Quadric-Based Silhouette Sampling for Differentiable Rendering

MARIIA SOROKA, Cornell University, USA and Intel, Germany
CHRISTOPH PETERS, Delft University of Technology, Netherlands and Intel, Germany
STEVE MARSCHNER, Cornell University, USA

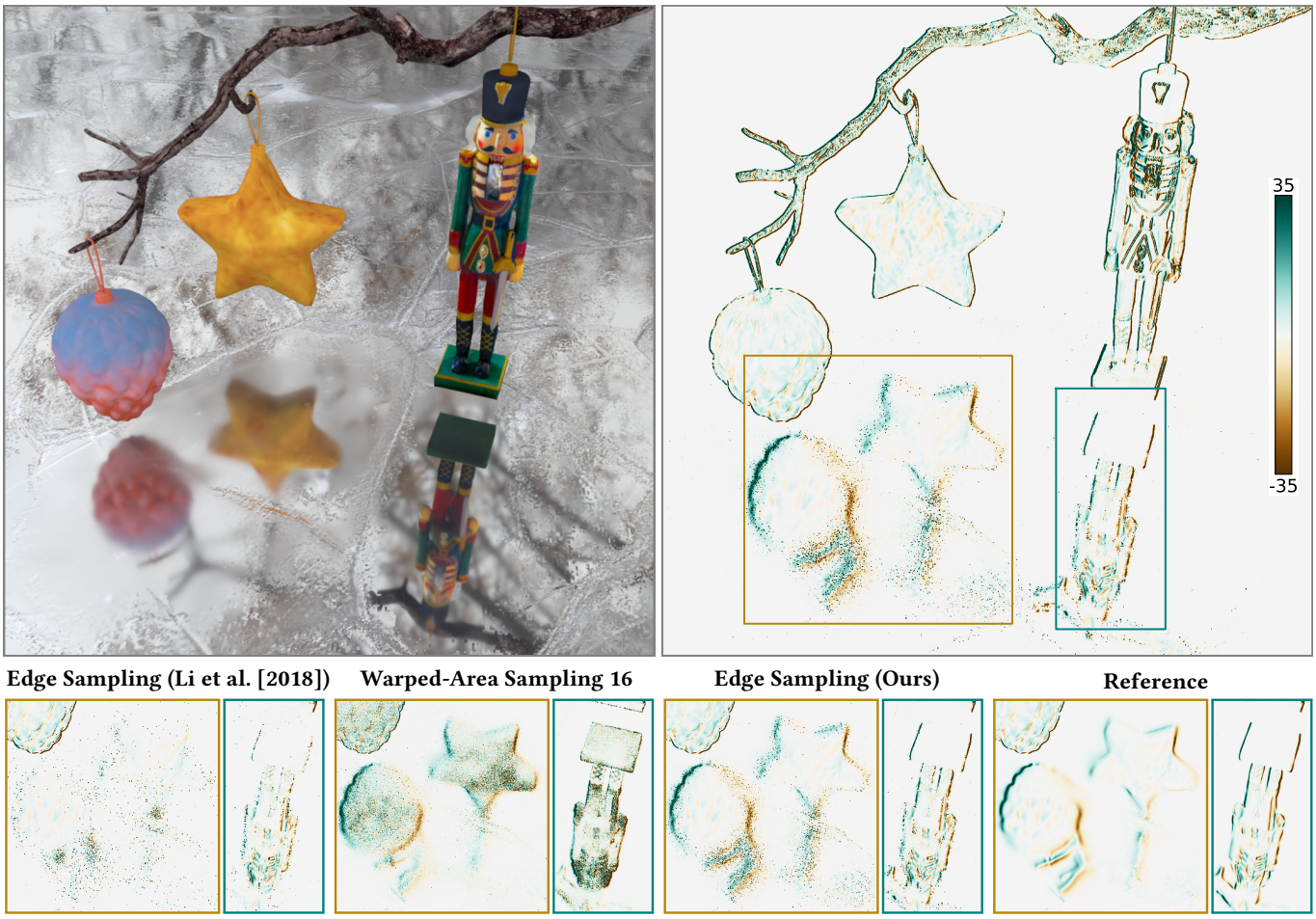


Fig. 1. We render the derivative w.r.t. horizontal translation of several objects observed through glossy reflections. In an equal-time comparison, we achieve much lower variance than prior work on edge sampling [Li et al. 2018]. Warped-area sampling (with 16 auxiliary rays) [Bangaru et al. 2020] exhibits bias on reflections of branches and noise across object areas.

Authors' addresses: Mariia Soroka, Cornell University, Ithaca, USA and Intel, Karlsruhe, Germany, ms3663@cornell.edu; Christoph Peters, Delft University of Technology, Delft, Netherlands and Intel, Karlsruhe, Germany, c.j.p.peters@tudelft.nl; Steve Marschner, Cornell University, Ithaca, USA, steve.marschner@cornell.edu.

ACM 0730-0301/2025/8-ART
https://doi.org/10.1145/3731146

Physically based differentiable rendering has established itself as key to inverse rendering, in which scenes are recovered from images through gradient-based optimization. Taking the derivative of the rendering equation is made difficult by the presence of discontinuities in the integrand at object silhouettes. To obtain correct derivatives w.r.t. changing geometry, accounting e.g. for changing penumbras or silhouettes in glossy reflections, differentiable renderers must compute an integral over these silhouettes. Prior work proposed importance sampling of silhouette edges for a given shading point. The main challenge is to efficiently reject parts of the mesh without silhouettes during sampling, which has been done using top-down traversal of a tree. Inaccuracies of this existing rejection procedure result in many samples with zero contribution. Thus, variance remains high and subsequent work has focused on alternatives such as area sampling or path

space differentiable rendering. We propose an improved rejection test. It reduces variance substantially, which makes edge sampling in a unidirectional path tracer competitive again. Our rejection test relies on two approximations to the triangle planes of a mesh patch: a bounding box in dual space and dual quadrics. Additionally, we improve the heuristics used for stochastic traversal of the tree. We evaluate our method in a unidirectional path tracer and achieve drastic improvements over the original edge sampling and outperform methods based on area sampling.

CCS Concepts: • **Computing methodologies** → **Rendering**.

Additional Key Words and Phrases: Differentiable rendering, inverse rendering, edge sampling, light transport, importance sampling, data structure

ACM Reference Format:
Mariia Soroka, Christoph Peters, and Steve Marschner. 2025. Quadric-Based Silhouette Sampling for Differentiable Rendering. *ACM Trans. Graph.* 44, 4 (August 2025), 20 pages. https://doi.org/10.1145/3731146

1 INTRODUCTION

Inverse rendering infers scene information, such as object geometry or material properties, from images. Early works in this field relied on simplified physical models and used direct fitting methods to recover these parameters from measurements [Gardner et al. 2003; Marschner 1998; Woodham 1992]. Analysis by synthesis is an attractive alternative to direct fitting methods. It assumes existence of a differentiable image generation algorithm (forward model) and uses a variant of gradient descent to iteratively update scene parameters until the output of the forward model is sufficiently close to the measured data. Physically based rendering [Pharr et al. 2023] enables generation of images indistinguishable from real-world photos and can serve as an excellent forward model. Recently proposed physically based differentiable rendering methods [Nimier-David et al. 2020; Vicini et al. 2021] compute gradients of images using a Monte Carlo simulation similar to the one used in forward rendering.

Evaluating gradients with respect to object geometry is crucial for shape reconstruction. Changes in these parameters cause discontinuous changes in visibility that must be handled explicitly by differentiable renderers. Due to the complexity of this problem and its importance for geometry reconstruction, it has received considerable attention in the last few years. Li et al. [2018] realized that the gradients due to changing discontinuities can be computed as an integral over silhouette and sharp edges of triangle meshes. They introduced edge sampling as a means to evaluate this integral with an unbiased Monte Carlo estimate.

Since this edge sampling suffers from high variance, subsequent work has explored alternatives. The warped area sampling (WAS) family of methods [Bangaru et al. 2020; Loubet et al. 2019; Xu et al. 2023] casts the boundary integral into an area integral using the divergence theorem. These methods trace auxiliary rays to construct their gradient estimate. Bangaru et al. [2020] propose a consistent variant with a fixed number of auxiliary rays and an unbiased variant with an unbounded number of rays. Path space differentiable rendering (PSDR) [Zhang et al. 2020, 2023] computes the boundary integral by starting construction of light paths on edges. From there, it extends the paths towards emitters and the sensor in a bidirectional fashion. With suitable guiding data structures, PSDR outperforms the existing edge sampling and WAS [Zhang et al. 2023]. However, it is fundamentally incompatible with unidirectional path tracing, which is the most popular approach in production rendering.

In our work, we revisit the edge sampling approach in a unidirectional path tracer [Li et al. 2018] and propose a new geometric data structure that substantially reduces its variance. The integrand in the boundary integral is non-zero only on silhouettes of objects and on sharp edges. Finding silhouettes is challenging, as silhouettes of triangle meshes are sparse, highly irregular, and vary from one shading point to another. Like Li et al. [2018], we rely on a hierarchical data structure where each node describes a set of edges in a mesh patch. A stochastic top-down traversal serves to sample a silhouette edge for a given shading point. In each step, child nodes that are known to contain no silhouette edges are rejected. For the remaining children, we and Li et al. [2018] estimate their contribution to the boundary integral using linearly transformed cosines (LTC) [Heitz et al. 2016] and randomly select one child with a probability proportional to this estimate of importance.

Our main improvement over the existing edge sampling [Li 2019; Li et al. 2018] is a considerably more accurate method to reject nodes without silhouette edges. That is important because inaccurate rejection results in lots of failed samples with zero contribution that drive up the variance. Disregarding visibility, an edge appears as a silhouette for any shading point in the wedge between its two adjacent triangle planes (Fig. 3 (a)). Hence, each node must store a compact and conservative description of the set of planes in these wedges. As a first step, we describe planes by their homogeneous coordinates (Sec. 3.1) and construct a tight 3D bounding box in this dual space (Sec. 3.2). To describe smooth geometry more accurately, the nodes additionally store a pair of quadrics such that each relevant plane is tangent to a quadric in between these two quadrics (Sec. 3.3). A node gets accepted when the given shading point lies on a plane that is both inside the dual 3D bounding box and also tangent to a quadric in this family of quadrics (Sec. 3.4).

In addition to this improved rejection test, we also improve the LTC-based estimate of the importance of a node (Sec. 3.5). When the scene contains polygonal area lights, the corresponding penumbras make significant contributions to gradients. We account for these explicitly by incorporating knowledge of these light sources into our importance heuristics (Sec. 3.6). We found that a forest of 4-wide trees works well as a data structure. To partition the mesh into a hierarchy of patches, we simply use a BVH build (Sec. 4.1). Then our algorithms construct the corresponding nodes without relying on any user-defined parameters (Secs. 4.2, 4.3 and 4.4).

Unlike PSDR, we use unidirectional path tracing, and as opposed to WAS, we do not need auxiliary rays and are always unbiased. We compare the performance of our method to existing solutions (Sec. 5). We show that our approach consistently outperforms both the original edge sampling [Li et al. 2018] and WAS [Bangaru et al. 2020; Xu et al. 2023]. Compared to a PSDR method with sophisticated guiding data structures [Zhang et al. 2023], our method has higher errors at equal sample count. However, we anticipate that this gap could be closed by augmenting our method with guiding as well. Overall, we show that edge sampling methods with carefully designed sampling data structures make unidirectional differentiable renderers a promising alternative to existing methods, contrary to conclusions based on the original edge sampling.

An open-source implementation of our method is available at https://github.com/mariasoroka/QuadricBasedSilhouetteSampling.

2 BACKGROUND AND RELATED WORK

2.1 Inverse Rendering

Inverse problems have received significant attention from the graphics community. With recent developments in inverse rendering it became possible to reconstruct complex material appearance [Deng et al. 2022, 2024], or to infer scene geometry [Vicini et al. 2022] or media properties [Nimier-David et al. 2022] relying solely on images. Using the analysis-by-synthesis approach as a foundation, these methods assume a forward image generation model F that takes a vector $\boldsymbol{\theta}$ of scene parameters as input and outputs an image. The similarity of two images is evaluated using a loss function $L(\cdot,\cdot)$. Inverse rendering algorithms attempt to find scene parameters $\boldsymbol{\theta}^*$ that minimize $L(F(\boldsymbol{\theta}), I_t)$—the loss for the reference image I_t and the image rendered by the forward model given scene parameters $\boldsymbol{\theta}$. The minimization problem is solved with a gradient-based optimization method, which requires evaluation of $\frac{\partial L(F(\boldsymbol{\theta}),I_t)}{\partial \boldsymbol{\theta}}$.

