Moment Bounds are Differentiable: Efficiently Approximating Measures in Inverse Rendering

MARKUS WORCHEL, Technische Universität Berlin (TU Berlin), Germany
MARC ALEXA, Technische Universität Berlin (TU Berlin), Germany

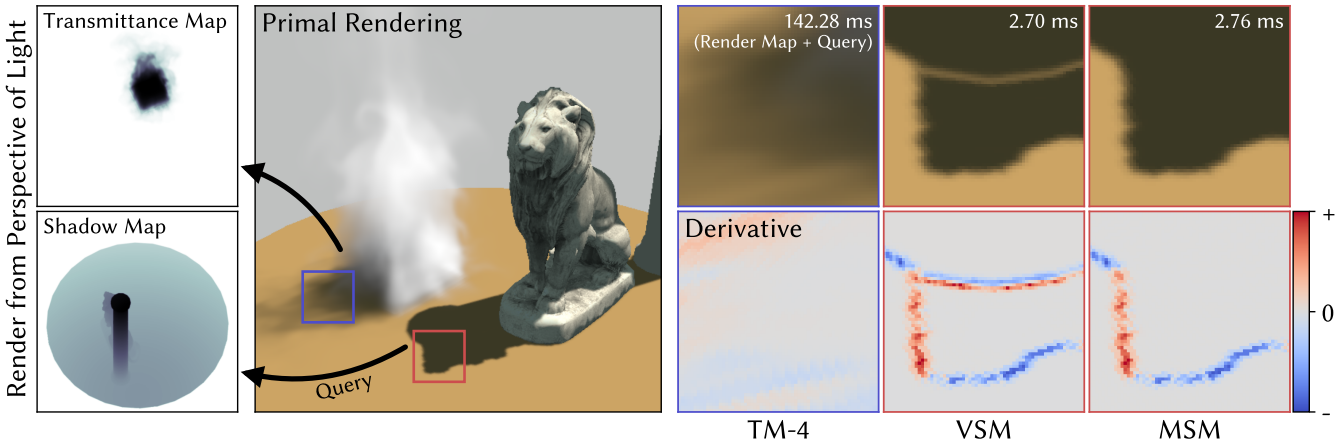


Fig. 1. We prove that moment-based bounds are differentiable. This enables various applications of approximate methods from real-time graphics in differentiable rendering: we show differentiable transmittance mapping ("TM-*n*") akin to volumetric shadow mapping and differentiable shadow mapping for triangle meshes. The suggested moment-based techniques (e.g. **M**oment **S**hadow **M**apping), are more principled, more accurate, and similarly efficient as existing approaches (e.g. **V**ariance **S**hadow **M**apping). The (forward) derivatives show the effect of moving the directional light source.

All rendering methods aim at striking a balance between realism and efficiency. This is particularly relevant for *differentiable* rendering, where the additional aspect of differentiablity w.r.t. scene parameters causes increased computational complexity while, on the other hand, in the common application of inverse rendering, the diverse effects of real image formation must be faithfully reproduced. An important effect in rendering is the attenuation of light as it travels through different media (visibility, shadows, transmittance, transparency). This can be modeled as an integral over non-negative functions and has been successfully approximated in forward rendering by so-called moments. We show that moment-based approximations are differentiable in the parameters defining the moments, and that this leads to efficient and practical methods for inverse rendering. In particular, we demonstrate the method at the examples of shadow mapping and visibility in volume rendering, leading to approximations that are similar in efficiency to existing ad-hoc techniques while being significantly more accurate.

CCS Concepts: • **Computing methodologies** → **Rendering**; • **Mathematics of computing** → **Mathematical analysis**.

Additional Key Words and Phrases: differentiable rendering, shadows

Authors' addresses: Markus Worchel, Technische Universität Berlin (TU Berlin), Berlin, Germany, m.worchel@tu-berlin.de; Marc Alexa, Technische Universität Berlin (TU Berlin), Berlin, Germany, marc.alexa@tu-berlin.de.

0730-0301/2025/8-ART
https://doi.org/10.1145/3730899

ACM Reference Format:
Markus Worchel and Marc Alexa. 2025. Moment Bounds are Differentiable: Efficiently Approximating Measures in Inverse Rendering. *ACM Trans. Graph.* 44, 4 (August 2025), 21 pages. https://doi.org/10.1145/3730899

1 INTRODUCTION

Differentiable renderers are an integral part of solutions to inverse rendering problems – problems in which information about a scene is recovered solely from images. If the observations are captured from the real world, one strives for a differentiable renderer that is capable of reproducing the various effects of physical light transport. As in forward rendering, the accuracy of these sophisticated rendering systems comes at the price of efficiency.

The complete picture shows that most applications require a differentiable renderer that strikes a good balance between accuracy *and* efficiency. Countless approximations from real-time graphics have been adopted in the inverse setting and form the foundation of practical differentiable rendering systems in 3D vision, graphics, and machine learning.

Many of these techniques are trivially differentiable and can be readily transferred, such as approximations for image-based lighting [Munkberg et al. 2022], shading models [Gao et al. 2022; Zhang et al. 2021], or texture mapping [Laine et al. 2020]. For other techniques, this is less obvious and naturally leads to the question: *which* rendering approximations *are* differentiable?

In this work, we seek an answer to the question for a family of approximations that bound *measures*, which, in the simplest case, can be understood as non-negative, non-decreasing functions. Various

problems in graphics can be cast in this framework, most notably, the visibility between a light and a point in the scene, which, arguably, is the most important information for the effect of the light source on the rendered image. Ignoring the visibility in inverse settings is acceptable only in very restrictive setups (e.g. with co-located camera and light [Luan et al. 2021; Zhang et al. 2022]).

Bounds to measures have previously been studied for approximate rendering using the *theory of moments* [Münstermann et al. 2018; Peters and Klein 2015; Peters et al. 2019] (Section 3).

Computing moment-based approximations, e.g. for visibility, is, compared to other approximate rendering techniques, quite involved such that differentiability is not apparent. Additionally, the representation has singularities, which could not only represent discontinuity points but evaluating the bound in their vicinity is numerically challenging. We prove that the bounds to measures derived from power moments are continuously differentiable (Section 4).

The study of differentiability reveals the behavior of the bound and its derivatives near singularities, which leads to an efficient and robust implementation of differentiable moment bounds (Section 5).

The proof justifies using various moment-based approximations in differentiable rendering (Section 6): for differentiable shadow mapping (Section 6.1), we use Moment Shadow Mapping [Peters and Klein 2015], which generalizes existing differentiable shadow mapping techniques [Donnelly and Lauritzen 2006; Worchel and Alexa 2023], and show that it is an efficient and more accurate approach. For differentiable visibility in volume rendering (Section 6.2), we use classical work on moment-based transmittance estimation [Münstermann et al. 2018; Peters et al. 2016] and show that even the lowest quality approximation is as fast as existing ad-hoc approximations while being significantly more accurate.

We conclude with limitations and future avenues (Section 7).

2 RELATED WORK

The techniques presented in our applications (Section 6) are related to approaches in vision and graphics, and contribute to the current state of the field. We, however, delay the discussion to the appropriate sections and take a more general perspective here.

Moment-based Bounds in Graphics. The study of moments dates back at least to the 19th century. For brevity, we will limit our discussion to moment-based bounds with a focus on graphics.

Moment-based approximations were introduced to graphics by Donelly and Lauritzen [2006] for real-time shadow mapping, with extensions explored by Salvi [2008] and introduced in the general form by Peters and Klein [2015] with Moment Shadow Mapping. Their main algorithm computes bounds from four power moments and is derived from the conditions in the classical literature [Akhiezer 1965; Akhiezer and Kreĭn 1962; Kreĭn and Nudel'man 1977].

Later, moment-based bounds have been used for different approximations in graphics, including estimating volume transmittance for shadows [Peters et al. 2016] or for order-independent transparency [Münstermann et al. 2018; Sharpe 2018].

Our work extends this line to the differentiable setting: to the best of our knowledge, we are the first to show that the lower and upper bounds, as stated by Tari [2005] for the infinite case, are continuously differentiable functions in all parameters. Our proof is a generalization of the result by Worchel and Alexa [2023], who consider the special case of Variance Shadow Mapping [Donnelly and Lauritzen 2006]. It follows that many moment-based approximations can be directly combined with differentiable rendering.

Differentiable Rendering and Approximations. Physically-based differentiable rendering simulates light transport [Jakob et al. 2022b; Li et al. 2018; Nimier-David et al. 2019; Zhang et al. 2020], which, despite significant advances in efficiency [Jakob et al. 2022a; Nimier-David et al. 2020; Vicini et al. 2021], still does not match the efficiency of differentiable rasterization [Laine et al. 2020; Liu et al. 2019].

In practice, differentiable rendering is often combined with approximations, for illumination [Hasselgren et al. 2022; Munkberg et al. 2022], materials [Gao et al. 2022; Zhang et al. 2021], visibility [Yang et al. 2022], or even by modeling light transport entirely with neural networks [Worchel et al. 2022; Yariv et al. 2020].

Our work contributes to approximate differentiable rendering in two ways. First, by proving the differentiability of moment-based bounds, which underly a family of approximate rendering techniques, we justify their usage in the differentiable setting. Second, as an immediate consequence, we show that these techniques improve over previous methods in two applications, differentiable shadow mapping and differentiable transmittance approximation.

Forward rendering systems have been transformed to differentiable variants using differentiable shading languages [Bangaru et al. 2023] or stochastic gradient estimates [Deliot et al. 2024]. The burden of investigating approximate rendering techniques for differentiability, however, remains because applying automatic differentiation to a function does not necessarily lead to useful gradients: the prime example in differentiable rendering is discontinuous visibility.

3 BOUNDING MEASURES WITH MOMENTS

Given a non-negative function $f(z)$ on $\mathbb{R}$ with $\int_{-\infty}^{\infty} f(z)\ \mathrm{d}z > 0$. We are interested in the integral

$$F(x) = \int_{-\infty}^{x} f(z)\ \mathrm{d}z, \tag{1}$$

which is non-negative and non-decreasing. For example, we may encounter such integrals in volume rendering, as part of the transmittance, where f is the attenuation coefficient along a ray

$$T(x) = \exp\big(-F(x)\big) = \exp\Big(-\int_{0}^{x} f(z)\ \mathrm{d}z\Big), \tag{2}$$

and the lower limit is adjusted to 0, assuming zero attenuation in the negative ray direction. Now, consider the following task: from f derive a compact representation, i.e., a small set of scalar values, from which the measure $F(x)$ can be approximated. A practical motivation could be efficiency, if the transmittance is repeatedly evaluated at different points, given that the approximation can be evaluated faster than the integral.

This task may be approached using different classical techniques, including the representation of F or f in a (polynomial) function space. For many techniques, it is unclear how the essential properties of f (non-negative) or F (non-decreasing) could be preserved: transmittance ranges from 0 to 1, but approximations could yield values beyond 1 (approximation of F is negative) or *decrease* along

the ray. This causes (visual) artifacts in the best case but generally leads to undefined behavior as assumptions about T do not hold.

A suitable tool for studying and approximating integrals of non-negative functions is the theory of moments. Consider a non-decreasing measure $\mu(x)$[1] and the Riemann-Stieltjes integral

$$m_i = \int_{-\infty}^{\infty} z^i \, \mathrm{d}\mu(z), \qquad 0 \le i \le n. \tag{3}$$

Characterizing, and in particular, recovering μ from the *power moments* m_i is known as the *moment problem*. Specifically, the instance considering a finite set of moments and the real line is known as the truncated Hamburger moment problem. The measure $\mu(x)$ corresponds to $F(x)$ from above but this form is more general, as it includes measures that do not admit a representation with a function f, such as the step function.

In general, a measure cannot be uniquely determined from a finite set of moments because different measures can generate the same moments. Yet, it is possible to generate lower and upper *bounds* for the set of measures that are representations[2] of the given moments:

$$\begin{aligned} L(x) &= \inf\left\{\mu(x) \,\middle|\, m_i = \int_{-\infty}^{\infty} z^i \, \mathrm{d}\mu(z)\right\} && \text{(Lower bound)} \\ U(x) &= \sup\left\{\mu(x) \,\middle|\, m_i = \int_{-\infty}^{\infty} z^i \, \mathrm{d}\mu(z)\right\}. && \text{(Upper bound)} \end{aligned} \tag{4}$$

Quite surprisingly, these bounds can be efficiently computed under mild assumptions on the moments [Tari et al. 2005] and in graphics they have been used in efficient methods for shadow mapping [Peters and Klein 2015; Peters et al. 2016] and order-independent transparency [Münstermann et al. 2018]. We only consider even orders $2n$, i.e., an odd number of moments $m_0, m_1, \ldots, m_{2n}$ for $n > 0$.

3.1 A Geometric Perspective and General Observations

We think it is helpful to introduce an intuitive, geometric perspective on moment-based bounds. Consider the following curve in $\mathbb{R}^{2n+1}$

$$\mathbf{u}(t) = \begin{pmatrix} 1 & t & t^2 & \ldots & t^{2n} \end{pmatrix}^\top, \qquad -\infty < t < \infty, \tag{5}$$

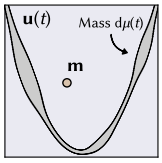
in the affine hyperplane $(1\ 0\ 0\ \ldots\ 0)^\top \mathbf{x} = 1$. Using $\mathbf{u}(t)$, the integral in Equation (3) can be re-written in vector notation as

$$\mathbf{m} = \int_{-\infty}^{\infty} \mathbf{u}(z) \, \mathrm{d}\mu(z), \tag{6}$$

so the moment sequence is a point in $\mathbb{R}^{2n+1}$

$$\mathbf{m} = \begin{pmatrix} m_0 & m_1 & \ldots & m_{2n} \end{pmatrix}^\top. \tag{7}$$

If μ is viewed as a measure of physical mass distributed along the curve, normalized such that the total mass is $\int_{-\infty}^{\infty} \mathrm{d}\mu(z) = m_0 = 1$, then the point $\mathbf{m}$ is simply the center of mass. More generally, the set of moments to all non-decreasing measures is the convex hull of the curve $\mathbf{u}(t)$ [Riesz 1911]. The inset visualizes a mass distribution on the curve of order two $\mathbf{u}(t) = \begin{pmatrix} 1 & t & t^2 \end{pmatrix}^\top$. If the requirement of normalized distributions is dropped, the set of moments is the *cone* formed by the rays from the origin through the convex hull of the curve – the *conic hull* of $\mathbf{u}$ [Kreĭn and Nudel'man 1977, Theorem 3.4, p.15].



[1]This definition of "measure" follows the conventions by Kreĭn and Nudel'man [1977].
[2]A measure μ is a "representation" of a moment vector $\mathbf{m}$ if it fulfills Equation (6).

A central observation in the theory of moments is that integrals of polynomials are exactly expressible in the moments $\mathbf{m}$ for any measure μ that is a representation of them. Given a polynomial $p(x) = \sum_{i=0}^{k} a_i x^i$, with $k \le 2n$, its integral is

$$\int_{-\infty}^{\infty} p(x) \, \mathrm{d}\mu(x) = \int_{-\infty}^{\infty} \sum_{i=0}^{k} a_i x^i \, \mathrm{d}\mu(x) = \sum_{i=0}^{k} a_i m_i. \tag{8}$$

For a moment sequence $\mathbf{m} = (m_0, \ldots, m_{2n})$ and a vector of polynomial coefficients $\mathbf{a} = (a_0, \ldots, a_{2n})$, $a_{j>k} = 0$ the operator

$$\mathfrak{S}_{\mathbf{m}}\{p\} = \mathbf{a}^\top \mathbf{m} \tag{9}$$

provides the exact integral of p for *any* μ being a representation of $\mathbf{m}$ [Kreĭn and Nudel'man 1977, p.58], and leads to a key property:

Definition 1. *A moment sequence* $\mathbf{m} = (m_0, \ldots, m_{2n})$ *is called* strictly positive *if from* $p(x) \ge 0$ *(*$p \not\equiv 0$*) it follows that* $\mathfrak{S}_{\mathbf{m}}\{p\} > 0$.

Strict positivity has an intuitive interpretation: the polynomial coefficients represent a vector $\mathbf{a} \in \mathbb{R}^{2n+1}$ and the condition $p(x) = \sum_{i=0}^{k} a_i x^i = \mathbf{a}^\top \mathbf{u}(x) \ge 0$ is fulfilled if the curve $\mathbf{u}$ lies in the closed upper half-space of the hyperplane $\mathbf{a}^\top \mathbf{x} = 0$. In other words, it is fulfilled if $\mathbf{a}$ defines a support hyperplane for the conic hull of the curve. The condition $\mathfrak{S}_{\mathbf{m}}\{p\} > 0$ is equivalent to $\mathbf{a}^\top \mathbf{m} > 0$, which demands that $\mathbf{m}$ does not lie on the support hyperplane: a strictly positive moment sequence lies *inside* the conic hull of $\mathbf{u}(t)$.

Strictly positive moments $\mathbf{m}$ are important because they can be represented as conical combinations of points on the moment curve:

Theorem 1 ([Akhiezer and Kreĭn 1962, Theorem 3a, p.8]). *If* $\mathbf{m}$ *is strictly positive, there exist infinitely many canonical representations*

$$\mathbf{m} = \sum_{i=0}^{n} w_i \mathbf{u}(x_i),$$

with $w_i > 0$ *the* weights *and* $x_i \in (-\infty, \infty)$ *the* roots *(*$x_i \ne x_j$*).*

We will later show that it is natural to identify the x_i as the zeros of a polynomial, hence the name. The importance is that any point $\mathbf{m} \in \mathbb{R}^{2n+1}$ strictly inside the conic hull can be represented by the (strictly) conical combination of $n+1$ points on the curve. Note that $n+1$ is roughly *half* the embedding dimensions of the curve. Applied to the physical example above, the theorem states that there are infinitely many ways of concentrating mass in at most $n+1$ points on the curve, which result in the same center of mass. The infinite set is a one-parameter family:

Theorem 2 ([Akhiezer and Kreĭn 1962, Theorem 3c, p.8]). *If* $\mathbf{m}$ *is strictly positive, for any value* $\eta \in \mathbb{R} \setminus A$*, there is a unique canonical representation of* $\mathbf{m}$ *where* η *is one of the* $n+1$ *roots.*

A is a small, finite set that depends on the moments and we delay the discussion to Section 4.1.1. Geometrically, given a point on the curve $\mathbf{u}(\eta)$, n unique points $\{\mathbf{u}(x_i)\}_1^n$ on the curve can be found so that their weighted combination yields $\mathbf{m}$.

