 2017]

(2) Render transparent surfaces to an off-screen framebuffer, capturing only moments and total optical depth.
   - shares depth texture with primary framebuffer (depthtest enabled, depthwrite disabled)
   - Equations 2 and 3 used to capture moments
   - Equation 4 used to capture total optical depth
(3) Render transparent surfaces to a second off-screen framebuffer.
   - shares depth texture with primary framebuffer (depthtest enabled, depthwrite disabled)
   - Equation 1 used for composition
   - Equation 5 used for $T(z)$ while sourcing data produced from Stage 2
   - accumulate coverage in parallel
(4) Composite second off-screen transparency buffer to primary buffer while performing normalization (as based off accumulated coverage from Stage 3).

Stages 1 and 4 are unchanged from WBOIT, so we will discuss only Stages 2 and 3.

To handle Stage 2 we allocate at render resolution a FP32_RGBA framebuffer for the moments, and a FP32_R framebuffer for total optical depth. Both need to be initialized to zero and blend mode set to additive. Stage 2 is executed by rasterizing all transparent geometry to the screen, using Equations 2 and 3 to capture the moments, and Equation 4 to capture total optical depth.

For Stage 3 we replace the standard WBOIT weight function $w(z, \alpha)$ with our moments-based version, as defined by Equation 5.

With none of the equations being reliant on fragment order, moment transparency is strictly order-independent.

Further details are given in the supplementary material.

3.3 Converting Z to Unit Depth

As stated in Section 3.1 we require our depth range to be $z \in [0, 1]$. This is to help with precision issues, especially when accumulating moments using Equation 2. To convert any depth z to unit range, given near plane Z_{near} and far plane Z_{far}, we use a logarithmic warp which helps give equal relative precision at all scales.

$$z' = \frac{\log(z/Z_{near})}{\log(Z_{far}/Z_{near})}. \tag{6}$$

4 IMPROVEMENTS AND DISCUSSION

We now outline optional improvements to moment transparency, and finish with results and discussion.

4.1 Low Resolution Moments

As shown in Section 3.1 moments can be filtered. This allows us to decouple the resolution of the moment buffers (i.e. the transmittance function) from that of the image. Constructing moments at low resolution increases performance and reduces memory. It requires the following changes to moment transparency:

- Reduced resolution for moment and total optical depth buffers.
- Downsample of the primary framebuffer's depth texture after Stage 1 (using max for the filter), for use in Stage 2.
- Implementation of texture sampling in $T(z)$ changed to account for low resolution moment buffers (Stage 3).

(a) MT with low resolution moments **(b)** Moment transparency (left) and two-moments-passes (right)

Figure 3: Low resolution moments (a) offer a significant gain in performance, at the expense of some rendering artefacts (e.g. grass blades disappearing against the fire). Semi-transparent grass seen through smoke (b); both moment transparency and two moments passes can have issues in this case.

Our experiments have found a reduction to 1/8 of render resolution to be a good trade-off between memory and efficiency versus quality.

Results of this can be seen in Figure 3a. Thin surfaces can have significant issues, such as the grass blades disappearing against the fire, as well as a loss of contrast with the grass in general. Large surfaces tend to cope better, with little to no issues observed on the fire and smoke, or glass bottles.

4.2 Different Formats for Write and Read

To improve performance, it is possible to use a different format for moments capture (i.e. write) than used for moments reconstruction (i.e. read).

If moments-resolution is the same as render-resolution, then non-linearly quantized moments (NLQM) [Peters 2017] can be used for reconstruction, saving both arithmetic instructions and bandwidth. This can be integrated into moment transparency by converting the moments data produced by Stage 2 to NLQM before it is consumed by Stage 3. NLQM compress from 32-bits to 16-bits with no observable loss in quality.

NLQM cannot be filtered, so are incompatible if moments are at low resolution. An alternative for this case is to convert to a standard 16-bit moment representation [Peters and Klein 2015], as the reduced bandwidth can give a slight performance improvement.

Implementation details are given in the supplementary material.

4.3 Overestimation

As can be seen in Figure 2, the quality of the 4-moments transmittance function degrades as the number of fragments increases.

The problem comes from representing the transmittance function as optical depth (instead of the more optimal form described in [Peters et al. 2017]), which gives equal importance to fragments at the back of the list as to those in the front, even though they contribute less to the final pixel. This results in poor compression.

Overestimation [Peters et al. 2017] can help, but it can also result in over-shadowing of the foreground layers, and sometimes needs

to be tweaked to get the best look. We've used overestimation of 25% in Figures 1b and 3a.

4.4 Two Moments Passes

Another alternative for improving compression is to use two moments passes, which attempts to account for the occlusion the front fragments have on those behind. To do this, we can construct a second set of moments (essentially repeating Stage 2), using the first set of moments as an estimate for this occlusion. This adds bias in that each fragment will be occluded by twice the amount it should be. In practice we find the bias is offset by the loss of detail in the moments reconstruction, which tends to under occlude.