The properties and limitations of the resulting reconstruction strongly depend on the underlying forward model. Neural radiance fields (NeRF) [Mildenhall et al. 2020] and Gaussian splatting (GS) [Kerbl et al. 2023] methods achieve impressive geometry recovery and allow re-rendering of a reconstructed scene from different viewpoints. However, these methods use a simplified physically inaccurate forward model which is limiting for scenes with highly specular materials and does not directly support re-rendering with different illumination. Differentiable rendering algorithms use physically based rendering as a forward model and are capable of handling intricate global illumination effects such as shadows or reflections. Existing differentiable renderers are general and can be directly used for problems that previously required dedicated algorithms [Nicolet et al. 2024; Nimier-David et al. 2019].

2.2 Unidirectional Differentiable Rendering

The rendering equation [Kajiya 1986] relates outgoing radiance L_o scattered by a shading point $\mathbf{p}$ into direction ω_o to emission L_e and an integral of incoming radiance L_i multiplied by the cosine-weighted BSDF f_s over the hemisphere $\mathcal{H}^2$ around the surface normal:

$$L_o(\omega_o,\mathbf{p}) = L_e(\omega_o,\mathbf{p}) + \int_{\mathcal{H}^2} L_i(\omega_i,\mathbf{p}) f_s(\omega_i,\omega_o,\mathbf{p}) \mathrm{d}\omega_i. \quad (1)$$

We are interested in evaluating the derivative of outgoing radiance w.r.t. a scene parameter θ. Our main focus is on parameters controlling scene geometry such as object or vertex positions. For simplicity, we assume that emitted radiance L_e is independent of θ.

For a fixed shading point $\mathbf{p}$, the incoming radiance L_i is discontinuous across silhouette edges w.r.t. $\mathbf{p}$, boundary edges adjacent to only one triangle, sharp edges with discontinuous normals, or implicit edges caused by self-intersections in a mesh [Zhang et al. 2019]. Boundary, sharp, and implicit edges present discontinuities for any shading point and can be computed before the gradient evaluation. In contrast, the silhouette edges differ for each $\mathbf{p}$, making it challenging to handle this type of discontinuity. Moreover, objects used in inverse rendering are usually represented with watertight meshes with smooth normals and no self-intersections. Thus, we restrict ourselves to this type of mesh and handle only silhouette discontinuities. We note that this is merely a technical limitation of our implementation which can be trivially extended to support other types of discontinuities using methods proposed by Li et al. [2018].

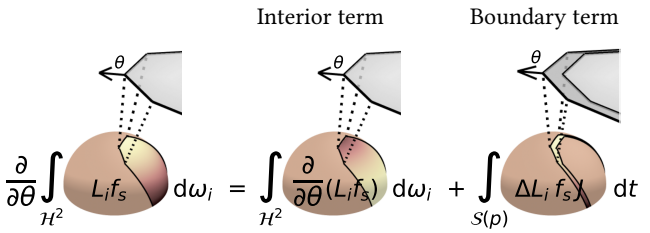


Fig. 2. Illustration for Equations 2 and 3. The scene features a moving object (shown in grey). The position of the shape is defined by parameter θ. Outgoing radiance L_o at a shading point is an integral of $L_i f_s$ over the upper hemisphere $\mathcal{H}^2$. A change in θ causes L_o to change in two ways. First, the observed color of the object is dependent on object position and changes as the shape moves. This is captured by the interior term. Second, as θ increases, more directions ω_i get occluded by the object. The boundary term accounts for this change by integrating over shape silhouettes.

Fig. 1 features a scene with rough reflections of several objects. The parameter θ controls the horizontal translation of the objects. A change in θ causes a change in the reflection that we would like to quantify. If we consider one point $\mathbf{p}$ on the reflecting plane and analyze how object motion changes the outgoing radiance (Fig. 2), we find that part of the difference is due to the smooth change in the observed color of the object (interior term, Eq. 2), and the other part is due to the change in visibility that occurs on the silhouette (boundary term, Eq. 3):

$$\frac{\partial}{\partial\theta} L_o(\omega_o,\mathbf{p}) = \int_{\mathcal{H}^2} \frac{\partial}{\partial\theta}\left(L_i(\omega_i,\mathbf{p}) f_s(\omega_i,\omega_o,\mathbf{p})\right) \mathrm{d}\omega_i \quad (2)$$

$$+ \sum_{e\in\mathcal{S}(\mathbf{p})} \int_0^1 \Delta L_i(\omega(t,e),\mathbf{p}) f_s(\omega(t,e),\omega_o,\mathbf{p}) J(t,e)\mathrm{d}t, \quad (3)$$

where $\mathcal{S}(\mathbf{p})$ denotes the set of silhouette edges of the mesh. If $\mathcal{E}$ is the set of all edges, and $\mathbf{n}_0(e)$, $\mathbf{n}_1(e)$ are consistently oriented geometric normals of the two triangles adjacent to an edge e,

$$\mathcal{S}(\mathbf{p}) = \{e \in \mathcal{E} \mid (\mathbf{n}_0^T(e)\mathbf{w}(0,e))\,(\mathbf{n}_1^T(e)\mathbf{w}(0,e)) \le 0\}.$$

For each edge e with endpoints $\mathbf{p_0}(e)$ and $\mathbf{p_1}(e)$, we define

$$\mathbf{w}(t,e) = (1-t)\,\mathbf{p_0}(e) + t\,\mathbf{p_1}(e) - \mathbf{p}, \quad \omega(t,e) = \frac{\mathbf{w}(t,e)}{\|\mathbf{w}(t,e)\|}$$

$$\mathbf{v}(t,e) = \frac{\partial}{\partial\theta}\mathbf{w}(t,e), \quad J(t,e) = \frac{\det\left(\mathbf{w}(0,e),\mathbf{w}(1,e),\mathbf{v}(t,e)\right)}{\|\mathbf{w}(t,e)\|^3}.$$

The difference of incoming radiance on two sides of the edge can be expressed as

$$\Delta L_i(\omega(t,e),\mathbf{p}) := \big(L_i(\omega(t,e),\mathbf{p}) - L_i(\omega(t,e),\mathbf{w}(t,e)+\mathbf{p})\big) V_{\mathbf{p}}(\mathbf{w}(t,e)),$$

where $V_{\mathbf{p}}(\mathbf{w})$ is the visibility function that equals 1 when the point $\mathbf{w}+\mathbf{p}$ is directly visible from $\mathbf{p}$ and zero otherwise. Evaluating this term requires traversing two light paths: one starting at point $\mathbf{p}$ and intersecting scene geometry at the edge point $\mathbf{w}(t,e) + \mathbf{p}$, and

another path originating right behind the edge point $\mathbf{w}(t,e)+\mathbf{p}$ and traveling further in direction $\omega(t,e)$.

The orientation of an edge e relative to the velocity $\mathbf{v}(t,e)$ impacts the contribution of an edge point $\mathbf{w}(t,e)$ to the gradient. If $\mathbf{v}(t,e)$ is parallel to the plane formed by the edge and the shading point, edge point $\mathbf{w}(t,e)$ claims a zero change in gradients. This effect is captured by the Jacobian term $J(t,e)$. Prior work uses different albeit equivalent formulations of the boundary integral [Li et al. 2018]. For our complete derivation, we refer the reader to the Appendix A.

2.3 Edge Sampling

Li et al. [2018] introduced an edge sampling technique to compute boundary integrals. Later, Li iterated on it in his thesis [Li 2019] and published an implementation in the differentiable renderer redner[1]. Since our work employs the same approach, we describe their method in detail.

The idea of the edge sampling method is to approach the problem of evaluating the boundary integral directly and to compute it using Monte Carlo integration. Li et al. [2018] suggest using an importance sampling technique to reduce variance of the Monte Carlo estimator. They first choose an edge e out of all the edges in a scene with discrete probability $\Pi(e)$ and then choose a point $\mathbf{w}(t,e)+\mathbf{p}$ on the edge by sampling parameter $t\in[0,1]$ with probability density $\Pi(t,e)$. Ideally, $\Pi(e)\Pi(t,e)$ should match the integrand in Eq. 3 up to a constant factor. This implies that $\Pi(e)$ should be non-zero only for silhouette edges and should be proportional to the contribution of an edge to the boundary integral, and that $\Pi(t,e)$ should be proportional to contributions of points along the edge.

Li et al. [2018] use a binary tree for stochastic edge sampling, similar to light bounding volume hierarchies (BVH) [Conty Estevez and Kulla 2018]. Each leaf holds one edge, and interior nodes correspond to all edges of a geometry patch and store aggregate information required for traversal. In the most recent version of the edge sampling method, Li [2019] uses a 6D BVH where the first three dimensions bound the spatial extent of the edges, and the last three dimensions encode information about triangle planes using Hough transforms [Olson and Zhang 2006]. The *v-sphere* test is used to reject nodes without silhouettes (see Appendix F).

When sampling an edge, the data structure is probabilistically traversed for each shading point. To descend one level down, the importance of each child node is computed as $I=L_w\max(f_s)/H$, where L_w is the total length of edges weighted by external dihedral angle, H is the distance to the center of the bounding box and $\max(f_s)$ is an upper bound on the value of the BSDF in the bounding box computed using the LTC approximation [Heitz et al. 2016]. Moreover, the silhouette detection test is used to discard irrelevant children. The next node is chosen stochastically with probabilities proportional to the computed importances. Once a leaf with a single edge is reached, a point on the edge is sampled proportional to the LTC value along this edge [Li et al. 2018; Peters 2021].

We use the latest version of the method implemented in redner as one of our baselines. We noticed that the *v-sphere* test implementation had a bug degrading the performance (Appendix F), which we fixed for our comparisons. We also want to point out that there is a mismatch in the implementation of the importance function and the formulas stated in the paper. To compute $\max(f_s)$, the authors find the direction that maximizes the cosine in the transformed domain, but in doing so, they ignore the Jacobian term that is crucial for LTCs to provide good fits to BRDFs. Since there is no way to correctly estimate the maximum of an LTC over a polygon with the same computational cost as the existing incorrect implementation, we keep this function unchanged for our comparisons.

It is important to note that the rejection test is designed to be conservative to ensure unbiasedness of the importance sampling but can safely give false positive results, i.e. the test is allowed to wrongly claim that there is a silhouette in a node. Tests producing more false positive results are less efficient, as they produce more zero-contribution samples. In our work, we suggest a completely new rejection test that is significantly more accurate, and come up with an improved importance function that better matches the edge contribution to the boundary integral.

2.4 Warped Area Sampling

Warped-area sampling (WAS) methods avoid explicit sampling of silhouettes by turning boundary integrals into area integrals. Loubet et al. [2019] proposed an approximate mapping based on local reparametrizations that keep the discontinuities fixed in the integration domain. Later, Bangaru et al. [2020] used the divergence theorem to derive an exact mapping and came up with an unbiased and a consistent gradient estimator. Both estimators require computing a warp field $\vec{\mathcal{V}}$ for every sampled ray direction. The field $\vec{\mathcal{V}}$ does not have an analytical closed-form expression and can only be evaluated numerically. To this end, multiple auxiliary rays are sampled from a von Mises–Fisher (vMF) distribution. The concentration of the vMF distribution is a user-controlled parameter that should be chosen based on scene geometry and can affect performance if not chosen well. If the number of auxiliary rays N is fixed, the estimator is biased but consistent, i.e. the bias vanishes as $N\to\infty$. The authors also propose a debiasing technique that chooses N at random for each ray. Xu et al. [2023] improved WAS by introducing a new distance function and extended it to support bidirectional path tracing. Vicini et al. [2022] and Bangaru et al. [2022] extended the WAS to support shapes represented with signed distance functions.