The unique canonical representation is associated with a piecewise constant measure $\mu_\eta(x)$, increasing in the discrete set of the $n+1$ points, where $d\mu_\eta(x)$ consists of $n+1$ Dirac distributions placed

at η and the $\{x_i\}$:

$$\int_{-\infty}^{y} \mathbf{u}(x)\, \mathrm{d}\mu_\eta(x) = \int_{-\infty}^{y} \sum_{\substack{i=0 \\ x_i<y}}^{n} \delta(x-x_i) w_i \mathbf{u}(x)\, \mathrm{d}x = \sum_{\substack{i=0 \\ x_i<y}}^{n} w_i \mathbf{u}(x_i). \tag{10}$$

The measure μ_η is important because it defines a bound on the set of all measures consistent with a given set of moments:

Theorem 3 ([Kreĭn and Nudel'man 1977, Theorem 3.1, IV.§3, p.125], *Chebyshev-Markov inequalities*). *Let μ be any measure with the same moments as μ_η, then*

$$\int_{-\infty}^{\eta-0} d\mu(x) \geq \int_{-\infty}^{\eta-0} d\mu_\eta(x)$$
$$\int_{-\infty}^{\eta+0} d\mu(x) \leq \int_{-\infty}^{\eta+0} d\mu_\eta(x)$$

Remark 1. *The notation follows Kreĭn and Nudel'man [1977, p.15]: the first inequality considers the mass in the interval $(-\infty, \eta)$ and the second inequality the mass in the interval $(-\infty, \eta]$.*

With the discrete representation from Eq. (10), this yields an explicit expression for the bounds [Tari 2005, Corollary 2.7]:

$$\begin{aligned} \int_{-\infty}^{\eta-0} \mathrm{d}\mu(x) &\geq \sum_{\substack{i=1 \\ x_i<\eta}}^{n} w_i &&= L(\eta) \\ \int_{-\infty}^{\eta+0} \mathrm{d}\mu(x) &\leq w_0 + \sum_{\substack{i=1 \\ x_i<\eta}}^{n} w_i &&= U(\eta) \end{aligned} \tag{11}$$

Computing the bounds from Eq. (4) therefore boils down to computing the unique canonical representation consisting of the weights $\{w_i\}$ and the roots $\{x_i\}$ given η.

3.2 Computing the Unique Canonical Representation

If η is identified with x_0, the unique canonical representation is the solution of a system of $2n+1$ equations (c.f. Theorem 1)

$$m_k = \sum_{i=0}^{n} w_i x_i^k, \qquad k = 0, \ldots, 2n \tag{12}$$

in $2n+1$ unknowns: the $n+1$ weights and the n points $x_1, \ldots, x_n$ (because $\eta = x_0$ is given). Maybe surprisingly, the computation of the points and weights can be separated, and it is possible to solve for the points x_i not knowing the weights w_i. Once all x_i are given, solving for the weights is just a linear system of equations. The equations in this section are provided without proof, and details can be found in the work by Tari [2005].

The main tool is the *Hankel matrix* of the moments

$$\mathbf{H} = \begin{bmatrix} m_0 & m_1 & \ldots & m_n \\ m_1 & m_2 & & \vdots \\ \vdots & & \ddots & \\ m_n & \ldots & & m_{2n} \end{bmatrix}, \tag{13}$$

which also indicates strict positivity of $\mathbf{m}$:

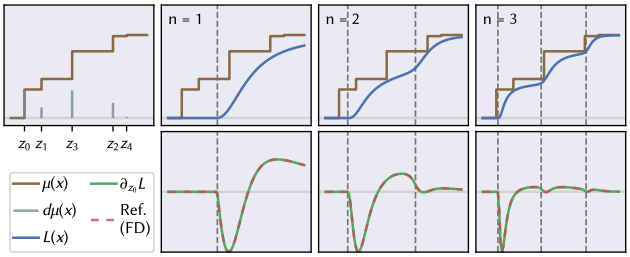


Fig. 2. A discrete measure μ with five points of increase $z_0, \ldots, z_4 \in \mathbb{R}$. Top row: the lower bound L for different degrees, corresponding to 3, 5, and 7 moments, respectively. Bottom row: the derivative of the lower bound for the point z_0, which affects the moments. The bound and its derivatives are continuous. The dashed vertical lines mark singularity points of the bound.

Theorem 4 ([Akhiezer and Kreĭn 1962, Theorem 1, p.3]). *For a moment sequence to be strictly positive, it is necessary and sufficient that the Hankel matrix is positive definite.*

With the shorthand notation

$$\mathbf{x} \triangleq \begin{pmatrix} 1 & x & \ldots & x^n \end{pmatrix}^\top, \tag{14}$$

the $\{x_i\}$ can be computed as solutions to

$$\mathbf{x}_i^\top \mathbf{H}^{-1} \mathbf{x}_0 = \mathbf{x}_i^\top \mathbf{H}^{-1} \boldsymbol{\eta} = 0, \quad i > 0. \tag{15}$$

This relates the unknown x_i to the known quantities η and $\mathbf{H}$: the points $\{x_i\}$ are the roots of the so-called *kernel polynomial*

$$K(x) = \mathbf{x}^\top \mathbf{H}^{-1} \boldsymbol{\eta} = \sum_{k=0}^{n} x^k (\mathbf{H}^{-1} \boldsymbol{\eta})_k. \tag{16}$$

Therefore, finding the n points x_i of the unique canonical representation given $x_0 = \eta$ amounts to finding the roots of the n-degree kernel polynomial. If $\mathbf{m}$ is strictly positive and $\eta \notin A$, the roots are real and simple, i.e., K has a set of n distinct roots (Corollary 19).

With all points x_i, the weights can be either computed by solving the (reduced) Vandermonde system in Eq. (12) or individually as

$$w_i = \frac{1}{\mathbf{x}_i^\top \mathbf{H}^{-1} \mathbf{x}_i}. \tag{17}$$

4 DIFFERENTIATING MOMENT-BASED BOUNDS

The approach detailed in Section 3 gives rise to two functions that, depending on an evaluation point $\eta \in \mathbb{R}$ and a sequence of moments $\mathbf{m} \in \mathbb{R}^{2n+1}$, compute an upper and a lower bound on all measures μ that are representations of the moments. When viewed as a function in all parameters, the lower bound takes the form (c.f. Eq. (11))

$$L(\eta; \mathbf{m}) = \sum_{\substack{i=1 \\ x_i(\eta;\mathbf{m})<\eta}}^{n} w_i(\eta; \mathbf{m}), \tag{18}$$

where the w_i are the weights and the x_i are the points of the unique canonical representation. The upper bound includes the weight of η

$$U(\eta; \mathbf{m}) = w_0(\eta; \mathbf{m}) + L(\eta; \mathbf{m}) \tag{19}$$

If one intends to use the lower or upper bounds in a differentiable setting, one may ask if these functions are differentiable in $\mathbf{m}$ and η. It turns out that the answer to this question is difficult to find in the

given form, in particular, because the *number of terms* in the sum in L varies depending on *both*, $\mathbf{m}$ and η.

Worchel and Alexa [2023] have shown that a special case of the lower bound is continuously differentiable, in particular, the visibility function from Variance Shadow Mapping [Donnelly and Lauritzen 2006]. This function is based on Chebyshev's inequality and is a special case of the lower bound for probability measures and $\mathbf{m} = (1, m_1, m_2)$. We generalize this result to sequences with an arbitrary number of moments (see the example in Figure 2).

The proof can be broken down into the following high-level parts:

(1) Section 4.1: we partition the space $\mathbb{R} \times \mathbb{R}^{2n+1}$ into regions where L is defined and regions where it is not, and show that L is trivially differentiable in the regions that form its domain. A key observation is that the number of terms in the bound, i.e., the number of summed weights (c.f. Eq.18) remains constant inside of the valid regions.
(2) Section 4.2: we then analyze the behavior of the bound as one approaches the boundary between two regions. The main insight is that at each transition between distinct regions only *one* weight is included or excluded from the sum in L.
(3) Section 4.3: finally, we show that not only the weight but also its derivative with respect to $\mathbf{m}$ and η vanish: the limit exists, so the singularities at region boundaries are removable, which makes L (and U) continuously differentiable in their domain and across the singular points.

Since the position of η among the other roots is relevant in the following, we assume, for this section, that all roots, including η, are ordered from lowest to highest, i.e., $x_0 < \cdots < x_j (= \eta) < \cdots < x_n$.

4.1 Partitioning the Domain

The main difficulty with the bound L and its derivative is the varying number of summed weights (c.f. Equation (18)): including a weight is a *binary* decision that can lead to discontinuities in the function or its derivative. The central idea is to first dissect the space $\mathbb{R} \times \mathbb{R}^{2n+1}$ into regions with a constant number of terms. Inside of these regions, differentiability can be easily investigated in terms of the w_i.

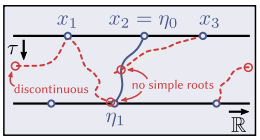
4.1.1 Regions with Constant Number of Terms. According to Theorem 1, canonical representations, in particular the unique one that defines the bound L, only exist for strictly positive moments $\mathbf{m}$. Moreover, Theorem 2 indicates that there is the additional restriction of $\eta \notin A$: A contains the n simple roots of a special degree-n polynomial $P_n(x; \mathbf{m})$. This characterizes the *domain* of L:

$$\Omega = \{(\eta, \mathbf{m}) \in \mathbb{R} \times \mathbb{R}^{2n+1} : \text{positive}(\mathbf{m}) \wedge P_n(\eta; \mathbf{m}) \neq 0\}. \quad (20)$$

At any given point $(\eta_0, \mathbf{m}_0) \in \Omega$ the bound L will be a sum over some number of terms, so, under which conditions does the number change if one moves to *another* point $(\eta_1, \mathbf{m}_1) \in \Omega$, on a continuously differentiable path? The answer is our first result:

Theorem 5. *If the number of terms in the bounding functions differs between two points $(\eta_0, \mathbf{m}_0) \in \Omega$ and $(\eta_1, \mathbf{m}_1) \in \Omega$, any continuously differentiable path connecting the points crosses at least one point $(\eta', \mathbf{m}') \in \mathbb{R} \times \mathbb{R}^{2n+1}$ where either η' is a root of $P_n(x; \mathbf{m}')$ or the sequence of moments $\mathbf{m}'$ is not strictly positive.*

The proof can be found in Appendix C, Section C.3. It exploits that the number of terms changes if and only if one of the $\{x_i\}$ changes its place with η on the real line. The points $\{x_i\}$ and η are the roots of an orthogonal polynomial $\psi_{n+1}(x; \eta, \mathbf{m})$, which has $n+1$ real and simple roots in the domain Ω. Since the roots are continuously differentiable functions in $\mathbf{m}$ and η, the $\{x_i\}$ and η can only change their order outside of the domain Ω. The inset visualizes the theorem and shows the evolution of the roots on a continuously differentiable path $\boldsymbol{\gamma}(\tau) \in \mathbb{R} \times \mathbb{R}^{2n+1}$ from a point $(\eta_0, \mathbf{m}_0) \in \Omega$ to a point $(\eta_1, \mathbf{m}_1) \in \Omega$. Red trajectories cannot be taken in Ω.



Geometrically, strict positivity can be violated on a path when moments $\mathbf{m}$ cross through $\mathbf{u}(\eta)$ on the boundary of the moment curve (more generally boundary regions on the cone) or due to numerical inaccuracy. But, strict positivity can be enforced by *biasing* [Peters and Klein 2015]: the moments $\mathbf{m}$ are slightly "pulled" inside of the conic hull, away from its boundary.

Proposition 6. *If $\alpha \in (0, 1]$ and $\mathbf{m}^\star \in \mathbb{R}^{2n+1}$ is a bias vector such that $(1-\alpha)\mathbf{m} + \alpha\mathbf{m}^\star$ is strictly positive, the number of terms in the bounding functions changes only in points where η is a root of the polynomial $P_n(x; (1-\alpha)\mathbf{m} + \alpha\mathbf{m}^\star)$.*

Therefore, one can see the domain Ω as a partition of regions where η is not a root of P_n, and inside of each region, the number of terms in L remains constant.

4.1.2 Differentiable Weights. Each region of the domain Ω has a fixed number of terms, so, inside of it, the derivative of L simply reduces to the sum of derivatives of the weights w_i, and we observe:

Theorem 7. *For $(\eta, \mathbf{m}) \in \Omega$ the weights $w_0, \ldots, w_n$ associated with all points of the unique canonical representation are continuously differentiable in $\mathbf{m}$ and η.*

See Appendix C, Section C.4 for the (short) proof; it follows:

Corollary 8. *The bounding functions $L(\eta, \mathbf{m})$ and $U(\eta, \mathbf{m})$ are continuously differentiable in their domain Ω.*

4.2 Approaching Singularities

Although having established differentiability on the domain of L in Section 4.1, the result does not rule out discontinuities at points in

$$\Lambda = \{(\eta, \mathbf{m}) \in \mathbb{R} \times \mathbb{R}^{2n+1} : \text{positive}(\mathbf{m}) \wedge P_n(\eta; \mathbf{m}) = 0\}, \quad (21)$$

which is the set of points with strictly positive $\mathbf{m}$ but η is a root of P_n. Recall that the bounding functions are undefined at these points. In the following, we investigate the behavior of the points x_i as a point $(\eta, \mathbf{m}) \in \Omega$ in the domain of L approaches such *singularities*.

The polynomial $P_n(x, \mathbf{m})$ is part of a sequence of polynomials $\{P_i\}$, which are pairwise orthogonal under measures that are representations of $\mathbf{m}$. As an orthogonal polynomial, P_n has n real and simple roots $\{y_i\}_1^n$: at a point $(\eta, \mathbf{m}) \in \Omega$, η must lay between *at most* two roots of P_n, in either of the intervals $(-\infty, y_1)$, (y_j, y_{j+1}), or (y_n, ∞).

Geometrically, one can think of the roots as n distinct points $\mathbf{u}(y_i)$ on the moment curve, with $\mathbf{u}(\eta)$ between (at most) two of them. The situation $P_n(\eta; \mathbf{m}) = 0$ can now arise in different ways: (1) the

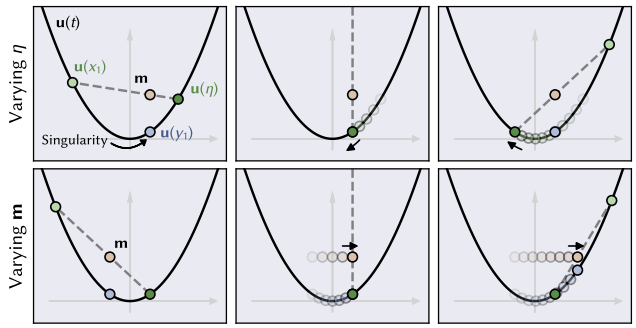
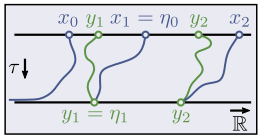


Fig. 3. Geometric relationship between the points of the canonical representation η and x_1 for a moment sequence $\mathbf{m} = (1, m_1, m_2)$ and the singularity point y_1, a root of P_n. The first dimension is omitted. First row: η approaches y_i. Second row: y_i approaches η as the moments are varied.

point $\mathbf{u}(\eta)$ moves towards one of the two roots on the curve, (2) one of the two points moves towards $\mathbf{u}(\eta)$ on the curve, or (3) a combination of the two. Figure 3 shows the behavior of the roots:

Theorem 9. *If $x_j(=\eta) < x_n$ approaches y_{j+1} on a continuously differentiable path, then $x_n \to \infty$. If $x_j(=\eta) > x_0$ approaches y_j on a continuously differentiable path, then $x_0 \to -\infty$.*



The proof is found in Appendix C, Section C.5. It exploits that the points $\{x_i\}_0^n$ (including η) strictly *interlace* with the roots $\{y_i\}_1^n$ of P_n in the domain Ω. At singularities, n of the points $\{x_i\}$ collapse to the n roots $\{y_i\}$ of P_n (see inset). As the $\{x_i\}$ and the roots $\{y_i\}$ are continuously differentiable functions in $(\eta, \mathbf{m})$, a path $\boldsymbol{\gamma}(\tau)$ from a point $(\eta_0, \mathbf{m}_0) \in \Omega$ to a singularity point $(\eta_1, \mathbf{m}_1) \in \Lambda$ is reversible, so each $\{y_i\}$ at the singularity corresponds exactly to one $\{x_i\}$. This leaves one of the outer points x_0 or x_n $(\neq \eta)$ without a corresponding root at the singularity, and it must go to $\pm\infty$. Since the theorem considers both directions, we can state the following:

Corollary 10. *If $x_j(=\eta) > y_j$ follows a continuously differentiable path through y_j such that after the singularity $x_{j-1}(=\eta) < y_j$, then the point x_0 goes to $-\infty$ and reappears as x_n from ∞. The other direction of x_j approaching y_{j+1} behaves analogously.*

Therefore, when crossing a singularity, exactly *one* root x_i changes its place with η, so for the sum in L (c.f. Eq. 18) the following holds:

Corollary 11. *As η follows a continuously differentiable path through a singularity, only the weight associated with x_0 or x_n is included in or excluded from the sum in the bounding functions.*

The last observation is essential as it allows studying only *one* of the weights and its derivatives to determine the limit behavior for the bounding functions L and U.