The additional rasterization pass can be expensive at full resolution, but if low resolution moments are used the cost can be acceptable.

Results of this second moments pass can be seen in Figures 1c and 2. In general the results are positive, with foreground objects appearing more clear, and volumetric effects improved. But in some cases the results are not as good as overestimation (Figure 3b).

Implementation details can be found in the supplementary material.

4.5 Performance and Memory

We now outline memory and performance metrics for the techniques discussed. Memory is in bytes-per-pixel. Performance is time (in milliseconds) taken to render the *burning house* scene from Figures 1 and 3a, and the *grass and smoke* scene from Figure 3b [2]. Times were measured on an NVIDIA GeForce GTX 1070.

Table 1: Memory requirements (bytes per pixel) and render times (milliseconds) for some of the techniques presented.

		burning house			grass and smoke		
Technique	Memory	Stage2	Stage3	Total	Stage2	Stage3	Total
WBOIT	12	0	7.7	7.8	0	11.1	11.2
MT	32	22.2	9.1	31.4	36.5	13.4	50.0
MT_NLQM	42	22.2	9.0	31.3	36.5	13.2	49.8
TMP	52	53.0	9.3	62.4	88.0	14.2	102.3
MT_LR	12.3125	1.4	9.1	10.6	1.1	13.4	14.6
TMP_LR	12.625	2.9	9.3	12.3	2.5	14.2	16.8

Moment transparency (MT) and moment transparency with nonlinear quantized moments (MT_NLQM) have a very similar time, which is curious to us. It perhaps indicates moment reconstruction cost (and bandwidth) for these scenes is hidden by general shader cost of the assets. Two moments passes (TMP) is slow at full resolution, but the two low resolution variants of MT and TMP give very good performance and memory cost, indicating a good option if the associated artefacts can be tolerated.

4.6 Visual Analysis

As seen in Figure 1, MT is a good improvement over WBOIT, although the foreground grass blades and bottle label appear slightly more transparent than they should, and the fireball still differs from ground truth (GT). Two moments passes (Figure 1c) improves even more, with foreground grass blades and bottle label appearing opaque, and the fireball appearing more similar to GT.

The *grass and smoke* scene from Figure 3b has the camera positioned within a dense smoke particle system, looking towards some semi-transparent grass. This presents a difficult case for all techniques discussed, with none producing a satisfactory result.

[2] Stage 2 is not applicable to WBOIT, and two moments passes (TMP) includes times for both moments-capture passes as Stage 2.

5 CONCLUSION AND FUTURE WORK

We have introduced moment transparency (MT), which constructs a transmittance function, based on four moments, to achieve approximate OIT. Transmittance is represented using optical depth which allows for additive construction. Because the moments-based transmittance function can be processed analytically using standard hardware blend operations, it is efficient and overcomes limitations of previous techniques.

Some options for improvement were presented, each with tradeoffs between quality, performance, and robustness.

In the future we plan to improve performance and accuracy. One idea is to further explore the use of different formats for write and read; another is to use *adaptive overestimation* [Peters et al. 2017].

ACKNOWLEDGMENTS

We thank Marc Droske for help throughout the authoring process. We also thank Alan Chambers, Alex Best, Andrea Weidlich, Andrew Sidwell, Andy Styles, Deanna Louie, Joe Letteri, Johannes Meng, Jon Hertzig, Lauchlan Robertson, Luca Fascione, Paolo Selva, Peter McGrattan, Robert Cannell, Sam Tack, and all others at Weta Digital.

REFERENCES

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, New York, NY, USA, 165–172. https://doi.org/10.1145/1730804.1730831

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/

Cedrick Münstermann, Stefan Krumpen, Reinhard Klein, and Christoph Peters. 2018. Moment-Based Order-Independent Transparency. *Proceedings of the ACM on Computer Graphics and Interactive Techniques* 1, 1 (May 2018), 7:1–7:20. https://doi.org/10.1145/3203206

Christoph Peters. 2017. Non-Linearly Quantized Moment Shadow Maps. In *Proceedings of the 9th Conference on High-Performance Graphics (HPG '17)*. ACM. https://doi.org/10.1145/3105762.3105775

Christoph Peters and Reinhard Klein. 2015. Moment Shadow Mapping. In *Proceedings of the 19th Meeting of the ACM SIGGRAPH Symposium on Interactive 3D Graphics and Games (i3D '15)*. ACM, New York, NY, USA, 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 (30 March 2017), 17–67. http://jcgt.org/published/0006/01/03/

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

Chris Wyman. 2016. Exploring and Expanding the Continuum of OIT Algorithms. In *Proceedings of High Performance Graphics (HPG '16)*. Eurographics Association, Aire-la-Ville, Switzerland, Switzerland, 1–11. http://dl.acm.org/citation.cfm?id=2977336.2977338