2.5 Path Space Differentiable Rendering

A path space formulation permits a different way of generating boundary paths. Instead of finding silhouette edges for a given shading point, PSDR starts path construction on an edge and then completes it to reach a camera and an emitter. Choosing the edge segment requires sampling a point on an edge and a segment direction. To reduce the variance of the final estimator, all existing PSDR methods use a precomputed guiding distribution that is stored either in a dense grid [Zhang et al. 2020] or in an adaptive data structure [Yan et al. 2022; Zhang et al. 2023]. Independently of the representation, the guiding distribution captures how much each edge segment contributes to the boundary gradient. Sampling proportionally to the stored function helps to account for the variability of the scene illumination (ΔL_i) and of the material properties of the objects (f_s).

[1] https://github.com/BachiLi/redner

To construct the guiding distribution, all PSDR methods require sampling many trial paths as a precomputation step.

2.6 Related Data Structures

The problem of edge sampling is similar to light sampling with many lights. Conty Estevez and Kulla [2018] stochastically traverse a BVH that spatially bounds light sources and approximates emission profiles using spherical caps. Moreau et al. [2019] demonstrate how to perform this traversal on the GPU. In non-photorealistic rendering, there is also a need to identify silhouette edges. Tsiapkolis and Bénard [2024] use nodes with rejection based on bounding spheres and spherical caps around normals. The nodes are built bottom-up, but the data structure is a flat list of nodes for disjoint mesh patches.

3 NODE TRAVERSAL PROBABILITY

In this section, we discuss what information is stored in inner nodes of our hierarchy and how it is used during traversal. First, we cover underlying concepts from projective geometry (Sec. 3.1) and describe how to compactly approximate a set of edges in a mesh patch (Secs. 3.2, 3.3). Then, we proceed to the discussion of our rejection test (Sec. 3.4) and importance function (Secs. 3.5, 3.6). Sec. 4 explains the node construction algorithm.

3.1 Silhouette Properties

We first study a single edge e adjacent to two triangle planes $\mathbf{q}_0(e)$ and $\mathbf{q}_1(e)$ (Fig. 3 (a.2)). This edge is a silhouette for a shading point $\mathbf{p}$ if one of the planes is front-facing and the other is back-facing w.r.t. $\mathbf{p}$ or, in other words, when the shading point lies in the wedge $\mathcal{W}(e)$ formed by $\mathbf{q}_0(e)$ and $\mathbf{q}_1(e)$.

The position of a point relative to a plane or a wedge can be compactly expressed using projective geometry. The homogeneous coordinates of the shading point $\mathbf{p}$ are $\mathbf{x} = [\mathbf{p}, 1]^T \in \mathbb{R}^4$. The homogeneous coordinates of a plane are $\mathbf{q} = [\mathbf{n}, n_o]^T \in \mathbb{R}^4$, where $\mathbf{n}$ is the normal direction of the plane, and n_o is the offset along the normal. A positive, negative or zero value of the dot product $\mathbf{q}^T\mathbf{x}$ indicates that the plane is front-facing, back-facing or incident to the point $\mathbf{p}$, respectively (Fig. 3 (a.1)).

Note that both points and planes are described by 4D vectors. Thus, a plane has a dual interpretation as a point and vice versa. This duality is a central concept in projective geometry [Richter-Gebert 2011, Sec. 3.5]. To distinguish between the two interpretations, for the original interpretation, we say that $\mathbf{x}$ is a *primal* point and $\mathbf{q}$ is a *primal* plane, and for the other interpretation, we say that $\mathbf{x}$ is a *dual* plane and $\mathbf{q}$ is a *dual* point. In Fig. 3, 4, and 5, we visualize planes as dual points. Since 4D vectors cannot be clearly pictured, we use lower-dimensional examples to illustrate our ideas. In Fig. 3 (a.1) and (a.4), for example, a line (or a primal hyperplane in 2D) $\mathbf{q}$ is plotted as a dual 3D point in the dual space. When interpreting illustrations with dual objects, it is important to keep in mind that distances between *dual* points (*primal* planes) are not defined in the same way as distances between *primal* points (Sec. 4.3 and Appendix E).

The wedge $\mathcal{W}(e)$ consists of all planes between the triangle planes $\mathbf{q}_0(e)$ and $\mathbf{q}_1(e)$. Given that the normals of the planes are chosen consistently, the set of planes in the wedge can be characterized using homogeneous coordinates as $\mathcal{W}(e) = \{\alpha\mathbf{q}_0(e) + (1-\alpha)\mathbf{q}_1(e) | \alpha \in [0,1]\}$, i.e. the wedge is a line segment connecting the homogeneous coordinates of the two bounding planes (Fig. 3 (a.5)). Now, we can conclude that $\mathbf{p}$ is in the wedge $\mathcal{W}(e)$ and that e is a silhouette edge for $\mathbf{p}$ if and only if there is a $\mathbf{q} \in \mathcal{W}(e)$ s.t. $\mathbf{q}^T\mathbf{x} = 0$. For a mesh patch with multiple edges, there is a silhouette edge for $\mathbf{p}$ if it belongs to one of the wedges (Fig. 3 (a.3)). Explicitly constructing the union of points in the wedges and checking whether $\mathbf{p}$ belongs to this set is a complex problem requiring computing halfspace intersections for a large number of planes. Thus, we take an alternative route and describe the wedges as a set of planes $\mathcal{W} = \{\mathcal{W}(e) \mid e \in \mathcal{E}\} \subset \mathbb{R}^4$. Now, to check that $\mathbf{p}$ belongs to one of the wedges, it is sufficient to check whether $\mathbf{p}$ is incident to one of the planes in $\mathcal{W}$, i.e. to verify that there is a $\mathbf{q} \in \mathcal{W}$ s.t. $\mathbf{x}^T\mathbf{q} = 0$. The set $\mathcal{W}$ is easy to compute since it is a set of line segments in dual space (Fig. 3 (a.6)). Explicitly computing the dot products for all line segments in $\mathcal{W}$ is infeasible for large meshes. Thus, we approximate the set $\mathcal{W}$ with a bounding set $\mathcal{W}^* \supset \mathcal{W}$ that allows us to easily validate that $\mathbf{x}^T\mathbf{q} \neq 0,\ \forall \mathbf{q} \in \mathcal{W}^*$.

3.2 Bounding Box Approximation

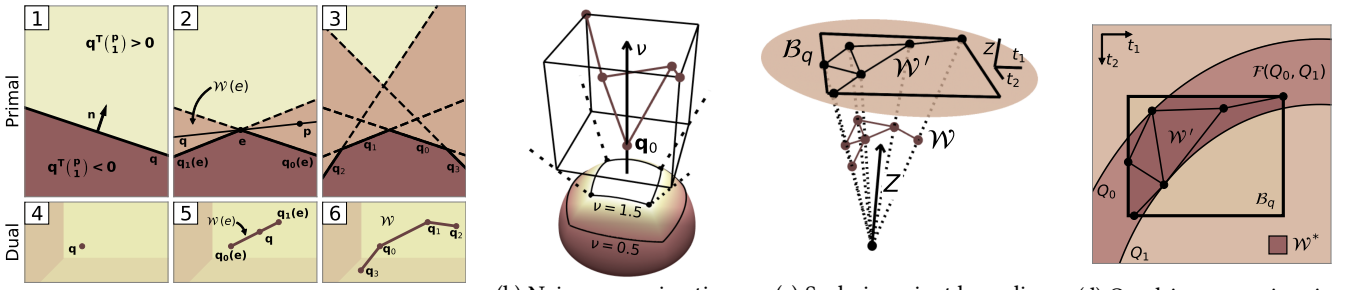
The simplest way to approximate the set $\mathcal{W}$ is to bound it with a 4D axis aligned bounding box. This approach, however, does not take into account that homogeneous coordinates of planes are scale invariant, i.e. that scaling a dual point $\mathbf{q}$ by a nonnegative constant does not change the plane it describes. Thus, this approach may result in a poor approximation of $\mathcal{W}$ if the scaling was not chosen well. In Fig. 3 (b), we provide an example of such a failure case in a lower-dimensional setting. Changing the scale ν of a plane $\mathbf{q}_0$ changes the bounding box and as a result the set of planes covered by it. We visualize this set with a contour plot on the hemisphere for each value of ν. It is clear that for small values of ν, the covered set is significantly larger than necessary.

To avoid the problem with scaling, we choose a unit vector $\mathbf{Z}$ s.t. $\mathbf{Z}^T\mathbf{q} > 0\ \forall \mathbf{q} \in \mathcal{W}$, and compute a new set of planes $\mathcal{W}'$ by scaling every plane $\mathbf{q} \in \mathcal{W}$ by $1/(\mathbf{Z}^T\mathbf{q})$. Since for any $\mathbf{q}' \in \mathcal{W}'$ it holds that $\mathbf{Z}^T\mathbf{q}' = 1$, $\mathcal{W}'$ lies on a dual plane. Fig 3 (c) provides an illustration of the sets $\mathcal{W}$ and $\mathcal{W}'$. We compute an orthogonal basis $A = [t_1, t_2, t_3, \mathbf{Z}] \in \mathbb{R}^{4\times4}$ and a 3D axis aligned bounding box $\mathcal{B}_q$ covering $\{[t_1^T\mathbf{q}', t_2^T\mathbf{q}', t_3^T\mathbf{q}']^T \in \mathbb{R}^3 \mid \mathbf{q}' \in \mathcal{W}'\}$. The set $\mathcal{A}(\mathcal{B}_q) = \{A[\mathbf{r}, 1]^T \mid \mathbf{r} \in \mathcal{B}_q\}$ covers all the wedges in the mesh patch and serves as an approximation of this set. The bounding box $\mathcal{B}_q$ is stored alongside A in every inner node.

Sec. 4.4 describes how we choose $\mathbf{Z}$. In short, it is the point or direction farthest from all planes in $\mathcal{W}$. This choice avoids unreasonably small values of $\mathbf{Z}^T\mathbf{q}$, which would result in large bounding boxes. When the described vector $\mathbf{Z}$ does not exist, it is guaranteed (Appendix B) that for any shading point, the mesh patch has a silhouette edge. This implies that the patch can never be skipped. Our method is able to identify such nodes during construction of the data structure.

3.3 Quadrics Approximation

The bounding box approximation is still crude. As illustrated in Fig. 3 (d), it can significantly enlarge the original set of planes. Many meshes used in practice have smooth geometry, and we utilize



(a) Planes, edges and wedges

(b) Naive approximation of a set of planes

(c) Scale-invariant bounding box approximation of $\mathcal{W}$

(d) Quadrics approximation of $\mathcal{W}$

Fig. 3. (a) Using homogeneous coordinates, a plane $\mathbf{q}$ can be visualized as a dual point (1 and 4); a wedge $\mathcal{W}(e)$ corresponding to an edge e formed by two planes $\mathbf{q}_0$ and $\mathbf{q}_1$ can be visualized as a line segment (2 and 5); finally, a set of wedges in a mesh patch can be visualized as a set of line segments (3 and 6). (b) A naive way to approximate the set of wedges is to use a bounding box. Due to scale invariance, multiplying only $\mathbf{q}_0$ by ν does not change the set of planes. However, it changes the bounding box. The color gradient on the unit hemisphere shows the set of planes covered by the bounding box for each ν. For small ν, the bounding box covers larger sets of planes and, thus, it is a worse approximation. (c) For a set of wedges $\mathcal{W}$ (brown line segments), set $\mathcal{W}'$ (black line segments) is constructed by scaling every $\mathbf{q} \in \mathcal{W}$ by $1/(Z^T\mathbf{q})$. Vectors t_1, t_2 and Z form an orthogonal basis in $\mathbb{R}^3$. $\mathcal{B}_q$ is a bounding box aligned with t_1 and t_2 covering planes in $\mathcal{W}'$. (d) View of the set $\mathcal{W}'$ and bounding box $\mathcal{B}_q$ from "above." The curves Q_0 and Q_1 illustrate two bounding quadrics defining the set $\mathcal{F}(Q_0, Q_1)$. The set $\mathcal{W}^*$ is the intersection of $\mathcal{B}_q$ and $\mathcal{F}(Q_0, Q_1)$.

this property by approximating the set of planes with a smooth surface. In our case, we use quadrics. A quadric is defined by a 4×4 matrix Q and consists of all 4D vectors satisfying $\rho^T Q \rho = 0$. When ρ represents a primal point, we call Q a primal quadric. Some examples of primal quadrics are ellipsoids, hyperboloids and paraboloids. When ρ represents a dual point (a plane), we refer to Q as a dual quadric. Note that Q^{-1} is also 4×4 matrix that can be interpreted as a primal or a dual quadric. The primal quadric Q^{-1} is fundamentally related to the dual quadric Q: if a plane $\mathbf{q}$ satisfies $\mathbf{q}^T Q \mathbf{q} = 0$, then $\mathbf{q}$ is tangent to the quadric Q^{-1} in primal space. Essentially, the set of planes that are tangent to the primal quadric Q^{-1} is described as planes incident to the dual quadric Q.