4.3 Weight (Derivative) at Singularities

In Section 4.1 we have shown that L is continuously differentiable on its domain Ω but the domain is a collection of distinct regions, connected by singularities at which L is undefined, not ruling out discontinuous behavior across singularities. We have then observed in Section 4.2 that when singularities are crossed, exactly one of the points $x_k \in \{x_i\}$ switches places with η, and therefore only its weight w_k is included or excluded from the sum in L (recall Eq. 18). It remains to investigate how the weight and its derivative, and therefore L and its derivative, behave when a singularity is approached, which is equivalent to studying their limit as $x_k \to \pm\infty$ (c.f. Theorem 9).

The possible outcomes are apparent: if the weight w_k, in the limit, goes to zero, the singularities in L are removable. If additionally the derivatives $\frac{\partial w_k}{\partial \eta}$ and $\frac{\partial w_k}{\partial \eta}$ go to zero, the singularities in the derivatives $\frac{\partial L}{\partial \eta}$ and $\frac{\partial L}{\partial \mathbf{m}}$ are removable. Otherwise, L or its derivatives will be discontinuous. We make the following two observations (proven in Appendix C, Sections C.6 and C.7):

Lemma 12. *If x_k is the point that goes to $\pm\infty$, then the associated weight w_k goes to zero.*

Lemma 13. *If x_k is the point that goes to $\pm\infty$, then the associated derivatives of the weight $\frac{\partial w_k}{\partial \eta}$ and $\frac{\partial w_k}{\partial \mathbf{m}}$ go to zero.*

The proofs use the fact that both the weight and its derivatives are rational functions in x_k, where the denominator has a (much) higher degree than the numerator, so they vanish as $x_k \to \pm\infty$. We conclude this section with our main result:

Theorem 14. *The bounding functions $L(\eta; \mathbf{m})$ and $U(\eta; \mathbf{m})$ are continuously differentiable on their domains Ω and the singularities are removable so that L and U are continuously differentiable over the set $\Omega \cup \Lambda \subset \mathbb{R} \times \mathbb{R}^{2n+1}$, with all strictly positive moments.*

5 IMPLEMENTATION

Algorithm 1 computes the moment bounds (the *primal* algorithm) and loosely follows Münstermann et al. [2018, Algorithm 1], where β blends between the lower and upper bound. We have not yet materialized all operations (e.g. root finding) because, as will be shown, the choice significantly affects the stability of the primal algorithm and its derivative, which is calculated by the *adjoint* algorithm.

Algorithm 1 Computing a moment-based bound (primal algorithm)
Inputs: moments $\mathbf{m} = (m_0, \ldots, m_{2n})$, evaluation point η, overestimation weight $\beta \in [0, 1]$, bias $\alpha \in [0, 1]$
Output: bound $b = (1 - \beta)L + \beta U$

$\mathbf{m} \leftarrow (1 - \alpha)\mathbf{m} + \alpha\mathbf{m}^\star$ ▷ Biasing (Prop. 6)
$\mathbf{H} \leftarrow$ `hankel_matrix`$(\mathbf{m})$ ▷ Eq. (13)
$\mathbf{c} \leftarrow$ `cholesky_solve`$(\mathbf{H}, (1, \eta, \eta^2, \ldots, \eta^n)^\top)$ ▷ Eq. (16)
$x_1, \ldots, x_n \leftarrow$ `find_polynomial_roots`$(\mathbf{c})$ ▷ Eq. (15)
$\tilde{\mathbf{m}} \leftarrow (m_0, \ldots, m_n)$
$w_0, \ldots, w_n \leftarrow$ `vandermonde_solve`$(\eta, x_1, \ldots, x_n, \tilde{\mathbf{m}})$ ▷ Eq. (12)
$b \leftarrow \beta w_0$
for $i = 1, \ldots, n$ **do**
  if $x_i < \eta$ **then**
    $b \leftarrow b + w_i$
return b

Automatic Differentiation. Algorithm 1 can be readily implemented in an automatic differentiation (AD) framework like PyTorch [Paszke et al. 2019]: a Cholesky solver is available, the roots can be determined in closed-form or, for high degrees, as the eigenvalues of the companion matrix (as in `numpy` [Harris et al. 2020]), and a generic linear solver can be used for the Vandermonde system.

While this straightforward implementation is a useful reference, it is numerically unstable and inefficient, both in terms of memory consumption and runtime. Improving the stability requires careful consideration of the numerically critical parts (Section 5.1). The derived primal and adjoint algorithms can be efficiently implemented by fusing all operations, avoiding the construction of a computation graph as in automatic differentiation (Section 5.2).

5.1 Explicit Algorithm and Reverse-Mode Differentiation

In this section, we will gradually materialize Algorithm 1, one operation at a time, and derive expressions for reverse-mode derivatives. This leads to concrete primal and adjoint algorithms. We will use the notation δa to denote the adjoint of a variable a, i.e., the derivative $\frac{\partial \mathcal{L}}{\partial a}$ w.r.t. a scalar function $\mathcal{L}(a)$, which could be the loss function in a gradient-based optimization. We assume adjoints with the same "shape" as the primal variable, i.e., if $\mathbf{b} \in \mathbb{R}^m$ then $\delta\mathbf{b} \in \mathbb{R}^m$.

Solving for the Kernel Polynomial. Since the Hankel matrix $\mathbf{H}$ of the biased moment vector is positive-definite and symmetric,

$$\mathbf{H}\mathbf{c} = \boldsymbol{\eta} \tag{22}$$

(c.f. Eq. 16) can be solved stable by performing the Cholesky decomposition of $\mathbf{H}$, followed by forward and backward substitution.

The output of this operation is the vector $\mathbf{c}$, so, in reverse-mode differentiation, the adjoint operation uses the input $\delta\mathbf{c}$ to compute $\delta\mathbf{H}$ and $\delta\boldsymbol{\eta}$. For a linear system $\mathbf{A}\mathbf{x} = \mathbf{b}$ with invertible $\mathbf{A}$, one can derive the equalities defining the adjoints [Strang 2007, Section 8.7]

$$\begin{aligned} \delta\mathbf{b} &= \mathbf{A}^{-\top}\delta\mathbf{x} \\ \delta\mathbf{A} &= -\delta\mathbf{b}\,\mathbf{x}^\top, \end{aligned} \tag{23}$$

which lead to explicit expressions for the outputs

$$\begin{aligned} \delta\boldsymbol{\eta} &= \mathbf{H}^{-1}\delta\mathbf{c} \\ \delta\mathbf{H} &= -\delta\boldsymbol{\eta}\mathbf{c}^\top, \end{aligned} \tag{24}$$

since $\mathbf{H}$ is symmetric. Therefore, $\delta\boldsymbol{\eta}$ is obtained by solving another linear system involving the Hankel matrix, with the same Cholesky decomposition, and the adjoint $\delta\mathbf{H}$ is simply an outer product.

Finding the Kernel Roots. By the structure of the problem, and critical for our proofs in Section 4, we know that one of the roots $\{x_i\}_1^n$ of the kernel polynomial is unbounded when the query point η approaches a singularity, as its leading coefficient c_n vanishes. Therefore, besides efficiency, numerical stability and adequate precision in 32-bit arithmetic are desired properties.

The cases $n = 1$ (3 moments) and $n = 2$ (5 moments) lead to kernel polynomials of degree $d = 1$ and $d = 2$, respectively, whose roots can be found using closed-form expressions. For the quadratic equation, we have implemented a robust solver following Kahan [2004]. Although closed-form expressions exist for degrees 3 and 4, we opt for a fast and more accurate Newton-style approach for degrees > 2 [Yuksel 2022]. Notably, vanishing c_n is naturally handled in our implementation by returning one root with value `inf` instead of branching to a different version of the algorithm or trying to explicitly avoid this case by comparing η to the singularity points (the roots of P_n, Section 4.1.1). All subsequent operations gracefully handle roots at infinity by returning the limit value.

If $\{x_i\}_{i=1}^n$ are the roots of the kernel polynomial $K(\mathbf{c}, x)$ with coefficients $\mathbf{c}$, the adjoint operation computes $\delta\mathbf{c}$, given $\{\delta x_i\}$:

$$\delta\mathbf{c} = \sum_{i=1}^{n} \frac{\partial x_i}{\partial \mathbf{c}} \delta x_i. \tag{25}$$

The implicit function theorem relates the roots to the coefficients

$$\frac{\partial x_i}{\partial \mathbf{c}} = -\frac{\begin{pmatrix}1 & x_i & \dots & x_i^n\end{pmatrix}^\top}{\frac{\partial K}{\partial x}(\mathbf{c}, x_i)} \tag{26}$$

This expression is undefined for a root at infinity but our proofs in Section C.7 show that the product

$$\frac{\partial x_i}{\partial \mathbf{c}} \underbrace{\frac{\partial w_i}{\partial x_i}\delta w_i}_{:=\delta x_i} \tag{27}$$

goes to zero when approaching singularities. Since the result becomes less reliable with large x_i, we test if the quadratic factor in the denominator of the derivatives $\frac{\partial w_i}{\partial \eta}$ (Eq. (55)) and $\frac{\partial w_i}{\partial \mathbf{m}}$ (Eq. (60)) overflows and if so, set the contribution of x_i to $\delta\mathbf{c}$ to zero.

Solving the Vandermonde System. Instead of solving the Vandermonde system with a generic approach, we use the iterative algorithm by Björck and Pereyra [1970] as we found it to be efficient, stable, and simple to implement. The algorithm is derived from interpolation with divided differences, which is also the suggested method by Münstermann et al. [2018]. Additionally, we observe that it gracefully handles inputs close to or exactly at infinity: if a point x_i is at infinity with value `inf`, the associated solution w_i becomes zero. This corresponds exactly to the expected limit behavior (Section 4.3). The stability of the algorithm generally depends on the order of the inputs $\{\eta, x_1, \dots, x_n\}$ [Higham 1987]. We found that ordering the roots ascending by absolute value drastically improves the precision (this is effectively pivoting), and only then the limit value is obtained.

The inputs to the algorithm are the roots $x_0 (= \eta), x_1, \dots, x_n$ and the first $n + 1$ entries of the moment vector $\mathbf{m}$, which we denote by $\bar{\mathbf{m}} = (m_0, ..., m_n)^\top$. The adjoint of the partial moment vector is obtained by re-using the equalities for the linear system (Eq. (23)):

$$\delta\bar{\mathbf{m}} = \mathbf{V}^{-\top}\delta\mathbf{w}, \tag{28}$$

where $\mathbf{w} = (w_0, w_1, \dots, w_n)^\top$ is the vector of weights and its adjoint $\delta\mathbf{w}$ is an input. We obtain $\delta\bar{\mathbf{m}}$ with an algorithm for solving the *dual* Vandermonde system by Björck and Pereyra [1970]. Intuitively, the solution is a polynomial interpolating the values $\{\delta w_i\}$ at points $\{x_i\}$. If a root x_i is at infinity, its value δw_i does not contribute to the interpolation (this can be seen by expressing the derivative in Eq. (57) with the Vandermonde matrix), so the resulting polynomial has one less degree, i.e., $\delta\bar{\mathbf{m}}_n = 0$. This is exactly the output of the algorithm for an x_i at infinity.

The same equalities provide an expression for the adjoint matrix

$$\delta \mathbf{V} = -\delta \bar{\mathbf{m}} \mathbf{w}^\top, \tag{29}$$

which leads to the root adjoints

$$\delta x_i = \sum_{j=0}^{n} \sum_{k=0}^{n} \delta \mathbf{V}_{j,k} \left[\frac{\partial \mathbf{V}}{\partial x_i} \right]_{j,k} = -\frac{\partial}{\partial x_i} \Big(\sum_{j=0}^{n} \delta \bar{\mathbf{m}}_j w_i x_i^j \Big), \tag{30}$$

This is simply the derivative of a polynomial with coefficients $\delta \bar{\mathbf{m}} w_i$ evaluated at x_i. For x_i at infinity, it may be undefined in floating point arithmetic and this case is explicitly handled in the adjoint root finding (see discussion around Eq. (27)).

Remaining Operations. The moment biasing, Hankel matrix construction, and weight summation require no special considerations.

5.2 Fused Implementation

Implementing the AD version or the explicit algorithms (Section 5.1) in a framework like PyTorch can be disappointing performance-wise: each operation invokes a dedicated kernel, with the launch overhead quickly accumulating for complex computation graphs, and more importantly, all intermediate results are materialized in memory, which not only leads to excessive storage requirements but also results in expensive reads from (GPU) memory. Just-in-time (JIT) compilation can mitigate these issues but comes at the cost of losing control over the generated kernels and might not lead to the expected improvements: for example, we have JIT-compiled the AD implementation (Section 5, *Automatic Differentiation*) using PyTorch 2.x's `compile` feature, measuring either negligible improvements ($n = 1, 2$) or even worse runtime ($n > 2$), because some operations are currently unsupported (e.g. finding eigenvalues of a matrix).

Instead, we have implemented the primal and adjoint algorithms from Section 5.1 entirely in C++, only depending on a small subset of the standard library. The floating point type and n are known at compile time, leading to the generation of highly optimized code. We instantiate and test variants for 32-bit `float` and 64-bit `double`, and $n = 1, \ldots, 5$. GPUs are targeted using CUDA, such that all operations of the primal algorithm are compiled into a single CUDA kernel, and similarly, a single kernel is compiled for the adjoint algorithm. Global memory accesses are often the bottleneck of a GPU implementation, so we ensure that intermediate values can reside in registers. Specifically, we avoid register spilling to (local) memory by only statically indexing into arrays, for example during the root finding procedure [Peters 2023].

The C++/CUDA implementation is exposed to Python using nanobind [Jakob 2022], so it can readily interface with array programming frameworks like PyTorch, numpy, or JAX.

Accuracy Verification. We first verify the accuracy of the C++ implementation by comparing it to the PyTorch (AD) implementation for 1 million random samples. The average accuracy in the computed roots and weights is comparable (Table 1, Appendix A). The gradients of the bound computed by the adjoint procedure are verified by comparing them to finite differences. The C++ implementation produces errors almost equal to the PyTorch AD implementation, differing in the 3rd to 4th digit (Table 1). We exclude samples with η near a singularity, so this merely serves as a sanity check and the limit behavior is studied separately (Paragraph *Stability Analysis*)

Performance. We compare the runtime and memory consumption of the C++/CUDA implementation to the PyTorch-based AD implementation, on the GPU, using 2 million randomly sampled moments $\{\mathbf{m}_i\}$ and points $\{\eta_i\}$ for varying n (Figure 4). Across all tests, the custom CUDA implementation is considerably faster, both for the primal (*forward*) and adjoint (*backward*) computations as well as in 32-bit and in 64-bit precision. The over 1000× increase in runtime ($n = 3$, 32 bits) is related to root finding using the companion matrix for $n > 2$, because PyTorch seemingly does not solve for eigenvalues on the GPU (the CPU runtime is similar). Surprisingly, the corresponding function is also faster for 64-bit input, which explains why the 64-bit primal algorithm for $n > 2$ is faster than the 32-bit variant. We have no conclusive explanation but this behavior is consistent across different machines, operating systems, and PyTorch versions.

Regardless, the variants $n = 1$ and $n = 2$ use closed-form expressions for the roots, so they are unaffected by this anomaly. In 32-bit, the CUDA implementation is up to three orders of magnitude faster, and in 64-bit up to two orders. The default variant of the CUDA implementation recomputes *all* intermediate values from the forward pass in the backward pass, which makes it at least as expensive as the primal algorithm. Still, the runtime is much lower than the AD graph traversal. The observed performance benefit in an application will certainly be lower because a differentiable rendering pipeline consists of various costly operations and the GPU might not be saturated by the number of computed bounds. If the algorithm is used for shadow mapping, 2 million bounds correspond roughly to rendering a 1k image with two light sources. Even for 50× larger input (∼ 100 Million bounds), for $n \leq 2$, the 32-bit primal algorithm runs in less than 5 milliseconds and the adjoint algorithm in under 20 milliseconds (Figure 5, left column). The double-precision runtime is significantly worse and roughly aligns with the theoretical performance of the tested GPU (64× higher FP32 throughput).

The memory footprint of the CUDA implementation is very predictable as storage from global memory is only allocated for the output. If N is the number of bounds to compute, the primal algorithm generates an array of size N and the adjoint algorithm an array of size $N(2n + 2)$ (for $2n + 1$ moment adjoints and the adjoint of η), therefore the memory requirements are linear (Figure 5, right column, mind the log-scaled x-axes). Specifically, in contrast to the PyTorch AD implementation, no computation graph is constructed during the primal algorithm, which leads to a *significantly* smaller memory footprint (Figure 4). For the adjoint computation, the AD implementation also only allocates memory for the outputs.

Retaining Roots. The default C++/CUDA implementation outputs only the bounds from the primal kernel. In the adjoint kernel, the required quantities (e.g., the Cholesky factorization and the roots) are recomputed by re-running the primal algorithm. For $n > 2$, root finding is an iterative procedure (Section 5.1), which can become the dominating operation during the adjoint phase, in terms of runtime. We have therefore implemented a "retained roots" variant that returns the roots from the primal computation and re-uses them during the adjoint computation. The variant has a similar computation time for the primal part and an increased memory complexity, which is still significantly less than AD, but a considerably lower runtime for the adjoint part for $n > 2$ (Figures 4 and 5).