Our goal is to bound the set of wedges $\mathcal{W}$ and we use dual quadrics to accomplish that. For a general set of planes, one dual quadric cannot be enough to cover the set entirely. Thus, we use a family of dual quadrics. Given two bounding quadrics Q_0 and Q_1, we define

$$\mathcal{F}(Q_0, Q_1) = \{\mathbf{q} \in \mathbb{R}^4 \mid \exists \mu \in [0, 1] \text{ s.t. } \mathbf{q}^T(\mu Q_0 + (1 - \mu) Q_1)\mathbf{q} = 0\}.$$

In essence, this set contains all tangent planes of quadrics $(Q_0\mu + Q_1(1 - \mu))^{-1}$ as $\mu \in [0, 1]$ varies. The bounding quadrics are chosen in a way that $\mathcal{F}(Q_0, Q_1)$ covers the entire set $\mathcal{W}$. Our final approximation of $\mathcal{W}$ is

$$\mathcal{W}^* = \mathcal{F}(Q_0, Q_1) \cap \mathcal{A}(\mathcal{B}_q).$$

To compute node importance during traversal (Sec. 3.5), we additionally store a primal bounding box $\mathcal{B}_p$ covering all the vertices in the mesh patch, and the sum of edge lengths weighted by external dihedral angle L_w.

3.4 Rejection Method

Our rejection test takes a node and a point $\mathbf{x}$ in homogeneous coordinates as input. Our goal is to check whether there is a plane $\mathbf{q} \in \mathcal{W}^*$ for which $\mathbf{q}^T\mathbf{x} = 0$. If there is one, there may be a silhouette in the subtree and the node cannot be rejected. We start by finding

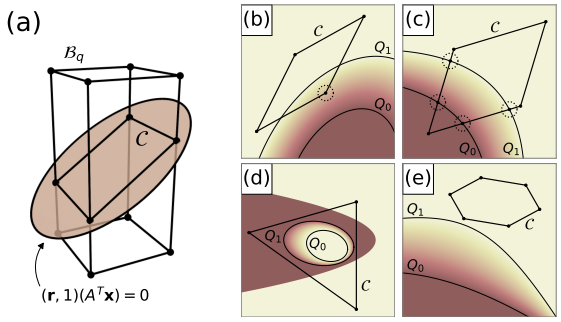


Fig. 4. Given a query point $\mathbf{x}$, the first step of the rejection test is to find the polygon C (possibly degenerate) lying in the intersection of the bounding box $\mathcal{B}_q$ and a plane with homogeneous coordinates $A^T\mathbf{x}$ (a). The second step is intersecting the polygon C with the set of planes $\mathcal{F}(Q_0, Q_1)$ which requires intersecting $\mathcal{F}(Q_0, Q_1)$ with polygon vertices (b), polygon edges (c), and polygon interior (d), or confirming that no intersection exists (e).

all planes $\mathbf{q}$ in the dual bounding box $\mathcal{A}(\mathcal{B}_q)$ for which $\mathbf{q}^T\mathbf{x} = 0$. Based on the definition of $\mathcal{A}(\mathcal{B}_q)$, that is equivalent to finding all vectors $\mathbf{r} \in \mathcal{B}_q$ such that $[\mathbf{r}, 1](A^T\mathbf{x}) = 0$. That is the same as intersecting the dual 3D bounding box $\mathcal{B}_q$ with a dual plane defined by the 4D vector $A^T\mathbf{x}$. The set of 3D points in this intersection is either empty or it is a (possibly degenerate) 2D polygon (Figure 4 (a)). If the intersection is empty, we can guarantee that $\mathbf{q}^T\mathbf{x} \neq 0$ for all $\mathbf{q} \in \mathcal{A}(\mathcal{B}_q)$ and there is no silhouette in the node. Otherwise, we explicitly compute the set of vertices $\mathcal{V}$ of the 2D polygon. The corresponding set of planes is a dual 2D polygon with vertices defined as $A[\mathbf{v}, 1]^T \in \mathbb{R}^4$ where $\mathbf{v} \in \mathcal{V}$. We denote this dual polygon as C.

The next step is to check whether C intersects the set of planes $\mathcal{F}(Q_0, Q_1)$. This is equivalent to finding a plane $\mathbf{q} \in C$ such that it

also belongs to $\mathcal{F}(Q_0, Q_1)$ or, in other words, for which

$$\exists \mu(\mathbf{q}) \in [0,1] \text{ s.t. } \mathbf{q}^T \left(\mu(\mathbf{q})Q_0 + (1-\mu(\mathbf{q}))Q_1\right)\mathbf{q} = 0 \Leftrightarrow$$
$$\mu(\mathbf{q}) = \frac{\mathbf{q}^T Q_1 \mathbf{q}}{\mathbf{q}^T (Q_1 - Q_0)\mathbf{q}} \in [0,1]$$

or proving that no such plane exists. Our choice of bounding quadrics guarantees that the function $\mu(\mathbf{q})$ is continuous on the polygon C (see Sec. 4.4). If there is a dual point $\mathbf{q} \in C$ for which $\mu(\mathbf{q}) \in [0,1]$, one of the following three mutually exclusive conditions holds:

- Cond. (1) There is a polygon vertex $\mathbf{q}_v$ s.t. $\mu(\mathbf{q}_v) \in [0,1]$ (Fig. 4 (b)).
- Cond. (2) Cond. (1) does not hold, but $\mu(\mathbf{q}) \in [0,1]$ on an edge of the polygon (Fig. 4 (c)).
- Cond. (3) Neither Cond. (1) nor Cond. (2) holds, but $\mu(\mathbf{q}) \in [0,1]$ in the interior of C (Fig. 4 (d)).

If none of the three conditions holds true (Fig. 4 (e)), polygon C does not intersect $\mathcal{F}(Q_0, Q_1)$ and thus, the node does not contain a silhouette. Checking Cond. (1) is trivial—it is sufficient to evaluate μ at each vertex of the polygon.

For Cond. (2), we notice that if $\mu(\mathbf{q}_v) \notin [0,1]$ for all vertices but there is a dual point $\mathbf{q}_e$ on an edge s.t. $\mu(\mathbf{q}_e) \in [0,1]$, it implies that there must be a dual point on an edge s.t. $\mu(\mathbf{q}) \in \{0,1\} \Leftrightarrow \mathbf{q}^T Q_0 \mathbf{q} = 0$ or $\mathbf{q}^T Q_1 \mathbf{q} = 0$. So, checking Cond. (2) boils down to verifying that one of the bounding quadrics intersects an edge of the polygon. Since any dual point $\mathbf{q}$ on an edge with endpoints $\mathbf{q}_0$ and $\mathbf{q}_1$ can be expressed as $\mathbf{q} = \lambda\mathbf{q}_0 + (1-\lambda)\mathbf{q}_1$, finding the intersection of the edge with a quadric Q is equivalent to solving a quadratic equation for $\lambda \in [0,1]$:

$$\mathbf{q}^T Q \mathbf{q} = (\lambda\mathbf{q}_0 + (1-\lambda)\mathbf{q}_1)^T Q (\lambda\mathbf{q}_0 + (1-\lambda)\mathbf{q}_1) = 0.$$

For Cond. (3), we use a similar reasoning as for Cond. (2). If $\mu(\mathbf{q}) \notin [0,1]$ for all dual points on the boundary of the polygon, but there is $\mathbf{q}_i$ inside of it s.t. $\mu(\mathbf{q}_i) \in [0,1]$, there must be a dual point inside of the polygon that belongs to one of the bounding quadrics. Thus, to check Cond. (3), it is necessary to intersect bounding quadrics with the polygon. This test can be efficiently performed using the algorithm provided in Appendix C.

3.5 Node Importance

To construct a good estimate of the boundary integral (Eq. 3), it is necessary to use importance sampling and to choose nodes based on their contribution to the integral. Importantly, this contribution varies from one shading point to another. Thus, the sampling probability should be dependent on the shading point as well. The perfect node importance function is proportional to the overall contribution of edges in the node, which is impossible to compute without explicitly integrating over all of them. Instead, we use a few approximations of the boundary integral that allow us to evaluate importance based on the information stored in inner nodes.

First, we assume that ΔL_i is constant along all edges in the scene. We deviate from this assumption only when explicitly accounting for polygonal lights (Sec. 3.6). Second, because the velocities are not known in advance during the backward gradient computation, we assume them to be constant, i.e. $\|\mathbf{v}(t,e)\| = 1$. We would like our node importance to be conservative, and replace $\det(\mathbf{w}(0,e), \mathbf{w}(1,e), \mathbf{v}(t,e))$ with its maximal possible value $\|\mathbf{w}(0,e) \times \mathbf{w}(1,e)\|$. These two simplifications leave us with the following expression for the node importance:

$$I^*(\omega_o, \mathbf{p}) = \sum_{e \in \mathcal{S}(\mathbf{p})} \int_0^1 f_s(\omega(t,e), \omega_o, \mathbf{p}) \frac{\|\mathbf{w}(0,e) \times \mathbf{w}(1,e)\|}{\|\mathbf{w}(t,e)\|^3} \mathrm{d}t. \quad (4)$$

To simplify the expression even further, we approximate f_s with an average BSDF value in the solid angle $\Omega(\mathcal{B}_p)$ of the spatial bounding box of the mesh patch $\mathcal{B}_p$:

$$I^*(\omega_o, \mathbf{p}) \approx \bar{f}_s(\Omega(\mathcal{B}_p)) \sum_{e \in \mathcal{S}(\mathbf{p})} \int_0^1 \frac{\|\mathbf{w}(0,e) \times \mathbf{w}(1,e)\|}{\|\mathbf{w}(t,e)\|^3} \mathrm{d}t,$$

where the average of the BSDF in the solid angle of the bounding box is given by

$$\bar{f}_s(\Omega(\mathcal{B}_p)) = \frac{\int_{\Omega(\mathcal{B}_p)} f_s(\omega_i, \omega_o, \mathbf{p}) \mathrm{d}\omega_i}{\int_{\Omega(\mathcal{B}_p)} 1 \mathrm{d}\omega_i}.$$

To estimate $\bar{f}_s(\Omega(\mathcal{B}_p))$, we employ LTCs [Heitz et al. 2016]. For the second term, we use a far-field approximation, i.e. we assume that for any edge e, its length $L(e)$ is small compared to the distance to its vertices $H(e) = \|\mathbf{w}(0,e)\|$. Additionally, we ignore the orientation of the edge and assume that $(\mathbf{w}(1,e) - \mathbf{w}(0,e)) \perp \mathbf{w}(0,e)$. These simplifications lead to the following approximation:

$$\sum_{e \in \mathcal{S}(\mathbf{p})} \int_0^1 \frac{\|\mathbf{w}(0,e) \times \mathbf{w}(1,e)\|}{\|\mathbf{w}(t,e)\|^3} \mathrm{d}t \approx \sum_{e \in \mathcal{S}(\mathbf{p})} \frac{L(e)}{H(e)^2}.$$

We approximate $H(e)$ for all edges in the node as the distance H from the shading point to the center of the bounding box $\mathcal{B}_p$. For $\sum_{e \in \mathcal{S}(\mathbf{p})} L(e)$, we use the length of all edges weighted by external dihedral angle L_w, which we precompute during tree build. Our final expression for the node importance is:

$$I(\omega_o, \mathbf{p}) = \frac{L_w}{H^2} \bar{f}_s(\Omega(\mathcal{B}_p)).$$

The proposed importance function is different from the one used by Li et al. [2018] in two ways. First, we use the average of the BSDF in the solid angle of the bounding box instead of its maximum. Unlike the maximum value, the average can be correctly and efficiently evaluated using the LTC approximation (see Sec. 2.3). Secondly, our derivations indicate that the importance function should be inversely proportional to H^2 instead of H as suggested by Li et al. [2018].

3.6 Handling Polygonal Lights

The node importance discussed in the previous section works well for low-frequency environment lights, but can be improved to explicitly take into account polygonal lights in the scene.

Instead of assuming that ΔL_i is constant along every edge, we take advantage of the fact that the radiance of a Lambertian polygonal light is known in advance and incorporate this knowledge into our importance function. Instead of computing the BSDF integral over the entire bounding box $\mathcal{B}_p$, we clip the solid angle $\Omega(\mathcal{B}_p)$ against

the solid angle subtended by the polygonal light $\Omega(\mathcal{L})$ and use

$$I_{\text{light}}(\omega_o, \mathbf{p}) = \frac{L_w}{H^2} \frac{\int_{\Omega(\mathcal{B}_p)\cap\Omega(\mathcal{L})} f_s(\omega_i, \omega_o, \mathbf{p}) d\omega_i}{\int_{\Omega(\mathcal{B}_p)} 1 d\omega_i}.$$

We then weight I_{light} and I by the average polygonal light and environment light intensities, respectively, and use it as the final node importance function. This modification greatly improves sampling quality in presence of polygonal lights (Sec. 5.3). This approach can be extended to scenes with multiple polygonal lights by summing the importances computed for each light source. For scenes with a large number of light sources, more sophisticated techniques are required.

4 BUILDING THE HIERARCHY

In this section, we cover all aspects of hierarchy construction. In Sec. 4.1, we discuss how to assign mesh patches to hierarchy nodes. Sec. 4.2-4.4 explain how to compute the bounding quadrics Q_0 and Q_1 and how to choose the vector Z used for the bounding box approximation in Sec. 3.2.

4.1 Tree Structure

Our rejection test and importance function are more accurate for smaller mesh patches. Thus, using a wide tree is beneficial as it enables faster descent to smaller patches, allowing for high-quality rejections early in the traversal. However, very wide trees can become computationally expensive as each traversal step requires many rejection tests. We use 4-wide trees: they have the same traversal cost as binary trees (descending one level is twice as expensive, but the tree is half as deep) and are easy to build. We construct a separate tree for each user-defined object. To avoid poor traversal decisions at the top of the hierarchy, we discard the root node, leaving us with four subtrees per object. The resulting data structure is a forest with $4N$ trees where N is the number of objects. At every step of the traversal—including the initial choice among the $4N$ trees—we sample the next node proportionally to the rejection test and the importance function.

To build the tree for an object, we start by eliminating all edges with dihedral angles greater than π since, for a watertight mesh with opaque BSDF, they never lie on a silhouette [Sander et al. 2001]. Next, we compute bounding boxes for the edges. We build the hierarchy in a top-down fashion. To create four child nodes from a parent node, we perform three splits: first, we split the parent node into two intermediate mesh patches, and then we split each of the intermediate patches once more to produce the four final child nodes. On each splitting step, we choose 10 equally spaced candidate split locations (also called buckets [Pharr et al. 2023]) along the largest axis of the parent bounding box, and choose the one that minimizes the surface area heuristic (SAH). The SAH is robust and performs well in the context of our problem. However, it was designed for the completely different task of building optimal data structures for ray tracing, and future work can potentially improve the quality of the hierarchy and computed gradients by developing a cost function tailored specifically to the problem of silhouette sampling.

4.2 Finding Bounding Quadrics

Once a node has been assigned a mesh patch with edges $\mathcal{E}$, triangle planes $\mathcal{P}$, and union of wedges $\mathcal{W}$, we construct a family of quadrics $\mathcal{F}(Q_0, Q_1)$. Quadric Q_f is fitted to the set of planes $\mathcal{P}$ (Sec. 4.3) and an offset quadric Q_o (Sec. 4.4) is used to form Q_0 and Q_1 as $Q_i = Q_f + \lambda_i Q_o$ for $i = 0, 1$. The offset coefficients λ_0 and λ_1 are chosen in such a way that $\mathcal{F}(Q_0, Q_1)$ is the smallest set that contains $\mathcal{W}$. We illustrate this idea in Fig. 5, where $\mathcal{W}$ is shown as a set of line segments. The background color shows which quadric $Q_f + \lambda Q_o$ each point belongs to, and highlighted quadrics correspond to the parameters that ensure the tightest bounds.

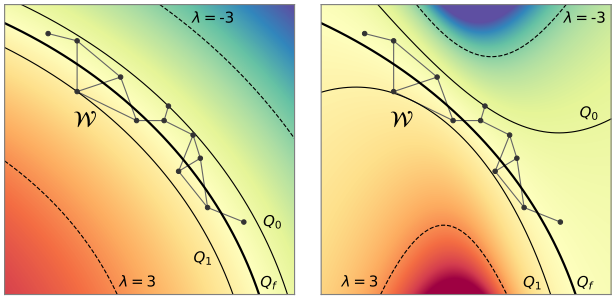


Fig. 5. 2D visualization of the union of wedges $\mathcal{W}$, fitted quadric Q_f and bounding quadrics Q_0 and Q_1. On the left and right images, the bounding quadrics are computed using two different offsets Q_o. The color gradient shows the value of the function $\lambda(\mathbf{q}) = -(\mathbf{q}^T Q_f \mathbf{q}) \,/\, (\mathbf{q}^T Q_o \mathbf{q})$. The left offset quadric gives a tighter fit.

In the following, we describe how to compute the optimal offset coefficients λ_0 and λ_1. First, we notice that

$$\mathcal{F}(Q_0, Q_1) = \{\mathbf{q} \in \mathbb{R}^4 \mid \exists \lambda \in [\lambda_0, \lambda_1] \text{ s.t. } \mathbf{q}^T (Q_f + \lambda Q_o)\mathbf{q} = 0\}.$$

To reflect the dependency of the set $\mathcal{F}$ on the offset parameters, we use the notation $\mathcal{F}(\lambda_0, \lambda_1) = \mathcal{F}(Q_0, Q_1)$.

A plane $\mathbf{q}$ belongs to the set $\mathcal{F}(\lambda_0, \lambda_1)$ if and only if

$$\exists \lambda(\mathbf{q}) \in [\lambda_0, \lambda_1] \text{ s.t. } \mathbf{q}^T (Q_f + \lambda(\mathbf{q}) Q_o)\mathbf{q} = 0 \Leftrightarrow$$

$$\lambda(\mathbf{q}) = -\frac{\mathbf{q}^T Q_f \mathbf{q}}{\mathbf{q}^T Q_o \mathbf{q}} \in [\lambda_0, \lambda_1].$$

For a single wedge w corresponding to an edge e, choosing $\lambda_0(e) = \min_{\mathbf{q}\in w} \lambda(\mathbf{q})$ and $\lambda_1(e) = \max_{\mathbf{q}\in w} \lambda(\mathbf{q})$ guarantees that every plane in the wedge belongs to $\mathcal{F}(\lambda_0(e), \lambda_1(e))$ and ensures optimal bounds. Note that for max and min to exist it is necessary that $\mathbf{q}^T Q_o \mathbf{q} \neq 0 \; \forall \mathbf{q} \in w$. The choice of Q_o in Sec. 4.4 accounts for that. We now seek an efficient way to compute these minima and maxima.

For a wedge w formed by two planes $\mathbf{q}_0(e)$ and $\mathbf{q}_1(e)$, any plane in w can be expressed as $\mathbf{q}(t) = (1-t)\mathbf{q}_0(e) + t\mathbf{q}_1(e)$ for some $t \in [0, 1]$. Since $\lambda(\mathbf{q}(t))$ is a differentiable function of $t \in [0, 1]$, its extrema are attained at $t = 0$, at $t = 1$ or at a critical point in between. Values of the function $\lambda(\mathbf{q}(t))$ at $t = 0$ and $t = 1$ are simply $\lambda(\mathbf{q}_0(e))$ and $\lambda(\mathbf{q}_1(e))$, respectively. Analytical expressions for the interior extremal points of $\lambda(\mathbf{q}(t))$ can be derived for a general offset quadric, but the resulting equations are complex. Thus, we restrict ourselves to the special case of rank-1 symmetric positive

semi-definite offset quadrics. We provide the full derivation of the expression for the interior extremal value of $\lambda(\mathbf{q}(t))$ in Appendix D and only present the final result here.

To compute extrema of $\lambda(\mathbf{q})$, we define

$$\lambda^*(e) = \frac{q_f^{00}(e)q_f^{11}(e) - (q_f^{01}(e))^2}{2q_f^{01}(e)q_o^{01}(e) - q_f^{00}(e)q_o^{11}(e) - q_f^{11}(e)q_o^{00}(e)}, \text{ where}$$

$$q_f^{ij}(e) = \mathbf{q}_i^T(e)Q_f\mathbf{q}_j(e) \text{ and } q_o^{ij}(e) = \mathbf{q}_i^T(e)Q_o\mathbf{q}_j(e),$$

and construct the set $\Lambda(e)$ as

$$\Lambda(e) = \tilde{\Lambda}(e) \cup \Lambda^*(e), \text{ where}$$

$$\tilde{\Lambda}(e) = \left\{\lambda(\mathbf{q}_0(e)), \lambda(\mathbf{q}_1(e))\right\},$$

$$\Lambda^*(e) = \begin{cases} \lambda^*(e) \text{ if } \left(\mathbf{q}_0(e)^T Q_{\lambda^*(e)}\mathbf{q}_1(e)\right)\left(\Delta\mathbf{q}(e)^T Q_{\lambda^*(e)}\Delta\mathbf{q}(e)\right) < 0, \\ \emptyset \text{ otherwise.} \end{cases}$$

Here we use $\Delta\mathbf{q}(e) = \mathbf{q}_0(e) - \mathbf{q}_1(e)$ and $Q_{\lambda^*(e)} = Q_f + \lambda^*(e)Q_o$.

The set $\tilde{\Lambda}(e)$ corresponds to the values of $\lambda(\mathbf{q})$ at the planes $\mathbf{q}_0(e)$ and $\mathbf{q}_1(e)$, and the set $\Lambda^*(e)$ contains the internal extremal value of the function if such an extremum exists. Now we can state that $\min_{\mathbf{q}\in w}\lambda(\mathbf{q}) = \min\Lambda(e)$ and $\max_{\mathbf{q}\in w}\lambda(\mathbf{q}) = \max\Lambda(e)$. Computing max and min of $\Lambda(e)$ is easy since $\Lambda(e)$ contains at most three values.

To extend this result from a single wedge w to the union of wedges $\mathcal{W}$, we define $\Lambda = \cup_{e\in\mathcal{E}}\Lambda(e)$ and compute $\lambda_0 = \min\Lambda$ and $\lambda_1 = \max\Lambda$. If the offset quadric Q_o is a rank-1 symmetric positive semi-definite matrix s.t. $\mathbf{q}^T Q_o\mathbf{q} > 0\ \forall\mathbf{q} \in \mathcal{W}$, such a choice of bounds is optimal and results in an unbiased rejection test.

4.3 Fitting a Quadric

Our goal is to find a quadric Q_f that best fits the set of triangle planes $\mathcal{P} \subset \mathbb{R}^4$. We base our fitting method on the work of Taubin [1991] that addresses a similar problem of fitting an implicit curve to a set of points. As a fitting error, the author uses an approximation of Euclidean distance from a point to a curve.