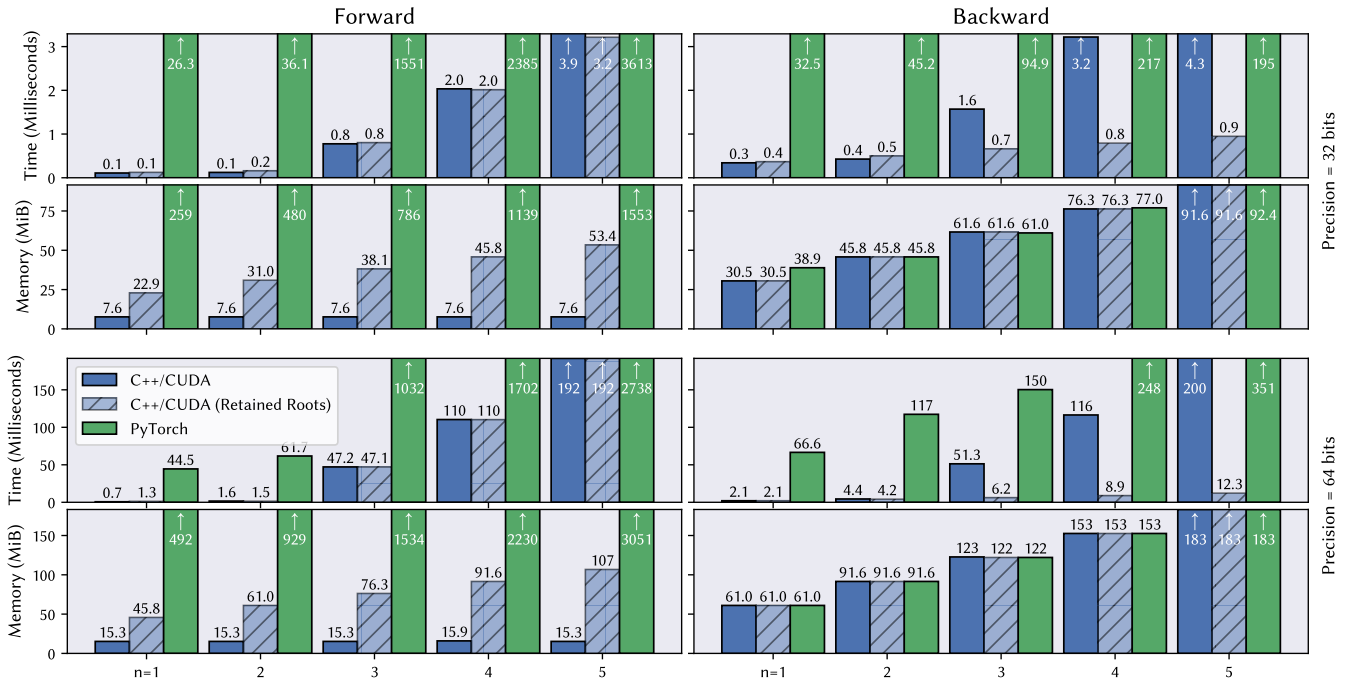


Fig. 4. Runtime and memory requirements of forward and backward implementations of moment-based bounds. The benchmark uses 2 million randomly sampled points $(\eta, \mathbf{m})$. All measurements were performed on a workstation with an Intel i7-13700 CPU and an NVIDIA RTX 3090 GPU (24 GB VRAM). The PyTorch implementation is based on automatic differentiation applied to Algorithm 1 (Section 5, Paragraph *Automatic Differentiation*), while the C++/CUDA implementation follows Section 5.1. The *Retained Roots* mode stores the roots of the kernel polynomial between the forward and backward pass.

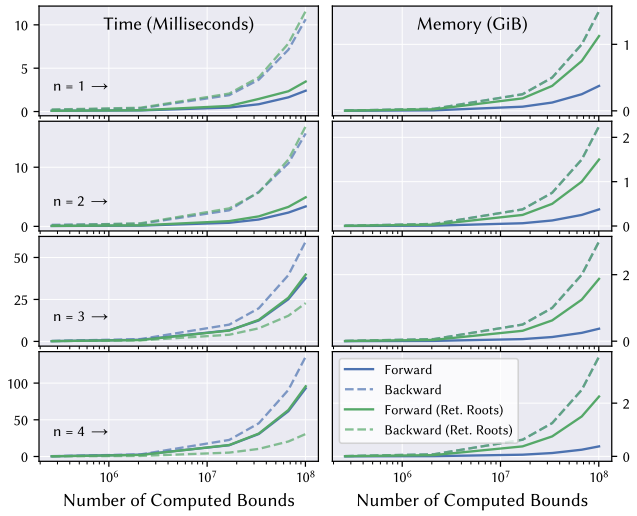


Fig. 5. Runtime and memory requirements of the 32-bit C++/CUDA implementation with increasing number of computed bounds. The rows correspond to orders $n = 1, 2, 3, 4$, from top to bottom. Mind the log-scaled x-axis.

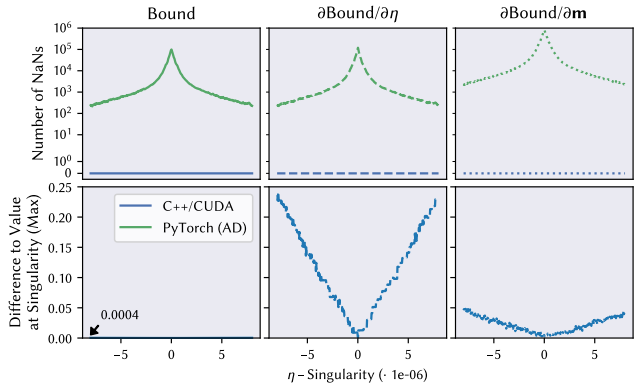


Fig. 6. Evaluating the bound near singularities can be numerically unstable, leading to undefined output in the forward and backward computation (top). The implementation of an algorithm that considers the properties of the bound at singularities ("C++/CUDA", Section 5.1) is more robust than the naive implementation based on automatic differentiation (AD).

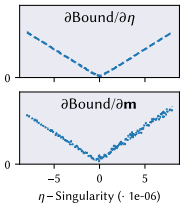
Stability Analysis. The bounding functions are undefined at a finite set of points, where η takes the value of one of the roots of the polynomial P_n (and this is not a pathological case, as shown in Section 6.1.3). As η approaches a singular point, one root x_i tends to infinity, which can compromise the accuracy of the bound and its derivatives, specifically in 32-bit precision.

We use the following procedure to study the implementations near the singularities: for moments $\mathbf{m}$, we generate points η_k inside an ϵ-neighborhood (ϵ = 8e-6) around each singularity y_j (j =

$1, \dots, n$), and compute the bound and its derivatives. We also evaluate the bound and the derivatives *exactly* at the singular point (up to floating-point precision). At each η_k, two traits are desirable : (1) the bound and the derivatives remain finite since undefined values can severely compromise a gradient-based optimization, and (2) their values tend towards the value at the singularity.

For $n = 1, \dots, 5$ we randomly generate moments, then compute the number of NaNs and the (absolute) difference to the value at the singularity (Figure 6). Results are for 32-bit precision and the difference is reported as the *maximum* across all n and moment vectors. The PyTorch (AD) implementation generates almost no usable values near a singularity, which, to a large part, is due to failures in its root-finding procedure. The C++ implementation generates no NaN output since its operations are carefully designed around the limit case (Section 5.1). This is also reflected by the values tending to the value at the singularity (Figure 6, second row).

We found the higher-order powers in the computation of δx_i (Eq. (30)) and the product in Eq. (27) particularly troublesome: without the measurements for $n = 5$, the derivatives tend even more clearly to the singularity value (inset). Unexpectedly, the 64-bit precision variant behaves *less* robust near singularities, which is likely due to the reliance on overflow semantics. We consider the 64-bit implementation experimental and leave a deeper investigation to future work.



6 APPLICATIONS

6.1 Differentiable Shadow Mapping

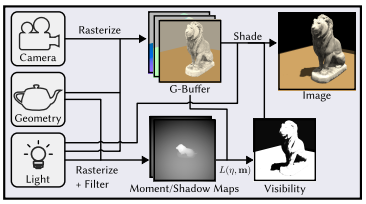
In rendering, shadows from point lights (including those at infinity) are often approximated with *shadow mapping* [Williams 1978]. Traditionally, the *shadow map* is a depth image rendered from the perspective of the light, and a point in the scene is in shadow if its distance to the light is larger than the corresponding depth value in the shadow map. However, the derivative of this binary test is not useful as it is zero for the point's distance and the depth value. Worchel and Alexa [2023] show that *pre-filtered shadow mapping* [Annen et al. 2007, 2008; Donnelly and Lauritzen 2006; Peters and Klein 2015] is a suitable framework: the binary visibility test is replaced with a soft test that induces image derivatives at shadow boundaries.

In moment-based techniques, one first obtains a set of moment maps by filtering the shadow map f with a compact kernel:

$$f_k(\mathbf{u}) = m_k(\mathbf{u}) = \int_I f(\mathbf{p})^k g(\mathbf{p} - \mathbf{u})\, \mathrm{d}\mathbf{p}, \quad 0 \le k \le 2n, \tag{31}$$

where the kernel fulfills $\int g(\mathbf{p}) = 1$, so $m_0 = 1$. If $d_{\mathbf{x}}$ is the distance of a point $\mathbf{x}$ to the light and $\mathbf{u}_{\mathbf{x}}$ is its pixel position in the shadow map, the shadow test is $L\big(d_{\mathbf{x}}, \mathbf{m}(\mathbf{u}_{\mathbf{x}})\big)$, which is a lower bound to the result obtained with percentage-closer filtering [Reeves et al. 1987].

Worchel and Alexa [2023] show that the visibility test of Variance Shadow Mapping (VSM) [Donnelly and Lauritzen 2006], based on two power moments $(1, m_1, m_2)$, is differentiable. Our proof shows that L is differentiable for arbitrary number of moments, motivating more general techniques like Moment Shadow Mapping (MSM) [Peters and Klein 2015] with four power moments $(1, m_1, m_2, m_3, m_4)$.

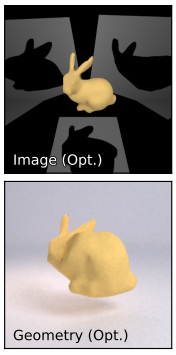


We have implemented a rasterization-based differentiable renderer with shadow mapping (inset): the geometry is first rasterized, producing the G-buffer, a collection of attribute images needed for shading (e.g. diffuse albedo and normals). Shadow maps are rendered and used to compute the surface visibility. All data is then combined to the final image. Gradients flow along the shown edges.

6.1.1 Light Leaking in Appearance Estimation. Variance Shadow Mapping suffers from light leaking: if the depth has high variance in the filter support, the computed bound is not sharp and visibility is overestimated. The adverse effect of light leaking is more significant in *inverse* settings: in forward rendering, subtle artifacts might stay unnoticed by human observers, but with differentiable rendering, images are often compared to a reference that potentially originates from the physical world. Moment Shadow Mapping strongly reduces light leaking by providing a sharper bound to the visibility (Figure 1).

Faithful estimation of visibility is critical when the *appearance* of an object and the *illumination* are decomposed, e.g. as a post-processing step of a (neural) 3D reconstruction pipeline. Jointly optimizing the light direction and albedo texture from a single reference image yields significantly more accurate results with Moment Shadow Mapping (Figure 7): while VSM and MSM obtain the correct light direction, the VSM albedo texture shows traces of the shadow because it must account for areas with reduced shadow intensity. Our MSM implementation not only generates higher quality textures but is as fast as the VSM implementation by Worchel and Alexa [2023] (Figure 8). The C++/CUDA implementation (Section 5.2) is critical to matching the performance: a PyTorch (AD) implementation of MSM is observed to have twice the VSM runtime, and, more importantly, does not converge due to NaN outputs (Section 6.1.3).

6.1.2 Comparison to Ray Traced Shadows. Instead of approximating visibility, it can be accurately determined using ray tracing, and, more generally, realistic images can be synthesized with systems for physically-based, differentiable rendering. The impact on runtime is significant: an approximate, differentiable renderer with Moment Shadow Mapping finishes an image-based reconstruction task in seconds while the physically-based system Mitsuba 3 [Jakob et al. 2022b] requires minutes (Figure 9). Importantly, the visibility gradients generated by MSM lead to a successful reconstruction: if the visibility is detached from the AD graph, the reconstruction fails (inset). With significant contribution from indirect illumination, approximating only shadows would not be sufficient, and additional techniques are required.



6.1.3 Numerical Considerations. The PyTorch AD implementation, in contrast to the C++/CUDA implementation, generates undefined outputs and gradients if the query point is close to a singularity

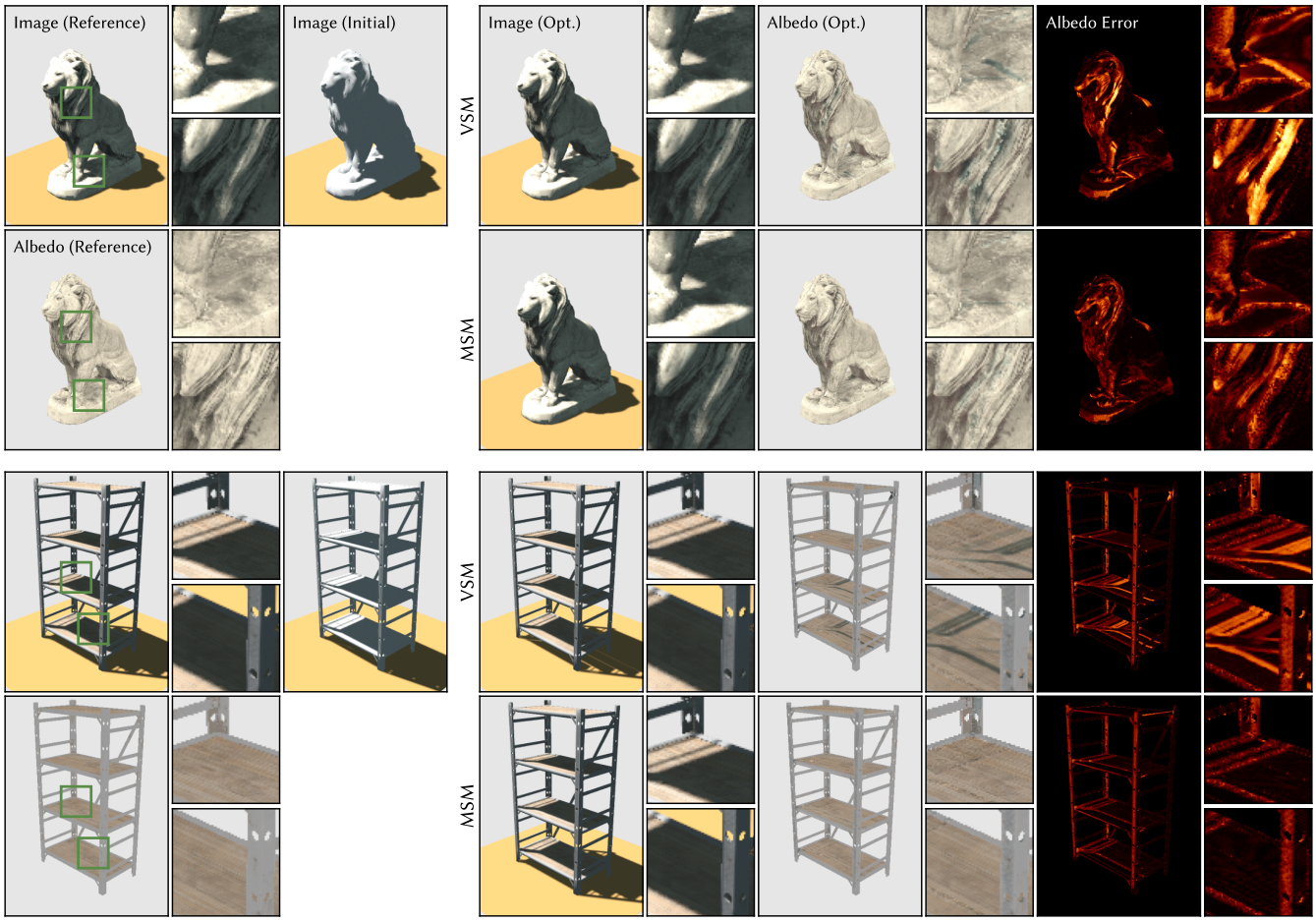


Fig. 7. Optimizing the light position and spatially-varying albedo of an object from a single reference image. Both, Variance Shadow Mapping (VSM) and Moment Shadow Mapping (MSM) are based on the theory of moments, but Moment Shadow Mapping uses more moments, which leads to a more accurate estimation of visibility. This is crucial if the appearance of an object is reconstructed because otherwise, remains of the shadow are baked into the albedo (see albedo optimized with VSM). The target reference image is generated using percentage-closer filtering [Reeves et al. 1987].

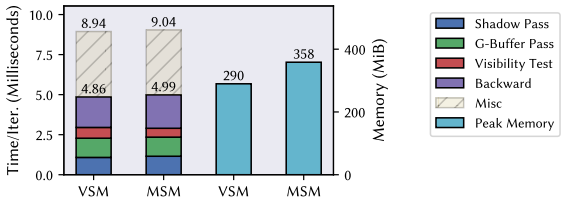


Fig. 8. Runtime of Variance Shadow Mapping (VSM) and Moment Shadow Mapping (MSM) for the appearance estimation task. Both methods are similarly efficient, but MSM requires slightly more memory. "Misc" includes the remaining operations, such as anti-aliasing and evaluating the objective.

(Section 5.2, Paragraph *Stability Analysis*). In forward rendering, this merely leads to visual artifacts, but in inverse rendering, this can corrupt a (potentially long-running) optimization. Figure 10 shows that this is not a pathological case: almost all surfaces directly visible to a light will result in queries close to singularities. The reason is that the moments are generated by filtering over a compact region in the shadow map (Eq. (31)). Except at discontinuities, the depth values in the filter support will be correlated and only subtend a small section of the moment curve with concentrated singularities. The depth values of directly visible points will be close to the depths in the filter support, so fall within that same segment of the moment curve and near the singularities. For this reason, we were unable to achieve a stable convergence using the PyTorch AD implementation.