Since we are working with planes instead of points, we should choose our metric to reflect this difference. The distance between planes is well-defined only for parallel planes. In other cases, we say that the distance is infinite. Given two planes $\mathbf{q}_0$ and $\mathbf{q}_1$ with aligned unit length normals, the distance $d(\mathbf{q}_0, \mathbf{q}_1)$ between them is the absolute value of the difference in plane offsets. Using $A = \text{diag}(0, 0, 0, 1)$, this can be expressed as $d^2(\mathbf{q}_0, \mathbf{q}_1) = (\mathbf{q}_0 - \mathbf{q}_1)^T A(\mathbf{q}_0 - \mathbf{q}_1)$. In Appendix E, we show that using our definition of distance between planes and following the derivations proposed by Taubin [1991], we can approximate distance from a plane $\mathbf{q}$ to a quadric Q as

$$\tilde{d}^2(Q, \mathbf{q}) = \frac{9}{4}\frac{(\mathbf{q}^T Q\mathbf{q})^2}{\mathbf{q}^T QAQ\mathbf{q}}.$$

Similarly to the original method, we define the fitting error $e(Q)$ to be proportional to the sum of squared distances from points in $\mathcal{P}$ to a quadric Q and approximate it with $E(Q)$ as follows:

$$e(Q) := \frac{4}{9}\sum_{\mathbf{q}\in\mathcal{P}}\tilde{d}^2(Q, \mathbf{q}) = \sum_{\mathbf{q}\in\mathcal{P}}\frac{(\mathbf{q}^T Q\mathbf{q})^2}{\mathbf{q}^T QAQ\mathbf{q}} \approx \frac{\sum_{\mathbf{q}\in\mathcal{P}}(\mathbf{q}^T Q\mathbf{q})^2}{\sum_{\mathbf{q}\in\mathcal{P}}\mathbf{q}^T QAQ\mathbf{q}} = E(Q).$$

It is possible to find a quadric Q minimizing the error $E(Q)$ using a direct method. For that, a symmetric 4×4 matrix Q is represented as a vector of ten coefficients $\mathbf{v} \in \mathbb{R}^{10}$ where the first four components of $\mathbf{v}$ correspond to diagonal entries of Q, the next three components of $\mathbf{v}$ are entries on the 1-diagonal of Q, etc. This allows us to express $\mathbf{q}^T Q\mathbf{q}$ as $\mathbf{v}^T s(\mathbf{q})$ for a suitable 10D vector $s(\mathbf{q})$. Incorporating this into the numerator of $E(Q)$ leads us to the following definition of a matrix $M \in \mathbb{R}^{10\times10}$:

$$\sum_{\mathbf{q}\in\mathcal{P}}(\mathbf{q}^T Q\mathbf{q})^2 = \mathbf{v}^T\left(\sum_{\mathbf{q}\in\mathcal{P}} s(\mathbf{q})s(\mathbf{q})^T\right)\mathbf{v} = \mathbf{v}^T M\mathbf{v}.$$

Similarly, we can define a matrix N s.t. $\sum_{\mathbf{q}\in\mathcal{P}}\mathbf{q}^T QAQ\mathbf{q} = \mathbf{v}^T N\mathbf{v}$.

With this notation at hand, minimizing $E(Q)$ is equivalent to

$$\text{finding } \mathbf{v} \in \mathbb{R}^{10} \text{ minimizing } \frac{\mathbf{v}^T M\mathbf{v}}{\mathbf{v}^T N\mathbf{v}}. \tag{5}$$

The solution vector $\mathbf{v}$ is a generalized eigenvector for $Mx = \mu Nx$ corresponding to the minimal generalized eigenvalue μ. In our implementation, we solve the generalized eigenvalue problem by explicitly computing matrices M and N and applying a few steps of inverse iteration followed by Rayleigh quotient iterations [Golub and Van Loan 1996].

4.4 Choosing the Offset Direction

While any symmetric positive semidefinite matrix Q_o for which $\mathbf{q}^T Q_o\mathbf{q} > 0\ \forall\mathbf{q} \in \mathcal{W}$ can serve as an offset quadric, some matrices produce a better approximation of $\mathcal{W}$ and, as a consequence, result in more reliable rejection of nodes. We illustrate that in Fig. 5, where we compute $\mathcal{F}(Q_0, Q_1)$ using two different offset quadrics. While on the left the family of quadrics does not deviate much from the set $\mathcal{W}$, on the right $\mathcal{F}(Q_0, Q_1)$ covers a lot of irrelevant planes. This example highlights how much the choice of the offset impacts the quality of the approximation $\mathcal{W}^*$.

We use rank-one matrices for ease of representation and define $Q_o = \mathbf{D}\mathbf{D}^T$ for some vector $\mathbf{D} \in \mathbb{R}^4$. Such a matrix is already symmetric positive semidefinite. To ensure that $\mathbf{q}^T Q_o\mathbf{q} > 0\ \forall\mathbf{q} \in \mathcal{W}$, it is sufficient to choose $\mathbf{D}$ s.t. $\mathbf{D}^T\mathbf{q} > 0\ \forall\mathbf{q} \in \mathcal{P}$. Thus, we are looking for a vector $\mathbf{D}$ that satisfies this condition and ensures a good approximation of $\mathcal{W}$.

There are different ways to define which $\mathbf{D}$ is the best choice. In the following, we present a definition that we found to work well in practice. We start by rewriting

$$\mathbf{q}^T Q_\lambda\mathbf{q} = \mathbf{q}^T(Q_f + \lambda\mathbf{D}\mathbf{D}^T)\mathbf{q} = \mathbf{q}^T Q_f\mathbf{q} + \lambda(\mathbf{D}^T\mathbf{q})^2.$$

Given that $\mathbf{q}$ is a plane, $\mathbf{D}^T\mathbf{q}$ can be interpreted as a signed distance from point $\mathbf{D}$ to the plane $\mathbf{q}$. If we treat $\mathbf{D}$ as a point, its last coordinate should be equal to 1. Using $\mathbf{i} = [0, 0, 0, 1]^T$ this constraint can be expressed with the linear equation $\mathbf{D}^T\mathbf{i} = 1$.

To achieve a good approximation of $\mathcal{W}$, the quadrics Q_λ should move fast through all the planes in $\mathcal{P}$ as the parameter λ changes, i.e. $\frac{\partial}{\partial\lambda}\left(\mathbf{q}^T Q_\lambda\mathbf{q}\right) = (\mathbf{D}^T\mathbf{q})^2$ should be large for all planes. Formally, taking into account that $\mathbf{D}^T\mathbf{q} > 0\ \forall\mathbf{q} \in \mathcal{P}$, that can be expressed as finding $\mathbf{D}$ which maximizes $\min\mathbf{D}^T\mathbf{q}$ across all $\mathbf{q} \in \mathcal{P}$. Geometrically, it means that we are looking for a point that has the largest minimal distance to all the planes in $\mathcal{P}$ (Fig. 6 (a)).

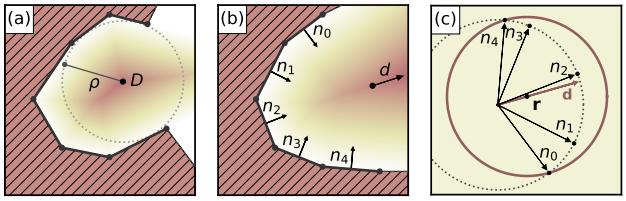


Fig. 6. In (a) and (b), the hatched region illustrates the interior of a mesh. Planes in $\mathcal{P}$ are visualized as black line segments. The value of ρ—distance from a point to the closest plane in $\mathcal{P}$—is shown with a color gradient. (a) For the bounded function ρ, there is a point $\mathbf{D}$ that maximizes ρ. (b) For the unbounded case, there is a unit direction $\mathbf{d}$ in which ρ grows the fastest. The set of vectors $\mathcal{N} = \{\mathbf{n}_0, \ldots, \mathbf{n}_4\}$ consists of unit normals for planes in $\mathcal{P}$. (c) The set $\mathcal{N}$ lies on the unit sphere centered at the origin (dashed line). If $\mathbf{r}$ is the center of the smallest bounding sphere (shown in brown) for the set $\mathcal{N}$, the vector $\mathbf{d}$ can be computed as $\mathbf{d} = \mathbf{r}/\|\mathbf{r}\|$.

This optimization problem is equivalent to a linear programming (LP) problem:

$$\begin{gathered} \text{Find } \rho \in \mathbb{R},\ \mathbf{D} \in \mathbb{R}^4 \text{s.t.} \\ \mathbf{D}^T\mathbf{i} = 1 \text{ and } \mathbf{D}^T\mathbf{q} > \rho,\ \forall \mathbf{q} \in \mathcal{P} \\ \text{maximizing } \rho. \end{gathered} \tag{6}$$

We use the CLP simplex solver [Forrest et al. 2024]—a part of the COIN-OR [Lougee-Heimer 2003] project—to solve problem (6). If it has a finite solution ρ, the optimal vector $\mathbf{D}$ is used to construct the offset quadric. If the problem is infeasible, it implies that for any shading point, there is a backfacing and frontfacing plane and the node always has a silhouette (see Appendix B). Finally, if the farthest point is infinitely far away or, in other words, if the linear programming problem is unbounded, we find a unit direction vector $\mathbf{d}$ with the smallest angle to all the plane normal vectors (Fig. 6 (b)) and use it to form vector $\mathbf{D} = [\mathbf{d}, 0]^T$. Formally, we solve the following optimization problem:

$$\begin{gathered} \text{Find } \mathbf{d} \in \mathbb{R}^3 \text{ s.t.} \\ \mathbf{d}^T\mathbf{d} = 1 \text{ and } \mathbf{d}^T\mathbf{n} > 0,\ \forall \mathbf{n} \in \mathcal{N} \\ \text{maximizing } \min_{\mathbf{n}\in\mathcal{N}} \mathbf{d}^T\mathbf{n}, \end{gathered} \tag{7}$$

where $\mathcal{N}$ refers to the set of all unit normals of planes in $\mathcal{P}$. Since all the plane normals have unit length, problem (7) can be solved by finding the smallest bounding sphere of $\mathcal{N}$ and using the direction to the sphere center as $\mathbf{d}$ (Fig. 6 (c)). We prove this claim in Appendix G.

We solve problem (7) approximately using a variant of the extremal points optimal sphere method proposed by Larsson [2008]. We precompute 16 directions uniformly distributed on a hemisphere and project points from the set $\mathcal{N}$ onto these directions. The set of extremal points consists of at most 32 points and contains points with the largest and the smallest projections for each of the directions. We compute the optimal bounding sphere for the set of extremal points using Welzl's algorithm with move-to-front heuristics [Gärtner 1999; Welzl 1991]. The resulting sphere may not cover the entire set of points $\mathcal{N}$. To obtain a proper bounding sphere, we iterate over all the points in $\mathcal{N}$ and if we encounter a point that is not covered by the current sphere s, compute a new sphere that covers both s and the problematic point. If $\mathbf{c}$ is the center of the final sphere, $\mathbf{D}$ can be computed as $[\mathbf{c}/\|\mathbf{c}\|, 0]^T$.

In practice, for the majority of nodes, the LP problem (6) is unbounded. Thus, to minimize the computational cost of computing $\mathbf{D}$, we start from solving problem (7). If it turns out that it does not have a feasible point, we attempt to solve problem (6). If it is infeasible as well, we conclude that the node contains a silhouette for any shading point and should never be rejected. The computed vector $\mathbf{D}$ is used to construct the offset quadric Q_o.

We also use $\mathbf{D}$ as vector $\mathbf{Z}$ during the construction of the bounding box $\mathcal{B}_q$ (Sec. 3.2). Such a choice of $\mathbf{Z}$ ensures minimal distortion of the set $\mathcal{W}'$ compared to the set $\mathcal{W}$.

5 RESULTS

We start discussion of our results by evaluating the quality of our rejection test as compared to the rejection method proposed by Li [2019] (Sec. 5.1). Next we compare the quality of gradients computed using our algorithm with edge sampling [Li 2019], warped-area sampling [Bangaru et al. 2020; Xu et al. 2023] and projective sampling [Zhang et al. 2023] (Sec. 5.2).

We discuss advantages of explicitly accounting for polygonal lights during the hierarchy traversal in Sec. 5.3, and study the effect of culling concave edges in Sec. 5.4. Sec. 5.5 covers inverse rendering examples, and Sec. 5.6 discusses limitations of our technique.

5.1 Rejection Test Quality

The variance of the boundary integral estimator depends on how closely the importance sampling density matches the integrand. In particular, it depends on how reliably the rejection test identifies nodes that do not contain silhouette edges. We call cases where the rejection test falsely claims that a node contains a silhouette *false accepts*.

Fig. 7 visualizes the accuracy of our rejection test for a single node. We use a spherical patch as an example of simple geometry and a toroidal patch to analyze a more complex scenario. Our method outperforms the approach proposed by Li [2019] in both cases. Additionally, Fig. 7 illustrates the importance of the quadrics approximation for our rejection test. We observe a significant improvement in rejection quality for the mesh patch with more complex geometry, where the bounding box approximation lacks expressiveness.

For the second experiment, we analyze rejection test quality for an entire hierarchy of mesh patches viewed from a single shading point. We traverse the data structure for the chosen point $\mathbf{p}$ and find all the edges that have non-zero probability of being sampled. Ideally, these should be only silhouette edges located in the upper hemisphere for the shading point $\mathbf{p}$. In practice, due to imperfections in the rejection test and data structure construction, edges with zero contribution to the boundary integral can be sampled as well. This negatively impacts the variance of the computed gradients. In Fig. 8, we highlight all the edges with non-zero sampling probability. Our hierarchy traversal samples fewer edges with zero contribution compared to the approach proposed by Li [2019].

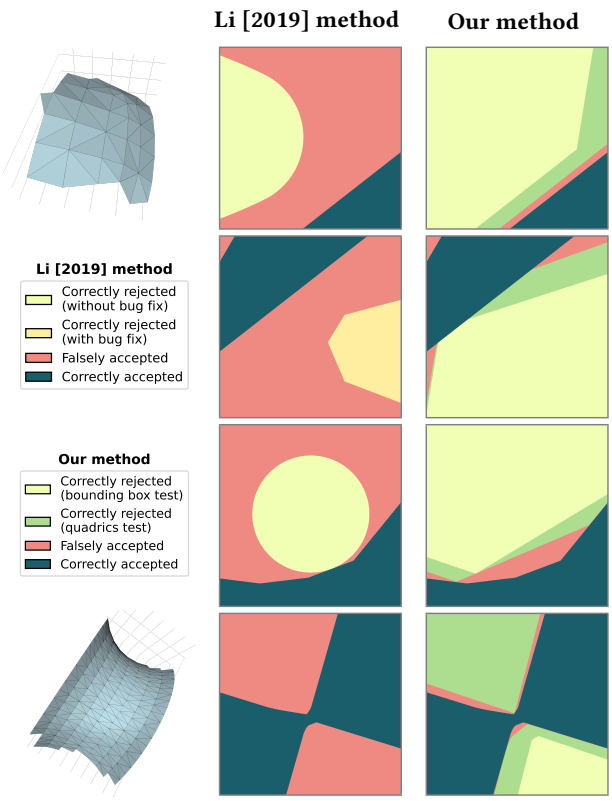


Fig. 7. **Evaluation of rejection test quality.** For each of the two mesh patches shown on the left, we choose two parallel planes that slice the scene at different heights. For each point on these planes, we perform either our or Li's [2019] rejection test and color the point according to its output. The first two rows show results for the mesh patch at the top. For Li's [2019] method, we additionally show the impact of the bug in redner (see Appendix F) on the rejection test quality.

5.2 Comparison with Prior Work

We implement our method in redner—the codebase used in the papers by Li et al. [2018] and Bangaru et al. [2020]. All aspects of path construction except for the silhouette sampling are unchanged: To compute the gradient, a path is sampled using BSDF importance sampling. Per bounce, reflected radiance is estimated using multiple importance sampling (MIS) between BSDF sampling and light sampling, and then stored in a buffer. Computation of the interior term reuses the already constructed path and the precomputed radiance and evaluates the gradient in a similar way as the radiance—using BSDF and light sampling combined with MIS. For the boundary term, silhouette points are sampled at each bounce. In case the sampling was successful, complete boundary paths are generated to evaluate the boundary term. In all of our experiments, we limit the maximal path length to three.

For our comparisons, we use a scene with an object placed in front of a reflecting surface with GGX microfacet BRDF and roughness $\alpha = 0.01$. The scene is illuminated with an environment light. The camera observes the object reflection while the object itself is outside the view frustum. The parameter θ controls the vertical translation

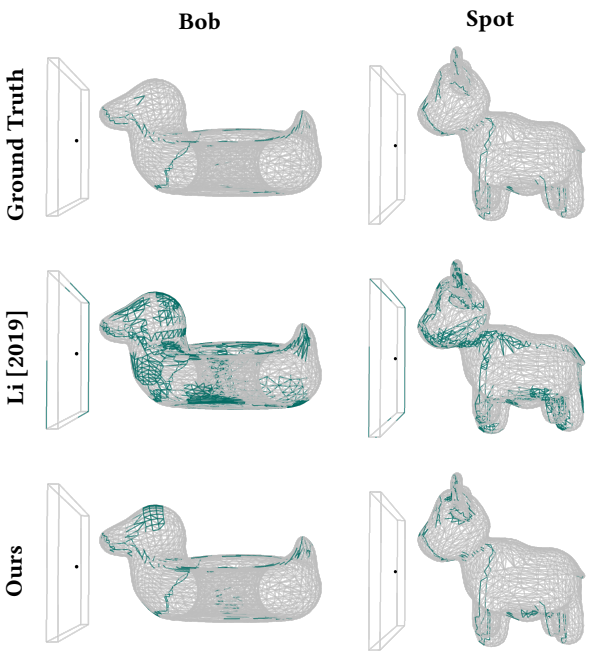


Fig. 8. A scene that consists of an object located in front of a flat surface, visualized as wireframe meshes. For a shading point (in black), we analyze the quality of our rejection test as compared with the method proposed by Li [2019]. All the edges that according to a rejection test may be silhouettes and have non-zero probability of being sampled are plotted in green.

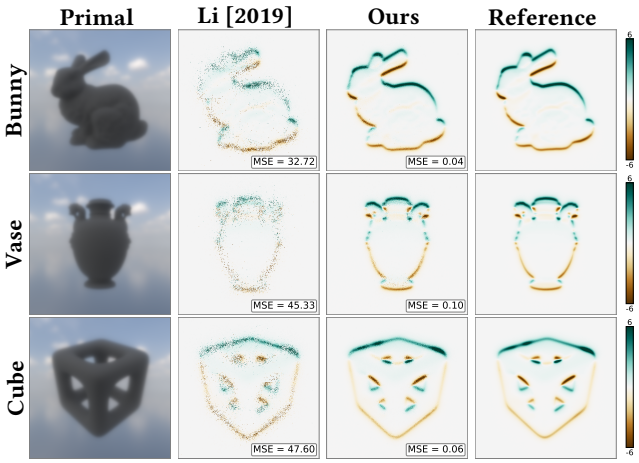


Fig. 9. Equal time comparison of our method with Li [2019]. Each row shows a primal rendering of an object observed through a rough mirror as well as gradient images computed with respect to vertical translation of the object.

of the object. We compute gradients with respect to θ and compare them with finite differences evaluated with 2^{16} samples per pixel.

All experiments are performed on an Intel Xeon Gold 6348 CPU with 16 threads. Throughout this section, we use 128 spp for all results produced by our method.

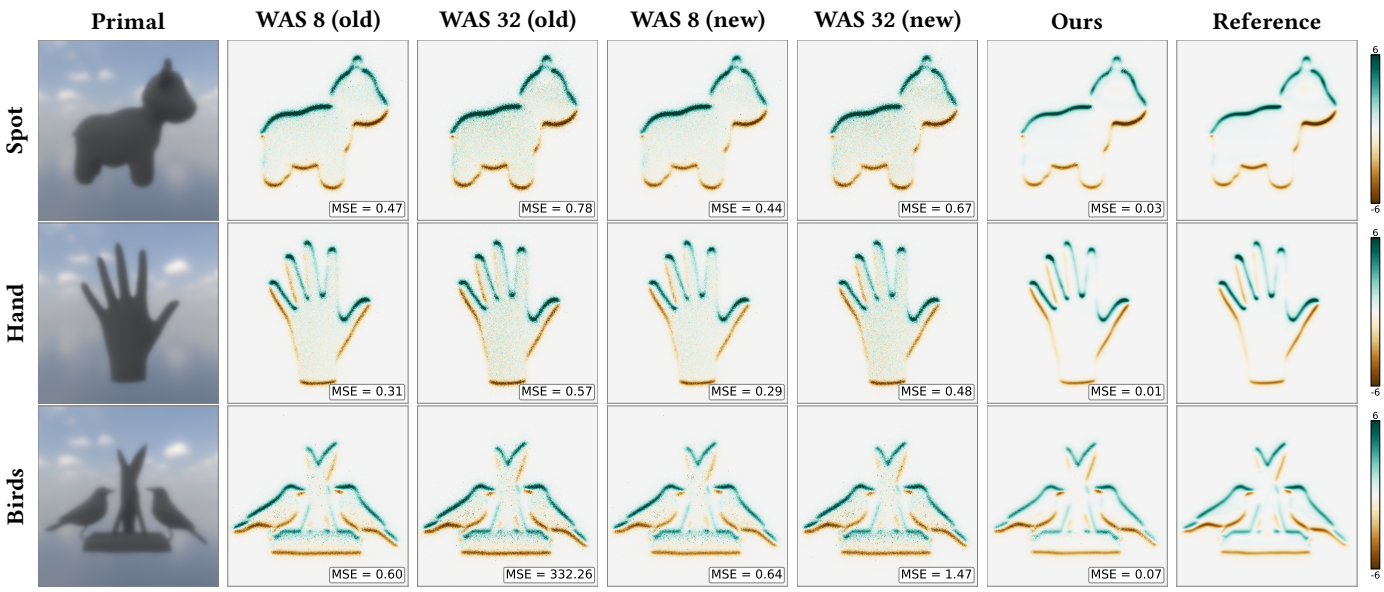


Fig. 10. An equal-time comparison of WAS methods [Bangaru et al. 2020; Xu et al. 2023] with our quadric-based silhouette sampling method. In the primal renderings, the camera observes a reflection of a moving object in a rough mirror. We compute gradients w.r.t. vertical translation of the object using our method, WAS (old) and WAS (new). For WAS methods, we compare gradients evaluated with 8 and 32 auxiliary rays.

Edge Sampling [Li 2019; Li et al. 2018]. Fig. 9 features an equal time comparison of our method to the original implementation of edge sampling [Li 2019]. Data structure construction times and sample counts are reported in the supplemental material. Although our hierarchy construction takes significantly longer, this time is only a fraction of the total gradient computation time. Our experiment shows that using our rejection test and the improved importance function leads to dramatically less noisy gradients at lower sample counts.

Warped-Area Sampling. WAS methods compute the boundary integral by casting it into an area integral. That requires sampling multiple auxiliary rays for each shading point and evaluating a distance function dependent on scene-specific hyperparameters. When the number of auxiliary rays is fixed and not chosen at random, WAS methods produce biased gradients [Bangaru et al. 2020].

Our comparison includes the original WAS [Bangaru et al. 2020] and its modification that uses a different distance function [Xu et al. 2023], referred to as WAS (old) and WAS (new), respectively. Both methods are implemented in redner allowing us to make fair equal-time comparisons to our method. We set the concentration parameter $\kappa = 10^5$ for WAS (old) and $a = 3$ and $\sigma_0 = 0.005$ for WAS (new). We use a corrected version of WAS (new) (see the errata of Xu et al. [2023]). Our comparisons are shown in Fig. 10.

Our method offers decisively higher-quality gradients in equal time (see Fig. 10). Unlike WAS with a fixed number of auxiliary rays, our method is unbiased. And, unlike the unbiased version of WAS, it has predictable execution time and can be parallelized more easily. Additionally, due to our direct approach to computing the boundary integral, there is no additional noise introduced to the interior term of the gradient and there is no need for hyperparameter tuning.

PSDR methods. Path space differentiable rendering methods compute the boundary integral explicitly by sampling silhouette paths. Unlike edge sampling methods that start path construction from the camera, PSDR methods first choose a silhouette segment and then complete it to form a valid light path. To ensure low variance of the gradient estimator, these methods require precomputation of a guiding distribution over all silhouette segments.

In the following, we discuss our approach as compared to the state-of-the-art projective sampling method [Zhang et al. 2023] with octree guiding implemented in Mitsuba 3 [Jakob et al. 2022]. Due to the performance differences between redner and Mitsuba 3, equal-time comparisons are not informative and we opt for equal-sample-budget comparisons instead.