Moment Biasing. A central feature of Moment Shadow Mapping is the ability to *exactly* reconstruct the visibility if the filter support region contains no more than two surfaces (perpendicular to the light direction) [Peters and Klein 2015]. For such cases, the moments are slightly biased to ensure that the Hankel matrix is non-singular. We use the bias vectors by Münstermann et al. [2018] and test the

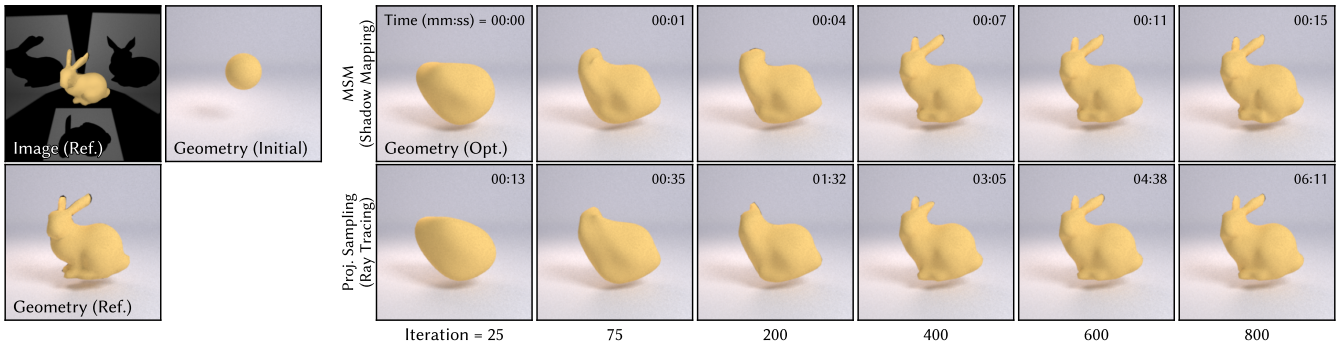


Fig. 9. 3D mesh reconstruction from a single image of an object illuminated by three spotlights. The differentiable, moment-based shadow mapping technique (MSM) convergences significantly faster than the ray tracing approach based on projective sampling [Zhang et al. 2023], with similar reconstruction results. The reference image is rendered using ray tracing and serves as a reference for both methods. The reference image does not include indirect illumination. The optimization uses gradient preconditioning [Nicolet et al. 2021] and a rotation-equivariant instance of ADAM [Kingma and Ba 2015; Ling et al. 2022].

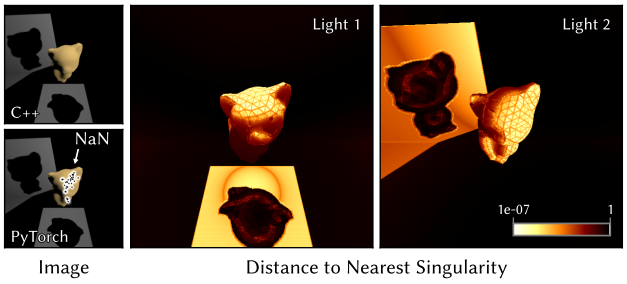


Fig. 10. Moment shadow mapping uses bounding functions that are undefined at certain points, close to which evaluating the bound and its derivatives becomes numerically challenging. Even in a simple scene, query points are close to these singularities, which leads to undefined output in naive implementations ("PyTorch"), motivating more robust approaches ("C++").

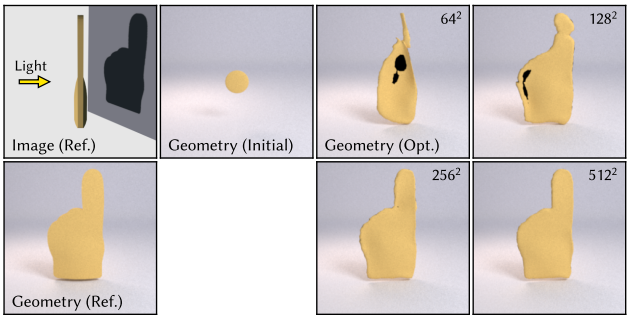
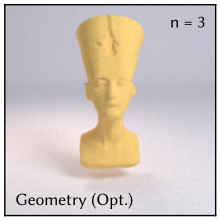


Fig. 11. An optimization with shadow mapping can fail if the underlying differentiable renderer misses the relation between changes in geometry to changes in the shadow map (here at caused by low shadow map resolution).

recommended bias for $n = 2$ ($\alpha = 5 \cdot 10^{-7}$), but singular matrices are still occasionally encountered. We use a slightly higher bias (as a rule of thumb, at least twice the recommended value).

Some experiments run successfully with the bias recommended by Münstermann et al. [2018, Table 1, Supplementary Material] (e.g. Section 6.1.1), so it is worth testing different values to strike the right balance between accuracy and robustness in an application. With stable biasing, the number of moments can be increased from 4 ($n = 2$) to 6 ($n = 3$) and further, still resulting in successful 3D reconstructions (inset).



6.1.4 Quality of Rasterization Derivatives. The quality of the approximated visibility decreases with the shadow map resolution. Low-resolution shadow maps also harm the inverse setting (Figure 11). First, they cannot reproduce fine structures in the target shadow, and second, they are limited by the differentiable rasterizer: the implementation is based on NVDIFFRAST [Laine et al. 2020], which differentiates discontinuous coverage by anti-aliasing silhouettes. If pixels are larger than (projected) triangles, the silhouette detection can fail and coverage is not differentiated (Figure 12). As a result, shadow boundaries carry no gradients w.r.t. the geometry.

Worchel and Alexa [2023] also observe this behavior, although they do not show the adverse effect in a reconstruction task, and the experiments suggest that increasing the filter kernel size can mitigate the issue, which is, however, not the case for low-resolution shadow maps. This is not a conceptual issue of differentiable shadow mapping and it can likely be mitigated by using other approaches for differentiable rasterization [Liu et al. 2019; Pidhorskyi et al. 2025].

6.2 Differentiable Visibility for Volume Rendering

Volume rendering [Kajiya and Von Herzen 1984] is commonly used in computer vision for volumetric scene representations, such as (neural) radiance fields [Mildenhall et al. 2020]. If $\mathbf{r}(t) = \mathbf{o} + t\mathbf{d}$ is a camera ray, the image formation model assigns to it a color

$$\int_0^\infty \mathbf{c}(\mathbf{r}(t))\, T(t)\, \sigma(\mathbf{r}(t))\, \mathrm{d}t, \tag{32}$$

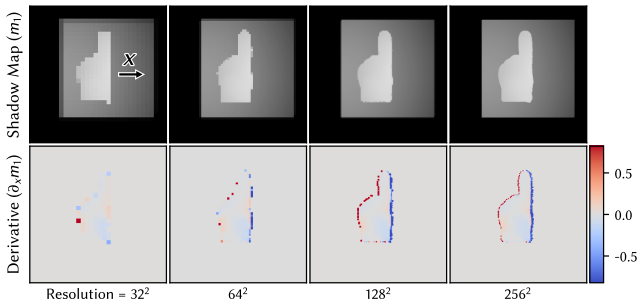


Fig. 12. Differentiable shadow mapping can fail at capturing the changes of the shadow map with respect to movement of the geometry, depending on the underlying differentiable renderer. In this specific case, the renderer relies on silhouette detection, which fails for low-resolution shadow maps.

where $\mathbf{c}$ is a color field, σ is a density field and T is the transmittance. Intuitively, the density σ measures the probability (per unit length) of absorbing or scattering a photon from its path and transmittance

$$T(t) = \exp\Big(-\int_0^t \sigma\big(\mathbf{r}(z)\big)\,\mathrm{d}z\Big) \tag{33}$$

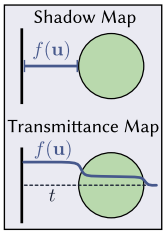
is the fraction of radiance being transported from 0 to t.

When volume rendering is combined with a physically-based image formation model, e.g. to decompose appearance and illumination or use shadows as additional reconstruction cues, one must compute the transmittance from a light source to each point on a primary ray. Numerically estimating the transmittance can lead to quadratic complexity, which, even for small images, quickly becomes infeasible, especially when storing a computation graph for reverse-mode differentiation. Several ways of *approximating* the transmittance have been proposed, one of which is applying shadow mapping by rendering and storing the expected termination depth from the perspective of a light, for each shadow map pixel $\mathbf{u}$,

$$f(\mathbf{u}) = \int_0^\infty t\,T(t)\,\sigma\big(\mathbf{r}(t)\big)\,\mathrm{d}t, \tag{34}$$

and using soft, ad-hoc extensions of the binary shadow test to make it differentiable [Liang et al. 2022; Tiwary et al. 2022]. While the approximation is efficient and induces dense gradients in a volume, it is difficult to achieve shadows with reasonable intensity (Section 6.2.1) and the approach is not volume-based, i.e., it does not properly handle empty space, which is desirable when combining multiple volumes or different scene representations (Section 6.2.2).

Transmittance Mapping. Inspired by classical work in real-time graphics [Delalandre et al. 2011; Jansen and Bavoil 2010; Lokovic and Veach 2000; Peters et al. 2016] we propose a principled approach to transmittance approximation for differentiable volume rendering. Similar to how a shadow map stores the minimum depth along a ray emanating from the light source, a *transmittance map* stores a compact representation of the transmittance *function* along a ray (inset).



Following Münstermann et al. [2018], we define absorbance as

$$A(t) = -\ln\big(T(t)\big) = \int_0^t \sigma\big(\mathbf{r}(z)\big)\,\mathrm{d}z. \tag{35}$$

A is non-decreasing and non-negative as long as σ, the volume density, is non-negative. This allows approximating A using moments (Section 3): a transmittance map is built by computing the moments

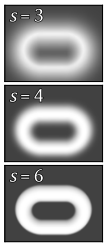
$$f_k(\mathbf{u}) = m_k = \int_0^\infty z^k \sigma\big(\mathbf{r}(z)\big)\,\mathrm{d}z, \quad 0 \le k \le 2n. \tag{36}$$

The representation is compact because it only stores $2n+1$ values in each pixel ("TM-n" will refer to order-2n transmittance mapping, i.e., a transmittance map with moments $m_0, \ldots, m_{2n}$). The bounding functions from Equation (4) then approximate the transmittance:

$$T(t) \approx \exp\Big(-\big(1-\beta\big)L(t) - \beta U(t)\big)\Big), \tag{37}$$

where $\beta \in [0,1]$ blends between the bounds [Münstermann et al. 2018]. The approximation quality increases with the order n and transmittance maps are filterable, i.e., the approximation still "works" as expected when they are filtered or sampled from with (bilinear) interpolation. Using the alternative form from Appendix B, following Peters et al. [2016], transmittance maps can also be easily integrated into existing volume rendering frameworks, e.g. by Li et al. [2023].

Implicit Surfaces and Volumes. Differentiable visibility in volumes might be of particular interest to the 3D reconstruction community because the properties of volume rendering, namely trivial differentiability and dense gradients, are often exploited for reconstruction using implicit surfaces [Oechsle et al. 2021; Wang et al. 2021; Yariv et al. 2021]: constructions like NeuS [Wang et al. 2021] use differentiable volume rendering for an indicator or signed distance function $g(\mathbf{x})$ transformed into a volume density[3]



$$\sigma\big(\mathbf{r}(t)\big) = \phi_s\Big(g\big(\mathbf{r}(t)\big)\Big). \tag{38}$$

The parameter s controls the spread of the bell-shaped function ϕ_s and therefore the opacity of the induced density field (inset). We use this construction throughout the section, so the shown volumes are induced by some signed distance function.

6.2.1 Visibility Gradients for the Density. We first verify, using simple, analytic shapes, that the gradients computed from the transmittance approximation affect the density field and the underlying implicit surface. Figure 13 shows that a perturbation of geometry is reflected in the derivatives, both for directly visible surface points and the cast shadow computed using transmittance mapping.

The quality of the approximation and its gradients is compared to shadow mapping using a simple task: recover the pose of an object, i.e., the position and orientation, from only a *single* image. Shadows can be important cues for this task [Liu et al. 2023; Shafer and Kanade 1983] and it might be approached by volume rendering implicit surfaces [Qu et al. 2023]. The image formation model simply accumulates the visibility V along the primary rays

$$\int_0^\infty V\big(\mathbf{r}(t)\big)\,T(t)\,\sigma\big(\mathbf{r}(t)\big)\,\mathrm{d}t, \tag{39}$$

[3] A direction-dependent normalization factor is omitted for brevity.

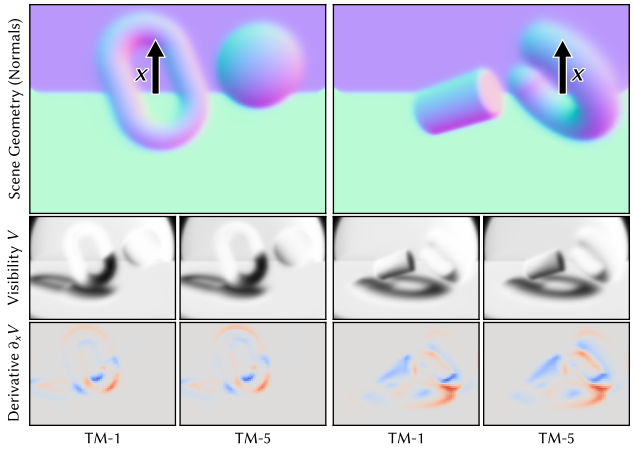


Fig. 13. The quality of the approximation of transmittance increases with the number of moments. TM-n denotes an order n with $2n + 1$ moments. The approximation is differentiable and perturbation of the geometry with a parameter x is correctly reflected in the derivative

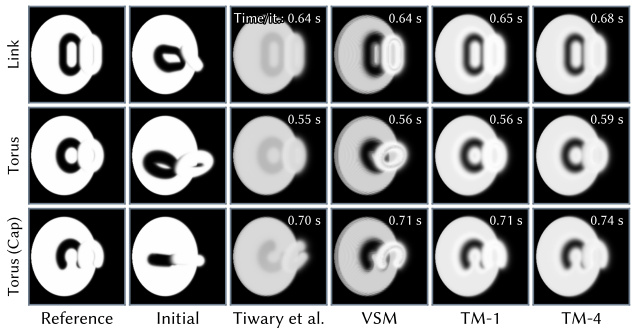


Fig. 14. Optimizing the pose of simple, analytic shapes with differentiable volume rendering. The approximation of shadow mapping ([Tiwary et al. 2022] and VSM) is not accurate and often fails to find the pose, while the more principled moment-based (TM-n) approach succeeds. The reference is rendered at a slightly higher density ($s = 5$) than used in the optimization ($s = 4$) to mimic a more "realistic" reference.

where V is the (approximate) transmittance from the light source to the point $\mathbf{r}(t)$. Even in this simple setting, differentiating shadow rays is already infeasible due to the excessive memory requirements.

We compare against the shadow mapping variants by Tiwary et al. [2022], which has a sigmoid-based soft visibility test, and Variance Shadow Mapping (VSM) by Donnelly and Lauritzen [2006][4]. Figure 14 shows that the shadow mapping visibility is far from the shadow ray reference, leading to inaccuracy in the estimated pose.

We measure the translation error $\Delta\mathbf{t}$, the Hausdorff distance D_H, both scaled by factor 10, and the runtime per iteration t, as the median over multiple runs, averaged over different shapes (inset table).

[4]While VSM has only been proposed for differentiable rasterization [Worchel and Alexa 2023], it is included because it has a more principled visibility test.

Transmittance maps consistently outperform shadow mapping in the geometric error measures. The difference between the transmittance map orders is marginal but could become noticeable for multiple (articulated) objects. Most importantly, transmittance mapping is more accurate and at the same time equally efficient.

	↓ $\Delta\mathbf{t}$	↓ D_H	↓ t [s]
Tiwary et al.	0.50	0.33	**0.58**
VSM	0.94	0.67	**0.58**
TM-1	0.09	**0.08**	0.59
TM-2	**0.07**	**0.08**	0.59
TM-3	0.08	0.09	0.60
TM-4	0.08	0.09	0.61
TM-5	**0.07**	**0.08**	0.63

6.2.2 Visibility Through Empty Space. The issue arising from shadow mapping not being a principled method for approximating transmittance is best shown by testing a trivial property of transmittance: if the density in a volume is zero, the transmittance is one, because no light is absorbed. The test setting is inspired by shadow art [Mitra and Pauly 2009]: planes (triangle meshes) are placed around a grid representing an SDF. The SDF and the density-controlling variable s are optimized to cast given target shadows on the planes, by using an objective function that compares renderings of the cast shadows to the targets. The topological flexibility of the SDF is beneficial for this task and volume rendering ensures dense gradients. To maintain a valid SDF, the function is redistanced by solving the Eikonal equation in each iteration [Detrixhe et al. 2013; Vicini et al. 2022].

Shadow rays do not have quadratic complexity in this setting because the receiver is rasterized, not volume-rendered, so they can serve as a reference. While transmittance maps generate results close to the shadow ray solution, shadow mapping fails (Figure 15): since the receiver planes are not volumes, they are not considered in volume rendering, so, whenever a light ray misses the implicit surface, the expected terminated depth will be zero (compare Eq. 34). Shadow mapping behaves as if an object directly in front of the light source obstructs the view. Therefore, the unobstructed parts of the receiver plane will appear in shadow, and the obstructed parts will, too. Since the shadow intensity in the method by Tiwary et al. [2022] depends on the distance between the occluder and receiver some areas in shadow appear grayish. The optimization behaves contrary to the expectation and recreates the bright parts of the target image using the volume (the sphere emerges) while the shadow is recreated by generating free space. Transmittance maps handle free space correctly by reporting constant 1 transmittance, while being about 4× faster than shadow rays – as fast as shadow mapping.

6.2.3 3D Reconstruction from Low-Density Volumes. Shadow mapping does not necessarily fail and has been successfully used in differentiable volume rendering: if s is high, the volume density spikes at the surface, and volume rendering collapses to surface rendering. Since s (c.f. Eq. 38) is often optimized with the volume, from low to high [Li et al. 2023; Wang et al. 2021], the point at which shadow mapping becomes a good approximation depends on the optimization schedule. This leads to an intricate balance: starting with a very low s induces dense gradients, which more robustly localize the surface, independent of the initial state. However, the inaccurate visibility approximation at earlier iterations increases the risk of becoming stuck in suboptimal local minima.

This effect is observable in 3D reconstruction from multi-view images (Figure 16). Shadow mapping (VSM) navigates into an unrecoverable state by decreasing the target density until the volume

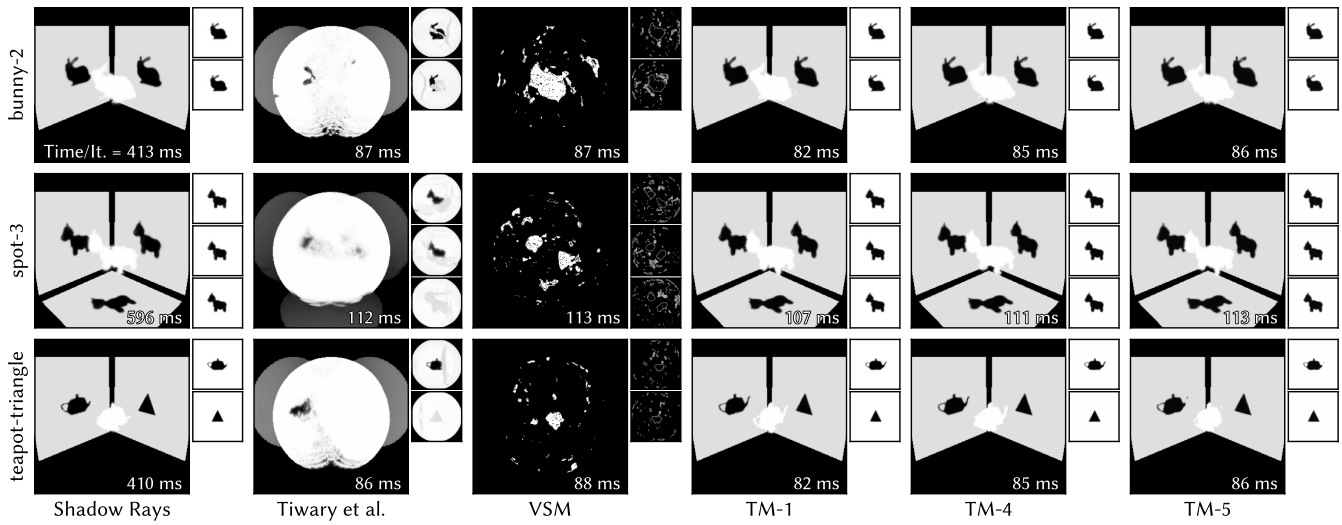


Fig. 15. Performing shadow art by using differentiable volume rendering. The shadow receiver planes are triangle meshes and a signed distance function, defined on a grid, is optimized such that its cast shadows resemble target shadows on the planes. Methods based on shadow mapping (by Tiwary et al. [2022] and Variance Shadow Mapping (VSM) [Donnelly and Lauritzen 2006; Worchel and Alexa 2023]), do not handle empty space because they are not volume-based. Using a moment-based approach to approximate the transmittance function (TM-n) yields results close to the shadow ray reference, with the same runtime.

dissolves and the implicit surface disappears, while transmittance mapping obtains a reasonable reconstruction of the target shape. The visibility gradients are substantial in this setting: with visibility detached from the AD graph, e.g. as in NVDIFFRECMC [Hasselgren et al. 2022, Sec.3.1, Shadow Gradients], the reconstruction fails.

We also want to highlight that this optimization, in each iteration, renders all views (each 190×256 pixels) using 128 samples while maintaining a performance of one iteration per second and reasonable memory consumption (Figure 16). Approaching this task with shadow rays would be hopeless without stochastic ray sampling.

7 CONCLUSION

We have shown that measures can be differentiably bounded using the theory of moments by proving that the lower and upper bounding functions arising from the truncated Hamburger moment problem are continuously differentiable in their parameters. As a consequence, the use of different approximate rendering techniques, which are based on these bounds, has been justified in the differentiable setting. We have shown that this contributes new approaches to differentiable shadow mapping and differentiable volumetric visibility, which are not only more principled than existing approaches but also more accurate and equally efficient.

Limitations and Future Work. The techniques proposed for differentiable shadow mapping and volumetric transmittance inherit the classical limitations of shadow mapping. The approach makes simplifying assumptions about the light sources such that area lights or image-based lighting are not supported. Also, the quality of the approximation depends on the spatial discretization. If the shadow/transmittance map is too coarse, small structures are missed; one can increase the resolution, but the memory requirements increase as well. Adaptive shadow mapping techniques might be applicable in this context [Fernando et al. 2001; Lefohn et al. 2007].

Numerical stability is a major challenge of moment-based bounds, especially for higher orders, as the involved matrices become increasingly ill-conditioned and robust polynomial root finding more difficult. We have shown that implementations must be designed around these constraints but our analysis is certainly not exhaustive. For the forward direction, numerical stability has mainly been investigated for quantization [Peters 2017; Peters and Klein 2015] and similar exploration could be useful for the backward direction.

Lastly, we have not yet explored applications that show the full potential of higher-order bounds: the transmittance mapping framework can be used for differentiable order-independent transparency based on the work by Münstermann et al. [2018], for which there seems to be a demand, e.g. in the context of Gaussian splatting [Hahlbohm et al. 2024; Keselman and Hebert 2023].

ACKNOWLEDGMENTS

We would like to thank the anonymous reviewers for their incredibly helpful feedback and suggestions. We also thank those who assisted in the early stages of this project, namely Marcus Zepp for providing test data and Dimitrios Bogiokas for discussing proofs related to transmittance representations. We thank Delio Vicini for making their code for figures publicly available. This work was funded by the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme (Grant agreement No. 101055448, ERC Advanced Grand EMERGE).

We thank and acknowledge the authors of the 3D models appearing in this paper: The Stanford 3D Scanning Repository (BUNNY), Keenan Crane (SPOT), Frank ter Haar by AIM@SHAPE (KITTEN), Martin Newell (UTAH TEAPOT), Yughues (GREEK PILLAR), karlwirbelwind (SEATED LION), plaggy (FOAM FINGER), Jan Nikolai Nelles and Nora Al-Badri (NEFERTITI), Inigo Quilez (3D SDFs), Taha Arslan (SHELF).

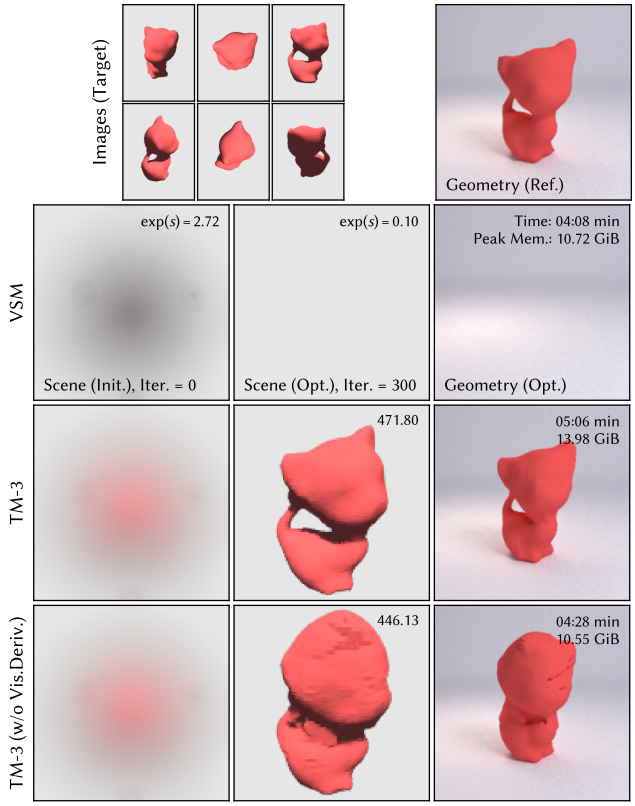


Fig. 16. Multi-view 3D reconstruction with differentiable volume rendering using approximate visibility. Shadow mapping (VSM) provides an inaccurate estimate of visibility, leading to divergence of the optimization. The principled moment-based technique (TM-3) succeeds, at slightly higher runtime and memory consumption. When visibility gradients are detached, the reconstruction fails (last row). Light and albedo are assumed to be known.

REFERENCES

N. I. Akhiezer. 1965. *The Classical Moment Problem and Some Related Questions in Analysis* (1 ed.). Society for Industrial and Applied Mathematics, Philadelphia, PA. arXiv:https://epubs.siam.org/doi/pdf/10.1137/1.9781611976397 https://epubs.siam.org/doi/abs/10.1137/1.9781611976397

N. I. Akhiezer and M. G. Kreĭn. 1962. *Some questions in the theory of moments*. Vol. 2. American Mathematical Soc.

Thomas Annen, Tom Mertens, Philippe Bekaert, Hans-Peter Seidel, and Jan Kautz. 2007. Convolution Shadow Maps. In *Proceedings of the 18th Eurographics Conference on Rendering Techniques* (Grenoble, France) *(EGSR'07)*. Eurographics Association, Goslar, DEU, 51–60.

Thomas Annen, Tom Mertens, Hans-Peter Seidel, Eddy Flerackers, and Jan Kautz. 2008. Exponential Shadow Maps. In *Proceedings of Graphics Interface 2008* (Windsor, Ontario, Canada) *(GI '08)*. Canadian Information Processing Society, CAN, 155–161.

Sai Praveen Bangaru, Lifan Wu, Tzu-Mao Li, Jacob Munkberg, Gilbert Bernstein, Jonathan Ragan-Kelley, Frédo Durand, Aaron Lefohn, and Yong He. 2023. SLANG.D: Fast, Modular and Differentiable Shader Programming. *ACM Trans. Graph.* 42, 6, Article 264 (dec 2023), 28 pages.

Åke Björck and Victor Pereyra. 1970. Solution of Vandermonde systems of equations. *Mathematics of computation* 24, 112 (1970), 893–903.

Cyril Delalandre, Pascal Gautron, Jean-Eudes Marvie, and Guillaume François. 2011. Transmittance function mapping *(I3D '11)*. Association for Computing Machinery, New York, NY, USA, 31–38.

Thomas Deliot, Eric Heitz, and Laurent Belcour. 2024. Transforming a Non-Differentiable Rasterizer into a Differentiable One with Stochastic Gradient Estimation. *Proc. ACM Comput. Graph. Interact. Tech.* 7, 1, Article 3 (may 2024), 16 pages.

Miles Detrixhe, Frédéric Gibou, and Chohong Min. 2013. A parallel fast sweeping method for the Eikonal equation. *J. Comput. Phys.* 237 (2013), 46–55.

William Donnelly and Andrew Lauritzen. 2006. Variance Shadow Maps. In *Proceedings of the 2006 Symposium on Interactive 3D Graphics and Games* (Redwood City, California) *(I3D '06)*. Association for Computing Machinery, New York, NY, USA, 161–165.

Randima Fernando, Sebastian Fernandez, Kavita Bala, and Donald P. Greenberg. 2001. Adaptive Shadow Maps. In *Proceedings of the 28th Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH '01)*. Association for Computing Machinery, New York, NY, USA, 387–390.

Géza Freud. 1969. *Orthogonale Polynome*. Birkhäuser Basel, Basel.

Jun Gao, Tianchang Shen, Zian Wang, Wenzheng Chen, Kangxue Yin, Daiqing Li, Or Litany, Zan Gojcic, and Sanja Fidler. 2022. GET3D: A Generative Model of High Quality 3D Textured Shapes Learned from Images. In *Advances In Neural Information Processing Systems*.

Florian Hahlbohm, Fabian Friederichs, Tim Weyrich, Linus Franke, Moritz Kappel, Susana Castillo, Marc Stamminger, Martin Eisemann, and Marcus Magnor. 2024. Efficient Perspective-Correct 3D Gaussian Splatting Using Hybrid Transparency. *arXiv preprint arXiv:2410.08129* (2024).

Charles R. Harris, K. Jarrod Millman, Stéfan J. van der Walt, Ralf Gommers, Pauli Virtanen, David Cournapeau, Eric Wieser, Julian Taylor, Sebastian Berg, Nathaniel J. Smith, Robert Kern, Matti Picus, Stephan Hoyer, Marten H. van Kerkwijk, Matthew Brett, Allan Haldane, Jaime Fernández del Río, Mark Wiebe, Pearu Peterson, Pierre Gérard-Marchant, Kevin Sheppard, Tyler Reddy, Warren Weckesser, Hameer Abbasi, Christoph Gohlke, and Travis E. Oliphant. 2020. Array programming with NumPy. *Nature* 585, 7825 (Sept. 2020), 357–362.

Jon Hasselgren, Nikolai Hofmann, and Jacob Munkberg. 2022. Shape, Light, and Material Decomposition from Images using Monte Carlo Rendering and Denoising. *arXiv:2206.03380* (2022).

Nicholas J Higham. 1987. Error analysis of the Björck-Pereyra algorithms for solving Vandermonde systems. *Numer. Math.* 50 (1987), 613–632.

Wenzel Jakob. 2022. nanobind: tiny and efficient C++/Python bindings. https://github.com/wjakob/nanobind.

Wenzel Jakob, Sébastien Speierer, Nicolas Roussel, Merlin Nimier-David, Delio Vicini, Tizian Zeltner, Baptiste Nicolet, Miguel Crespo, Vincent Leroy, and Ziyi Zhang. 2022b. Mitsuba 3 renderer. https://mitsuba-renderer.org.

Wenzel Jakob, Sébastien Speierer, Nicolas Roussel, and Delio Vicini. 2022a. DR.JIT: A Just-in-Time Compiler for Differentiable Rendering. *ACM Trans. Graph.* 41, 4, Article 124 (jul 2022), 19 pages.

Jon Jansen and Louis Bavoil. 2010. Fourier opacity mapping. In *Proceedings of the 2010 ACM SIGGRAPH symposium on Interactive 3D Graphics and Games*. 165–172.

W. Kahan. 2004. On the cost of floating-point computation without extra-precise arithmetic. (2004).

James T. Kajiya and Brian P Von Herzen. 1984. Ray Tracing Volume Densities. In *Proceedings of the 11th Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH '84)*. Association for Computing Machinery, New York, NY, USA, 165–174.

Leonid Keselman and Martial Hebert. 2023. Flexible techniques for differentiable rendering with 3d gaussians. *arXiv preprint arXiv:2308.14737* (2023).

Diederik P. Kingma and Jimmy Ba. 2015. Adam: A Method for Stochastic Optimization. In *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, Yoshua Bengio and Yann LeCun (Eds.). http://arxiv.org/abs/1412.6980

M. G. Kreĭn and A. A. Nudel'man. 1977. *The Markov Moment Problem and Extremal Problems*. Translations of Mathematical Monographs, Vol. 50. American Mathematical Society.

Samuli Laine, Janne Hellsten, Tero Karras, Yeongho Seol, Jaakko Lehtinen, and Timo Aila. 2020. Modular Primitives for High-Performance Differentiable Rendering. *ACM Trans. Graph.* 39, 6, Article 194 (nov 2020), 14 pages.

Aaron E. Lefohn, Shubhabrata Sengupta, and John D. Owens. 2007. Resolution-Matched Shadow Maps. *ACM Trans. Graph.* 26, 4 (oct 2007), 20–es.

Tzu-Mao Li, Miika Aittala, Frédo Durand, and Jaakko Lehtinen. 2018. Differentiable Monte Carlo Ray Tracing through Edge Sampling. *ACM Trans. Graph.* 37, 6, Article 222 (dec 2018), 11 pages.

Zhaoshuo Li, Thomas Müller, Alex Evans, Russell H Taylor, Mathias Unberath, Ming-Yu Liu, and Chen-Hsuan Lin. 2023. Neuralangelo: High-Fidelity Neural Surface Reconstruction. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Ruofan Liang, Jiahao Zhang, Haoda Li, Chen Yang, Yushi Guan, and Nandita Vijaykumar. 2022. SPIDR: SDF-based Neural Point Fields for Illumination and Deformation. *arXiv preprint arXiv:2210.08398* (2022).

Selena Ling, Nicholas Sharp, and Alec Jacobson. 2022. VectorAdam for rotation equivariant geometry optimization. In *Proceedings of the 36th International Conference on Neural Information Processing Systems* (New Orleans, LA, USA) *(NIPS '22)*. Curran Associates Inc., Red Hook, NY, USA, Article 297, 12 pages.

Ruoshi Liu, Sachit Menon, Chengzhi Mao, Dennis Park, Simon Stent, and Carl Vondrick. 2023. What You Can Reconstruct From a Shadow. In *Proceedings of the IEEE/CVF*

Conference on Computer Vision and Pattern Recognition (CVPR). 17059–17068.

Shichen Liu, Tianye Li, Weikai Chen, and Hao Li. 2019. Soft Rasterizer: A Differentiable Renderer for Image-based 3D Reasoning. *The IEEE International Conference on Computer Vision (ICCV)* (Oct 2019).

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., USA, 385–392.

Fujun Luan, Shuang Zhao, Kavita Bala, and Zhao Dong. 2021. Unified Shape and SVBRDF Recovery using Differentiable Monte Carlo Rendering. *Computer Graphics Forum* (2021).

Ben Mildenhall, Pratul P. Srinivasan, Matthew Tancik, Jonathan T. Barron, Ravi Ramamoorthi, and Ren Ng. 2020. NeRF: Representing Scenes as Neural Radiance Fields for View Synthesis. In *ECCV*.

Niloy J. Mitra and Mark Pauly. 2009. Shadow Art. *ACM Trans. Graph.* 28, 5 (dec 2009), 1–7.

Jacob Munkberg, Wenzheng Chen, Jon Hasselgren, Alex Evans, Tianchang Shen, Thomas Müller, Jun Gao, and Sanja Fidler. 2022. Extracting Triangular 3D Models, Materials, and Lighting From Images. In *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 8270–8280.

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 (jul 2018), 20 pages.

Baptiste Nicolet, Alec Jacobson, and Wenzel Jakob. 2021. Large Steps in Inverse Rendering of Geometry. *ACM Trans. Graph.* 40, 6, Article 248 (dec 2021), 13 pages.

Merlin Nimier-David, Sébastien Speierer, Benoît Ruiz, and Wenzel Jakob. 2020. Radiative Backpropagation: An Adjoint Method for Lightning-Fast Differentiable Rendering. *ACM Trans. Graph.* 39, 4, Article 146 (aug 2020), 15 pages.

Merlin Nimier-David, Delio Vicini, Tizian Zeltner, and Wenzel Jakob. 2019. Mitsuba 2: A Retargetable Forward and Inverse Renderer. *ACM Trans. Graph.* 38, 6, Article 203 (nov 2019), 17 pages.

Michael Oechsle, Songyou Peng, and Andreas Geiger. 2021. UNISURF: Unifying Neural Implicit Surfaces and Radiance Fields for Multi-View Reconstruction. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*. 5589–5599.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Köpf, Edward Yang, Zach DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. 2019. PyTorch: An Imperative Style, High-Performance Deep Learning Library. In *Proceedings of the 33rd International Conference on Neural Information Processing Systems*. Curran Associates Inc., Red Hook, NY, USA, Article 721, 12 pages.

Christoph Peters. 2017. Non-linearly quantized moment shadow maps. In *Proceedings of High Performance Graphics* (Los Angeles, California) *(HPG '17)*. Association for Computing Machinery, New York, NY, USA, Article 15, 11 pages.

Christoph Peters. 2023. Finding Real Polynomial Roots on GPUs. https://momentsingraphics.de/GPUPolynomialRoots.html.

Christoph Peters and Reinhard Klein. 2015. Moment Shadow Mapping. In *Proceedings of the 19th Symposium on Interactive 3D Graphics and Games* (San Francisco, California) *(i3D '15)*. Association for Computing Machinery, New York, NY, USA, 7–14.

Christoph Peters, Sebastian Merzbach, Johannes Hanika, and Carsten Dachsbacher. 2019. Using moments to represent bounded signals for spectral rendering. *ACM Trans. Graph.* 38, 4, Article 136 (jul 2019), 14 pages.

Christoph Peters, Cedrick Munstermann, Nico Wetzstein, and Reinhard Klein. 2016. Beyond hard shadows: moment shadow maps for single scattering, soft shadows and translucent occluders. In *Proceedings of the 20th ACM SIGGRAPH Symposium on Interactive 3D Graphics and Games* (Redmond, Washington) *(I3D '16)*. Association for Computing Machinery, New York, NY, USA, 159–170.

Stanislav Pidhorskyi, Tomas Simon, Gabriel Schwartz, He Wen, Yaser Sheikh, and Jason Saragih. 2025. Rasterized Edge Gradients: Handling Discontinuities Differentiably. In *Computer Vision – ECCV 2024*, Aleš Leonardis, Elisa Ricci, Stefan Roth, Olga Russakovsky, Torsten Sattler, and Gül Varol (Eds.). Springer Nature Switzerland, Cham, 335–352.

Wentian Qu, Zhaopeng Cui, Yinda Zhang, Chenyu Meng, Cuixia Ma, Xiaoming Deng, and Hongan Wang. 2023. Novel-View Synthesis and Pose Estimation for Hand-Object Interaction from Sparse Views. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*. 15100–15111.

William T. Reeves, David H. Salesin, and Robert L. Cook. 1987. Rendering Antialiased Shadows with Depth Maps. *SIGGRAPH Comput. Graph.* 21, 4 (aug 1987), 283–291.

Frédéric Riesz. 1911. Sur certains systèmes singuliers d'équations intégrales. *Annales scientifiques de l'École Normale Supérieure* 28 (1911), 33–62. http://eudml.org/doc/81305

M Salvi. 2008. Probabilistic approaches to shadow maps filtering, February. In *A talk in the tutorial" Core Techniques and Algorithms in Shader Programming" at Game Developers Conference*.

Steven A Shafer and Takeo Kanade. 1983. Using shadows in finding surface orientations. *Computer Vision, Graphics, and Image Processing* 22, 1 (1983), 145–176. https://www.sciencedirect.com/science/article/pii/0734189X83900993

Brian Sharpe. 2018. Moment transparency. In *Proceedings of the Conference on High-Performance Graphics* (Vancouver, British Columbia, Canada) *(HPG '18)*. Association for Computing Machinery, New York, NY, USA, Article 8, 4 pages.

Gilbert Strang. 2007. *Computational Science and Engineering*. Wellesley-Cambridge Press, Philadelphia, PA. arXiv:https://epubs.siam.org/doi/pdf/10.1137/1.9780961408817 https://epubs.siam.org/doi/abs/10.1137/1.9780961408817

G. Szegő. 1975. *Orthogonal Polynomials*. American Mathematical Society.

Árpád Tari. 2005. *Moments based bounds in stochastic models*. Ph.D. dissertation. Budapest University of Technology and Economics.

Árpád Tari, Miklós Telek, and Peter Buchholz. 2005. A Unified Approach to the Moments Based Distribution Estimation – Unbounded Support. In *Formal Techniques for Computer Systems and Business Processes*, Mario Bravetti, Leïla Kloul, and Gianluigi Zavattaro (Eds.). Springer Berlin Heidelberg, Berlin, Heidelberg, 79–93.

Kushagra Tiwary, Tzofi Klinghoffer, and Ramesh Raskar. 2022. Towards Learning Neural Representations from Shadows. In *Computer Vision – ECCV 2022*, Shai Avidan, Gabriel Brostow, Moustapha Cissé, Giovanni Maria Farinella, and Tal Hassner (Eds.). Springer Nature Switzerland, Cham, 300–316.

Delio Vicini, Sébastien Speierer, and Wenzel Jakob. 2021. Path Replay Backpropagation: Differentiating Light Paths Using Constant Memory and Linear Time. *ACM Trans. Graph.* 40, 4, Article 108 (jul 2021), 14 pages.

Delio Vicini, Sébastien Speierer, and Wenzel Jakob. 2022. Differentiable Signed Distance Function Rendering. *ACM Trans. Graph.* 41, 4, Article 125 (jul 2022), 18 pages.

Peng Wang, Lingjie Liu, Yuan Liu, Christian Theobalt, Taku Komura, and Wenping Wang. 2021. NeuS: Learning Neural Implicit Surfaces by Volume Rendering for Multi-view Reconstruction. *NeurIPS* (2021).

Lance Williams. 1978. Casting Curved Shadows on Curved Surfaces. In *Proceedings of the 5th Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH '78)*. Association for Computing Machinery, New York, NY, USA, 270–274.

Markus Worchel and Marc Alexa. 2023. Differentiable Shadow Mapping for Efficient Inverse Graphics. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 142–153.

Markus Worchel, Rodrigo Diaz, Weiwen Hu, Oliver Schreer, Ingo Feldmann, and Peter Eisert. 2022. Multi-View Mesh Reconstruction With Neural Deferred Shading. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 6187–6197.

Wenqi Yang, Guanying Chen, Chaofeng Chen, Zhenfang Chen, and Kwan-Yee K. Wong. 2022. S^3-NeRF: Neural Reflectance Field from Shading and Shadow under a Single Viewpoint. In *Conference on Neural Information Processing Systems (NeurIPS)*.

Lior Yariv, Jiatao Gu, Yoni Kasten, and Yaron Lipman. 2021. Volume rendering of neural implicit surfaces. In *Thirty-Fifth Conference on Neural Information Processing Systems*.

Lior Yariv, Yoni Kasten, Dror Moran, Meirav Galun, Matan Atzmon, Basri Ronen, and Yaron Lipman. 2020. Multiview Neural Surface Reconstruction by Disentangling Geometry and Appearance. *Advances in Neural Information Processing Systems* 33 (2020).

Cem Yuksel. 2022. High-performance polynomial root finding for graphics. *Proceedings of the ACM on Computer Graphics and Interactive Techniques* 5, 3 (2022), 1–15.

Cheng Zhang, Bailey Miller, Kai Yan, Ioannis Gkioulekas, and Shuang Zhao. 2020. Path-Space Differentiable Rendering. *ACM Trans. Graph.* 39, 4, Article 143 (aug 2020), 19 pages.

Kai Zhang, Fujun Luan, Zhengqi Li, and Noah Snavely. 2022. IRON: Inverse Rendering by Optimizing Neural SDFs and Materials From Photometric Images. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 5565–5574.

Kai Zhang, Fujun Luan, Qianqian Wang, Kavita Bala, and Noah Snavely. 2021. PhySG: Inverse Rendering with Spherical Gaussians for Physics-based Material Editing and Relighting. In *The IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*.

Ziyi Zhang, Nicolas Roussel, and Wenzel Jakob. 2023. Projective Sampling for Differentiable Rendering of Geometry. *Transactions on Graphics (Proceedings of SIGGRAPH Asia)* 42, 6 (Dec. 2023).

Table 1. Accuracy of the implementations, measured as average over 1 Million random samples $\{(\mathbf{m}_k, \eta_k)\}$.

		32 bits	64 bits		
		$\Delta\mathbf{m}$	$\Delta\mathbf{m}$	$\Delta\frac{\partial B}{\partial \mathbf{m}}$	$\Delta\frac{\partial B}{\partial \eta}$
n = 1	PyTorch	1.70e-07	3.26e-16	3.54e-07	2.20e-07
	C++	1.68e-07	3.20e-16	3.54e-07	2.20e-07
2	PyTorch	1.53e-05	2.58e-14	2.27e-06	8.37e-07
	C++	3.14e-06	5.93e-15	2.27e-06	8.37e-07
3	PyTorch	7.07e-04	1.47e-12	1.90e-05	3.12e-06
	C++	2.51e-04	4.80e-13	1.90e-05	3.12e-06
4	PyTorch	1.56e-01	2.63e-10	3.46e-04	9.08e-06
	C++	3.96e-02	1.00e-10	3.46e-04	9.08e-06
5	PyTorch	–	1.10e-08	5.79e-03	2.82e-05
	C++	–	2.86e-09	5.79e-03	2.82e-05

A ACCURACY VERIFICATION RESULTS

Table 1 reports the results from the accuracy verification (Section 5.2, Paragraph *Accuracy Verification*), which has the purpose of empirically validating the accuracy of the (custom) C++/CUDA implementation (not establishing a ranking over accuracy). From 1 Million random samples, ones with a singular Hankel matrix or η near a singularity are omitted. For a positive moment vector $\mathbf{m}_k$ and a given η_k, $\mathbf{m}_k$ can be reconstructed from the roots and weights computed by the primal procedure (assuming no biasing). $\Delta\mathbf{m}$ is the Euclidean distance between a moment vector and its reconstruction $\|\mathbf{m}_k - \sum_{i=0}^{n} w_i \mathbf{u}(x_i)\|$, where $\mathbf{u}(x) = (1, x, x^2, ..., x^{2n})^\top$ evaluates the moment curve (c.f. Theorem 1). With larger n, the moment reconstruction becomes less meaningful ($n = 5$ involves 10th power), so we omit the measurement for 32-bit precision. $\Delta\frac{\partial B}{\partial \mathbf{m}}$ and $\Delta\frac{\partial B}{\partial \eta}$ are the mean absolute error between finite differences and the computed bound derivative w.r.t. the moments and η, respectively.

B TRANSMITTANCE AS OPACITY

Differentiable volume rendering frameworks in vision are often designed in a way that allows simple evaluation of expressions

$$\int_0^\infty \mathbf{v}\big(\mathbf{r}(t)\big) \underbrace{T(t)\,\sigma\big(\mathbf{r}(t)\big)\,\mathrm{d}t}_{w\left(\mathbf{r}(t)\right)}, \tag{40}$$

where $\mathbf{v} : \mathbb{R}^3 \to \mathbb{R}^d$ is some field (e.g. color or more general features) [Li et al. 2023; Mildenhall et al. 2020; Wang et al. 2021]. The weights are pre-computed such that the numerical integration is

$$\frac{1}{N}\sum_{i=1}^{N} \mathbf{v}_i w_i \tag{41}$$

Our differentiable volume shadows based on transmittance estimates can be trivially integrated into such frameworks, which is clear if one defines transmittance in terms of opacity

$$T(t) = 1 - \underbrace{\int_0^t T(z)\,\sigma(\mathbf{r}(z))\,\mathrm{d}z}_{\text{opacity}} \tag{42}$$

As an alternative to bounding the absorbance, we bound the opacity, which is also a non-decreasing measure. The moments can be simply computed with the weights and the N distance samples $\{t_i\}$

$$\mathbf{m} = \frac{1}{N}\sum_{i=1}^{N} \mathbf{u}(t_i) w_i \tag{43}$$

This is equivalent to the approach by Peters et al. [2016].

C PROOFS

This section first introduces useful definitions and theorems regarding orthogonal polynomials (from the literature) and then continues with our proofs.

C.1 Orthogonal Polynomials

Proposition 15 (e.g. [Akhiezer and Kreĭn 1962, p.4]). *If* $\mathbf{m} = (m_0, \ldots, m_{2n})$ *is strictly positive, the polynomials of the sequence*

$$P_0(x;\mathbf{m}) = 1$$

$$P_k(x;\mathbf{m}) = \det\begin{bmatrix} m_0 & m_1 & \dots & m_k \\ m_1 & m_2 & & m_{k+1} \\ \vdots & & \ddots & \vdots \\ m_{k-1} & m_k & \dots & m_{2k-1} \\ 1 & x & \dots & x^k \end{bmatrix}$$

are pairwise orthogonal, i.e., $\mathfrak{S}_{\mathbf{m}}\{P_i P_j\} = 0$ *for* $i \neq j$ *(c.f. Eq. (9)) under any measure* μ *that is a representation of the moments* $\mathbf{m}$.

We recall two important facts about orthogonal polynomials

(1) An orthogonal polynomial P_k has exactly k roots, all of which are real and simple (see, e.g., [Szegő 1975, Theorem 3.3.1, p.44]).
(2) Two different polynomials $P_i, P_j, i \neq j$ do not vanish at the same time, i.e., they have distinct roots. If $i = j + 1$ then the roots are interlaced

$$z_1 < y_1 < \cdots < z_j < y_j < z_i,$$

where $y_1, \ldots, y_j$ are the roots of P_j and $z_1, \ldots, z_i$ are the roots of P_i [Szegő 1975, Theorem 3.3.2, p.46]

The $\{P_k\}$ lead to the polynomial [Freud 1969, p.21]

$$\psi_{n+1}(x;\eta,\mathbf{m}) = P_{n+1}(x;\mathbf{m})P_n(\eta;\mathbf{m}) - P_{n+1}(\eta;\mathbf{m})P_n(x;\mathbf{m}). \tag{44}$$

If η is a root of P_n, ψ_{n+1} has degree n: P_{n+1} cannot vanish at the same time because it's orthogonal to P_n and P_n has degree n by definition. If η is not a root of P_n, ψ_{n+1} has degree $n + 1$. In either case, the following result holds:

Theorem 16 ([Akhiezer 1965, Theorem 1.2.2, p.10]). *All roots of* $\psi_{n+1}(x;\eta,\mathbf{m})$ *are real and simple.*

The roots of ψ_{n+1} are related to the roots of P_n:

Theorem 17 ([Tari 2005, Theorem A.39]). *If η is not a root of P_n, the roots $x_0, \dots, x_n$ of ψ_{n+1} and the roots $y_1, \dots, y_n$ of P_n interlace*

$$x_0 < y_1 < x_1 < \cdots < x_{n-1} < y_n < x_n.$$

and the significance of ψ_{n+1} is its relation to the kernel K:

Theorem 18 (Christoffel–Darboux theorem, [Szegő 1975, Theorem 3.2.2., p.43]). *The kernel K and ψ_{n+1} are related by*

$$K(x; \eta, \mathbf{m}) \propto \frac{\psi_{n+1}(x; \eta, \mathbf{m})}{(x - \eta)}.$$

Which yields the following characterization of the roots of K:

Corollary 19. *The kernel polynomial K has the same roots as ψ_{n+1}, except for η, thus all $\{x_i\}$ are real and simple. Either η is not a root of P_n, then ψ_{n+1} has $n+1$ real and simple roots which are the points $x_1, \dots, x_n$ and η. Or η is a root of P_n, then ψ_{n+1} has the root η and $n-1$ other roots $x_1, \dots, x_{n-1}$.*

C.2 Intermediate Statements

For the main results, some properties regarding differentiability are required. These are proven here.

Lemma 20. *The Hankel matrix $\mathbf{H}$ is continuously differentiable in the moments and, if the moments are strictly positive, its inverse $\mathbf{H}^{-1}$ is also continuously differentiable.*

Proof. The moments are the entries of the Hankel matrix, so the first claim is trivially true. For the second claim, we have $\mathbf{H}^{-1} = \frac{1}{\det(\mathbf{H})} \operatorname{adj}(\mathbf{H})$. The entries of the adjugate $\operatorname{adj}(\mathbf{H})$ and the determinant $\det(\mathbf{H})$ are polynomials in the entries of $\mathbf{H}$. By Theorem 4 from strict positivity follows $\det(\mathbf{H}) > 0$, so the second claim holds. □

The orthogonal polynomials depend smoothly on the moments:

Lemma 21. *The coefficients of the orthogonal polynomials $P_k(x; \mathbf{m})$ are infinitely differentiable in the moments $\mathbf{m}$.*

Proof. Expanding along the last row of the determinant form given in Definition 15, we get

$$P_k(x; \mathbf{m}) = \sum_{j=0}^{n} (-1)^{k+j} x^j \det(\mathbf{M}_{k,j}), \tag{45}$$

where $\det(\mathbf{M}_{k,j})$ is the (k, j) minor of the original matrix. The result is the canonical form of the orthogonal polynomial P_k with its coefficients being the determinants of matrices $\mathbf{M}_{k,j}$ formed from the set of moments. Since the determinant is a polynomial of the matrix coefficients, the minors are infinitely differentiable in the moments and so is P_k. □

It is well known that, if the roots of a polynomial are real and simple, they are continuously differentiable functions of the coefficients. For orthogonal polynomials, the roots are real and simple, which gives the following result:

Corollary 22. *The n roots of the orthogonal polynomials $P_k(x; \mathbf{m})$ are continuously differentiable functions in the moments.*

These observations extend to the roots of ψ_{n+1}:

Lemma 23. *If $P_n(\eta; \mathbf{m}) \neq 0$, the $n+1$ roots of $\psi_{n+1}(x; \eta, \mathbf{m})$ are continuously differentiable functions in η and $\mathbf{m}$. If $P_n(\eta; \mathbf{m}) = 0$, the n roots of $\psi_{n+1}(x; \eta, \mathbf{m})$ are continuously differentiable in η and $\mathbf{m}$.*

Proof. Recall that $\psi_{n+1}(x; \eta) = P_{n+1}(x)P_n(\eta) - P_{n+1}(\eta)P_n(x)$ (Eq. 44). Lemma 21 states that P_n and P_{n+1} are infinitely differentiable in η and $\mathbf{m}$, and consequently, so are the coefficients of ψ_{n+1}. Therefore all real and simple roots of ψ_{n+1} are continuously differentiable in η and $\mathbf{m}$. □

C.3 Proof of Theorem 5

Proof. Without loss of generality, we consider the lower bound $L(\eta; \mathbf{m})$. Choose any point $(\eta_0, \mathbf{m}_0) \in \Omega$. According to Theorem 2, a unique canonical representation exists with points $\{x_i(\eta_0; \mathbf{m}_0)\}$ and η_0 where the points can be ordered accordingly

$$x_0 < \cdots < x_{k-1} < x_k (= \eta_0) < x_{k+1} < \cdots < x_n. \tag{46}$$

Take another point $(\eta_1, \mathbf{m}_1) \in \Omega$ in the domain. Without loss of generality, assume that at $(\eta_1, \mathbf{m}_1)$ more of the $\{x_i\}$ are larger than η_1, i.e., the number of terms in $L(\eta_0; \mathbf{m}_0)$ is greater than in $L(\eta_1; \mathbf{m}_1)$.

The moments $\mathbf{m}_0$ and $\mathbf{m}_1$ are in the convex cone of the moment curve and η_0 and η_1 are in $\mathbb{R}$, so the points can be connected with an arbitrary, continuously differentiable curve

$$\boldsymbol{\gamma}(\tau) = \big(\eta(\tau), \mathbf{m}(\tau)\big) \in \mathbb{R} \times \mathbb{R}^{2n+1}, \tag{47}$$

with $\boldsymbol{\gamma}(0) = (\eta_0, \mathbf{m}_0)$ and $\boldsymbol{\gamma}(1) = (\eta_1, \mathbf{m}_1)$.

Let us assume the opposite of the claim: for all $\tau \in (0, 1)$ the curve is part of the domain, i.e., $\boldsymbol{\gamma}(\tau) \in \Omega$, so that at no point is $\eta(\tau)$ a root of P_n and the moments $\mathbf{m}(\tau)$ are strictly positive.

The polynomial $\psi_{n+1}\big(x; \eta(\tau), \mathbf{m}(\tau)\big)$, therefore, has $n+1$ real and simple roots along the curve, which are, according to Corollary 19, the $x_0, \dots, x_n$, including η. The order of roots at $(\eta_0, \mathbf{m}_0)$ identifies each root uniquely and we can define a function

$$g_i(\tau) = x_i\big(\eta(\tau); \mathbf{m}(\tau)\big) - \eta(\tau), \tag{48}$$

which indicates if root x_i is greater or smaller than η. According to Lemma 23, the roots are continuously differentiable functions for $\tau \in [0, 1]$ and so are the functions g_i.

The number of points greater than η is larger at η_1 than at η_0, so there is at least one $x_j \neq \eta_0$ such that

$$g_j(0) < 0 \quad \text{and} \quad g_j(1) > 0, \tag{49}$$

i.e., x_j "switches" places with η on the path. Since $\boldsymbol{\gamma}$ is continuous and g_i is continuous, according to the intermediate value theorem, there exists one $\tau' \in (0, 1)$ where

$$g_j(\tau') = 0, \tag{50}$$

which however is a contradiction because along the curve the $n+1$ roots of ψ_{n+1} are real and simple. Therefore, at some point along $\boldsymbol{\gamma}$, either the sequence of moments $\mathbf{m}(\tau)$ is not strictly positive or $\eta(\tau)$ is a root of $P_n\big(x; \mathbf{m}(\tau)\big)$. □

C.4 Proof of Theorem 7

Proof. Recall that, according to Equation 17, the weights are

$$w_i = \frac{1}{\mathbf{x}_i^\top \mathbf{H}^{-1} \mathbf{x}_i}.$$

The vectors $\mathbf{x}_i$ consist of monomials in the roots x_i (Equation 14) and the roots are continuously differentiable (Lemma 23). $\mathbf{H}^{-1}$ is continuously differentiable (Lemma 20). Since $\mathbf{m}$ is strictly positive, $\mathbf{H}$ is positive definite and the denominator is > 0. The claim now follows from the properties of function composition and differentiability of matrix multiplication. □

C.5 Proof of Theorem 9

Proof. Without loss of generality, assume $(\eta_0, \mathbf{m}_0) \in \Omega$ such that $x_j(=\eta_0) > x_0$ approaches y_j, as the other case can be treated equally. The roots $\{x_i(\eta, \mathbf{m})\}_{i=0}^{n}$ of the polynomial ψ_{n+1} (Corollary 19) and the roots $\{y_i(\mathbf{m})\}_{i=1}^{n}$ of the polynomial P_n strictly interlace at $(\eta_0, \mathbf{m}_0)$ (Theorem 17).

Let $(\eta_1, \mathbf{m}_1) \in \Lambda$ be a point on the singularity, such that $P_n(\eta_1, \mathbf{m}_1) = 0$. The roots of P_n are continuously differentiable functions in $(\eta, \mathbf{m})$ (Lemma 23), so the order established at $(\eta_0, \mathbf{m}_0)$ is maintained along a continuously differentiable path from $(\eta_0, \mathbf{m}_0)$ to $(\eta_1, \mathbf{m}_1)$. Each y_j can be uniquely identified from which follows that $y_j(\mathbf{m}_0) < x_j(\eta_0, \mathbf{m}_0)(=\eta_0) < y_{j+1}(\mathbf{m}_0)$ and $y_j(\mathbf{m}_1) = \eta_1$.

At $(\eta_1, \mathbf{m}_1) \in \Lambda$, the roots of ψ_{n+1} are the n roots of P_n (see Eq. (44)). The roots of ψ_{n+1} are continuously differentiable functions of η and $\mathbf{m}$ (Lemma 23), and in particular they are at $(\eta_1, \mathbf{m}_1)$. Therefore, if the path is reversed, it is a continuously differentiable path from the roots of $P_n(\mathbf{m}_1)$ to n of the roots of $\psi_{n+1}(x; \eta_0, \mathbf{m}_0)$.

By construction, $x_j(=\eta_0)$ follows a continuously differentiable path to $y_j(=\eta_1)$, so its reverse is a continuously differentiable path from y_j to x_j. If y_{j+1} exists, it must correspond to the root x_{j+1}: it cannot correspond to any root $\leq x_j$ as the roots of ψ_{n+1} are simple on the path (Lemma 16) and it cannot correspond to any root $\geq x_{j+2}$ because the x_i and y_i strictly interlace on the path and there is $x_{j+1} < y_{j+2} < x_{j+2}$. A similar argument holds for y_{j-1} if it exists. This establishes an order such that every y_j at $(\eta_1, \mathbf{m}_1)$ corresponds to x_j at $(\eta_0, \mathbf{m}_0)$.

Therefore, the only root without a corresponding y at $(\eta_1, \mathbf{m}_1)$ is x_0. ψ_{n+1} is a degree $n+1$ polynomial that is reduced to a degree n polynomial on a continuously differentiable path from $(\eta_0, \mathbf{m}_0)$ to $(\eta_1, \mathbf{m}_1)$, so its leading coefficient a_{n+1} is taken to zero on the path. As a_{n+1} vanishes, the roots $x_1 < \cdots < x_n$ remain finite, and therefore the root x_0 is unbounded and must go to $-\infty$. □

C.6 Proof of Lemma 12

Proof. Recall the definition of the weights $w_k = \frac{1}{\mathbf{x}_k^\top \mathbf{H}^{-1} \mathbf{x}_k}$. Since the moment-generating measures are finite, the moments are finite, and therefore, there is an upper bound on the eigenvalues of $\mathbf{H}$. Consequently, there is a lower bound on the eigenvalues of $\mathbf{H}^{-1}$. $\mathbf{H}$ is positive-definite for strictly positive moments (Theorem 4) and so is $\mathbf{H}^{-1}$. As $x_k \to \pm\infty$, the norm grows $\|\mathbf{x}_k\|_2 \to \infty$, from which follows that $\mathbf{x}_k^\top \mathbf{H}^{-1} \mathbf{x}_k \to \infty$ and $w_k \to 0$. □

C.7 Proof of Lemma 13

Proof. We first prove the case for the derivative $\frac{\partial w_k}{\partial \eta}$, which by the chain rule is

$$\frac{\partial w_k}{\partial \eta} = \frac{\partial w_k}{\partial \mathbf{x}_k} \frac{\partial \mathbf{x}_k}{\partial x_k} \frac{\partial x_k}{\partial \eta}, \tag{51}$$

since $\mathbf{H}^{-1}$ is independent of η. The point x_k is a root of the kernel polynomial K, which is explicitly given as (Eq. (16))

$$K(x) = \mathbf{x}^\top \mathbf{H}^{-1} \boldsymbol{\eta}. \tag{52}$$

The implicit function theorem relates η to the root

$$\frac{\partial x_k}{\partial \eta} = -\frac{\mathbf{x}_k^\top \mathbf{H}^{-1} \frac{\partial \boldsymbol{\eta}}{\partial \eta}}{\boldsymbol{\eta}^\top \mathbf{H}^{-1} \frac{\partial \mathbf{x}_k}{\partial x_k}}. \tag{53}$$

The other factor can be directly expressed using Eq. (17)

$$\frac{\partial w_k}{\partial \mathbf{x}_k} = -2 \frac{\mathbf{x}_k^\top \mathbf{H}^{-1}}{\left(\mathbf{x}_k^\top \mathbf{H}^{-1} \mathbf{x}_k\right)^2}. \tag{54}$$

The complete derivative therefore is

$$\frac{\partial w_k}{\partial \eta} = 2 \frac{\mathbf{x}_k^\top \mathbf{H}^{-1} \frac{\partial \mathbf{x}_k}{\partial x_k} \mathbf{x}_k^\top \mathbf{H}^{-1} \frac{\partial \boldsymbol{\eta}}{\partial \eta}}{\left(\mathbf{x}_k^\top \mathbf{H}^{-1} \mathbf{x}_k\right)^2 \boldsymbol{\eta}^\top \mathbf{H}^{-1} \frac{\partial \mathbf{x}_k}{\partial x_k}} \tag{55}$$

The derivative is a rational function in x_k and the limit is determined through the (ratio of) degrees of the numerator and the denominator. We assume full degree in the numerator $(3n-1)$.

Notice that the point $(\eta, \mathbf{m})$ is bounded and finite, and consequently so are $\mathbf{H}^{-1}$, $\boldsymbol{\eta}$, and $\frac{\partial \boldsymbol{\eta}}{\partial \eta}$. The quadratic part in the denominator is positive because the eigenvalues of $\mathbf{H}^{-1}$ are bounded from below ($\mathbf{m}$ is strictly positive) and the quadratic part necessarily has full degree $(4n)$ because its leading coefficient is the leading coefficient of full-degree P_n (the last row/column of $\mathbf{H}^{-1}$ are the coefficients of P_n [Tari 2005, Lemma A.26]). The second part of the denominator, $K'(x_k) = \boldsymbol{\eta}^\top \mathbf{H}^{-1} \frac{\partial \mathbf{x}_k}{\partial x_k}$, is the derivative of the kernel at its root x_k. The kernel roots are real and simple inside Ω, so the derivative is non-zero in the domain. At a singularity, the kernel has degree $n-1$ and the derivative reduces to degree $n-2$, with non-zero coefficient c_{n-1} (Corollary 19). For the case $n = 1$, the derivative *is* the vanishing coefficient c_1, which can be reparameterized in the root as $c_1 = -\frac{c_0}{x_k}$. Since $\mathbf{H}^{-1}$ has full rank and $\begin{bmatrix} c_0 & c_1 \end{bmatrix} = \boldsymbol{\eta}^\top \mathbf{H}^{-1}$, the coefficient c_0 cannot vanish at the same time as c_1. The denominator therefore has minimal degree $5n-2$.

We bound the derivative from above, assuming full degree in the numerator and minimal degree in the denominator:

$$\left| \frac{\partial w_k}{\partial \eta} \right| \leq c \left| \frac{x_k^{3n-1}}{x_k^{5n-2}} \right|, \quad c > 0. \tag{56}$$

The upper bound goes to 0 as $x_k \to \pm\infty$ and so does $\frac{\partial w_k}{\partial \eta}$. □

Proof. We continue with the derivative $\frac{\partial w_k}{\partial \mathbf{m}}$, and consider only a single moment m_i $(0 \leq i \leq 2n)$ without loss of generality. By the chain rule the derivative is

$$\frac{\partial w_k}{\partial m_i} = \left\langle \frac{\partial w_k}{\partial \mathbf{H}^{-1}}, \frac{\partial \mathbf{H}^{-1}}{\partial m_i} \right\rangle + \frac{\partial w_k}{\partial \mathbf{x}_k} \frac{\partial \mathbf{x}_k}{\partial x_k} \frac{\partial x_k}{\partial m_i}, \tag{57}$$

where $\langle \mathbf{A}, \mathbf{B} \rangle$ denotes the sum of the element-wise matrix product. Using the implicit function theorem for the kernel, we get

$$\frac{\partial x_k}{\partial m_i} = -\left\langle \frac{\mathbf{x}_k \boldsymbol{\eta}^\top}{\boldsymbol{\eta}^\top \mathbf{H}^{-1} \frac{\partial \mathbf{x}_k}{\partial x_k}}, \frac{\partial \mathbf{H}^{-1}}{\partial m_i} \right\rangle \tag{58}$$

and from Eq. 17 we also have

$$\frac{\partial w_k}{\partial \mathbf{H}^{-1}} = -\frac{\mathbf{x}_k \mathbf{x}_k^\top}{\left(\mathbf{x}_k^\top \mathbf{H}^{-1} \mathbf{x}_k\right)^2}, \tag{59}$$

which yields the full derivative

$$\frac{\partial w_k}{\partial m_i} = \Bigg\langle \underbrace{-\frac{\mathbf{x}_k \mathbf{x}_k^\top}{\left(\mathbf{x}_k^\top \mathbf{H}^{-1} \mathbf{x}_k\right)^2} + 2\frac{\mathbf{x}_k^\top \mathbf{H}^{-1} \frac{\partial \mathbf{x}_k}{\partial x_k} \mathbf{x}_k \boldsymbol{\eta}^\top}{\left(\mathbf{x}_k^\top \mathbf{H}^{-1} \mathbf{x}_k\right)^2 \boldsymbol{\eta}^\top \mathbf{H}^{-1} \frac{\partial \mathbf{x}_k}{\partial x_k}}}_{\mathbf{A}}, \frac{\partial \mathbf{H}^{-1}}{\partial m_i} \Bigg\rangle. \tag{60}$$

The second argument, $\frac{\partial \mathbf{H}^{-1}}{\partial m_i}$, remains finite and the result is a linear combination of the entries of the matrix $\mathbf{A} \in \mathbb{R}^{(n+1)\times(n+1)}$. So, we now study the entries of the matrix $\mathbf{A}$ and their limit behavior.

Expanding the terms to the same denominator gives

$$\mathbf{A} = \frac{-\mathbf{x}_k \mathbf{x}_k^\top \boldsymbol{\eta}^\top \mathbf{H}^{-1} \frac{\partial \mathbf{x}_k}{\partial x_k} + 2\mathbf{x}_k^\top \mathbf{H}^{-1} \frac{\partial \mathbf{x}_k}{\partial x_k} \mathbf{x}_k \boldsymbol{\eta}^\top}{\left(\mathbf{x}_k^\top \mathbf{H}^{-1} \mathbf{x}_k\right)^2 \boldsymbol{\eta}^\top \mathbf{H}^{-1} \frac{\partial \mathbf{x}_k}{\partial x_k}}. \tag{61}$$

Each entry $A_{i,j}$ of the matrix $\mathbf{A} = \{A_{i,j}\}_0^n$ is a rational function in x_k, with the same denominator as $\frac{\partial w_k}{\partial \eta}$ (minimal degree $5n-2$). The entry $A_{n,n}$ maximizes the degree of the numerator, so, in the limit, it gives an upper bound on all other entries. By assuming full degree in the numerator and minimal degree in the denominator, we get an upper bound on $A_{n,n}$:

$$\left|A_{n,n}\right| \leq c \left|\frac{x_k^{3n-1}}{x_k^{5n-2}}\right|, \quad c > 0. \tag{62}$$

The upper bound goes to 0 as $x_k \to \pm\infty$ and so does $\frac{\partial w_k}{\partial m_i}$. □