Both methods use samples for two purposes: finding silhouettes, and constructing light paths to compute the silhouette contribution to the gradient. In our method, the number of samples used in these two operations is controlled by only one parameter N. For an image with n_p pixels, we performs Nn_p hierarchy traversals to find a silhouette and trace $2Nn_p$ light paths (two light paths are needed to evaluate ΔL_i) to evaluate edge contributions to the final gradient. The behavior of the projective sampling method, on the other hand, is defined by three parameters: N_{proj}, N_{trial} and N_{sppi}. First, projective sampling performs $N_{\text{proj}} n_p$ projections to find silhouette segments. Then, it builds the octree and for each leaf it traverses $2N_{\text{trial}}$ paths to compute the probability density for the guiding distribution. Finally, it samples the octree $N_{\text{sppi}} n_p$ times and traces $2N_{\text{sppi}} n_p$ light paths to evaluate the gradient induced by indirectly observed discontinuities. In total, if the octree has n_l leaves, projective sampling traverses $2N_{\text{sppi}} n_p + 2N_{\text{trial}} n_l$ light paths and samples a silhouette edge $N_{\text{proj}} n_p$ times.

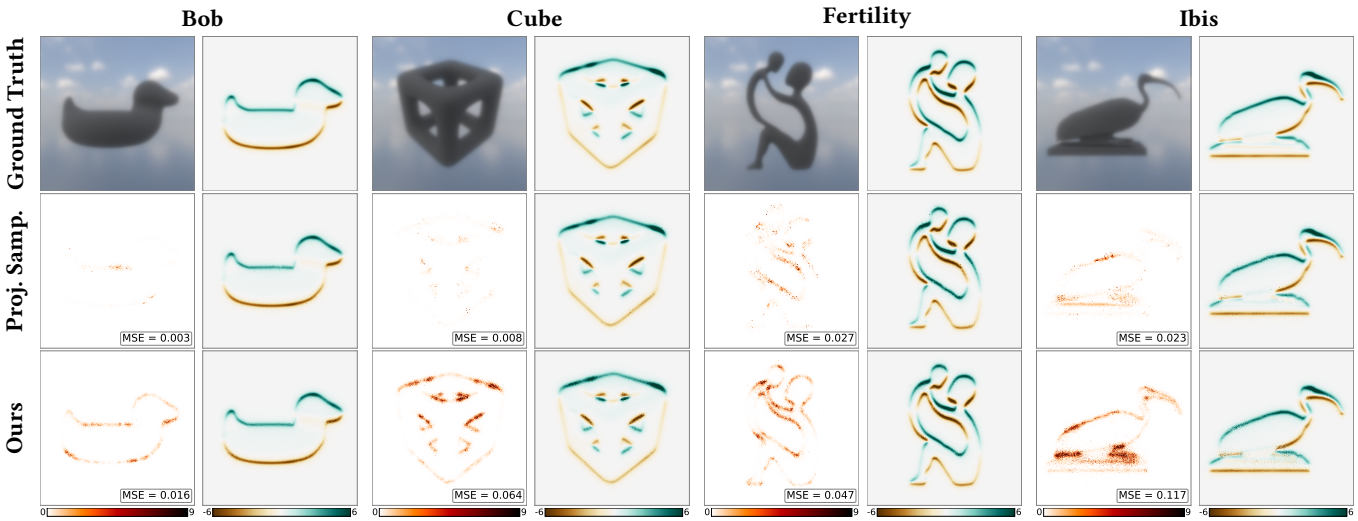


Fig. 11. An equal-sample-budget comparison of our method and the projective sampling approach [Zhang et al. 2023]. Each primal rendering features a reflection of an object in a rough mirror. Our method uses 128 samples per pixel for gradient computation. We set up the projective sampling method so that it traverses the same number of light paths and makes the same number of attempts to find a silhouette point as our method. Since there are multiple sets of parameters that fulfill this condition, we try a few different setups and report the result that produces minimal MSE. See Sec. 5.2 for more details. The red colormap is used to visualize the squared image error.

To make our comparison fair, we choose parameters so that each method gets the same number of attempts to find silhouette segments and an equal number of light paths to traverse. We set $N_{\text{proj}} = N = 128$ and experiment with 15 different values of $N_{\text{sppi}} \in (0, 128)$. The value of N_{trial} is set to satisfy $2Nn_p = 2N_{\text{sppi}}n_p + 2N_{\text{trial}}n_l$. We select the N_{sppi} producing gradients with minimal MSE and report results in Fig. 11. In this experiment, projective sampling outperforms our method. This result is expected, given that projective sampling utilizes a guiding distribution, while our method relies solely on local sampling decisions. We believe that adding guiding to our method would result in significantly improved gradients.

In Table 1, we report the data structure build time and the total gradient computation time for both methods. Our implementation runs 6 to 8.5 times slower. Although this difference is significant, we believe that it can be largely attributed to differences of the two codebases such as the usage of AVX instructions in Mitsuba 3.

Our method relies solely on unidirectional path tracing and samples a silhouette for any given shading point. It is a self-contained technique that can be used as is. However, it can also be incorporated into the projective sampling framework [Zhang et al. 2023]. In projective sampling, silhouette segments are constructed by tracing $N_{\text{proj}}n_p$ paths and applying a projection operation to them. This step could instead use our explicit edge sampling, which has the advantage that the probability of each sampled edge is known.

5.3 Scenes with Polygonal Lights

Sec. 3.6 describes a way to explicitly account for polygonal light sources in a scene. In Fig. 12, we evaluate performance of the modified importance function on a test scene that features an object illuminated with a small polygonal light source and a dim environment light. The object casts a soft shadow onto a diffuse plane placed in front of it and the camera observes this shadow. We use $I(\omega_o, \mathbf{p})$ and $I_{\text{light}}(\omega_o, \mathbf{p})$ as importance functions for hierarchy traversal. At equal sample count (we use 64 spp), using $I_{\text{light}}(\omega_o, \mathbf{p})$ results in a 50× MSE reduction for the Harp scene and a 20× improvement for the Elephant scene.

Table 1. Data structure build time and overall execution time for our method and the projective sampling approach in the equal-sample-budget scenario. These timings were measured on the hardware described in Sec. 5.2.

	Ours		Proj. Sampl.	
	Build time	Overall time	Build time	Overall time
Bob	0.24 s	141.70 s	11.10 s	23.57 s
Cube	1.91 s	149.61 s	16.94 s	31.95 s
Fertility	1.50 s	153.13 s	10.20 s	18.08 s
Ibis	1.32 s	151.66 s	10.83 s	20.17 s

5.4 Concave Edge Culling

For a watertight mesh with opaque BSDF, the concave edges never form a silhouette and can be culled (Sec. 4.1). The culling considerably reduces the number of edges in the hierarchy (see Table 2).

We perform an ablation study to evaluate the effect of the culling on the gradient quality (see Fig. 13). The equal-sample comparison shows that culling improves gradient quality. However, the extent of this improvement depends on the mesh.

5.5 Optimization Results

This section shows an application of our technique to an inverse rendering task where the goal is to infer the shape of an object from a single reference image. The scene has two mirrors (with roughness $\alpha = 0.01$) positioned in such a way that the key geometric

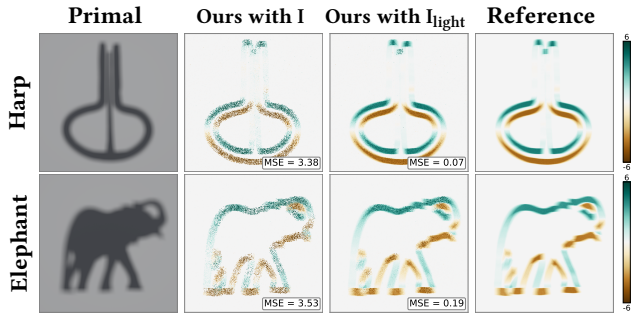


Fig. 12. An equal-sample comparison of gradients computed using either our importance function $I(\omega_o, \mathbf{p})$, which is oblivious to lighting, or $I_{\text{light}}(\omega_o, \mathbf{p})$, which explicitly accounts for polygonal lights. The primal image shows the shadow of an object illuminated by a small polygonal light source. Gradients in the second and third column are evaluated using 64 samples per pixel.

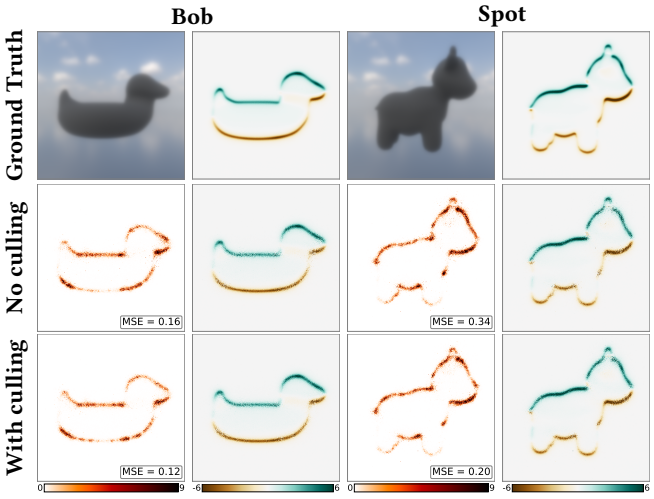


Fig. 13. **Ablation study for concave edge culling.** The scene features an object visible only through reflection in a smooth plane. Gradients of the image w.r.t. vertical translation of the object are evaluated using 16 samples per pixel. The second and third row show derivative images computed without and with concave edge culling, respectively. We show squared errors using a red colormap.

details of the object are visible only in reflections (see Fig. 14). Our optimization minimizes L_2 loss using simple gradient descent. To enforce mesh smoothness, we employ the mesh reparametrization proposed by Nicolet et al. [2021]. During optimization, we remesh the object multiple times to introduce additional vertices and to reconstruct finer details. Fig. 14 shows the object evolution throughout the optimization. We plot renderings of the original scene (top) and renderings of the same mesh in a different scene setup (bottom).

Using a physically-based inverse rendering approach and correctly computing the boundary term allows us to successfully reconstruct the object geometry even though most of the geometric details are visible only indirectly. Our data structure handles the meshes generated by intermediate optimization steps well.

Table 2. Number of edges in the hierarchy with and without concave edge culling for various meshes used in the paper.

	Bob	Spot	Hand	Cube	Bunny	Birds
No culling	13118	8716	23296	71008	11509	92894
With culling	8963	6186	15755	45028	7618	53471

5.6 Limitations

As we demonstrated in our experiments, the proposed method is able to compute gradients for objects of various geometries under different illumination conditions. However, the quality of the gradients is dependent on the complexity of the mesh and the illumination. Fig. 15 shows an example of a geometrically intricate Chandelier scene. We compute derivatives with respect to object translation using the same sample count as used in Figs. 9, 10, and 11. Nonetheless, variance of the gradients is orders of magnitude worse than for meshes with less complex geometry. Scenes with strong directional illumination present a challenge for our method as well since our importance function assumes constant ΔL_i (Sec. 3.5). While these are currently failure cases for our technique, we believe that this can be addressed in future work.

Furthermore, our current implementation has two technical limitations. First, our implementation does not support meshes with sharp or single-sided edges or meshes with self-intersections (Sec. 2.2). Second, our implementation of I_{light} currently supports scenes with only one light source. It is trivial to extend it to support multiple polygonal lights, but to handle large light counts efficiently, additional data structures may be needed.

6 CONCLUSION

In this paper, we propose a new edge sampling method for unbiased computation of the boundary integral in differentiable rendering. In terms of performance, traversal of the proposed data structure is an embarrassingly parallel computation, and the data structure construction can be parallelized using already existing strategies developed for the standard ray-tracing BVHs. Thus, we believe that our implementation can be extended to efficiently run on GPUs.

The problem of discontinuous integrands has received a lot of attention in the last few years. Yet, the most straightforward approach to handling boundary integrals—edge sampling—has remained underexplored. In this paper, we prove that edge sampling is not only a viable approach but is competitive with the leading existing methods. The simplicity of our method and its reliance solely on unidirectional path tracing make it easy to incorporate into existing renderers.

ACKNOWLEDGMENTS

We would like to thank Peiyu Xu for providing the implementation of WAS (new), Nicolas Roussel for his help with the Mitsuba 3 codebase, and Anton Kaplanyan for his support of the project. The Hand (github.com/odedstein/meshes), Golden Star (skfb.ly/oU7Zq), Christmas ornamentsr (skfb.ly/oo7ro), and Nutcracker (skfb.ly/oAWCM) meshes are provided under the CC BY 4.0 license. This work was supported in part by the National Science Foundation under award #2212084.

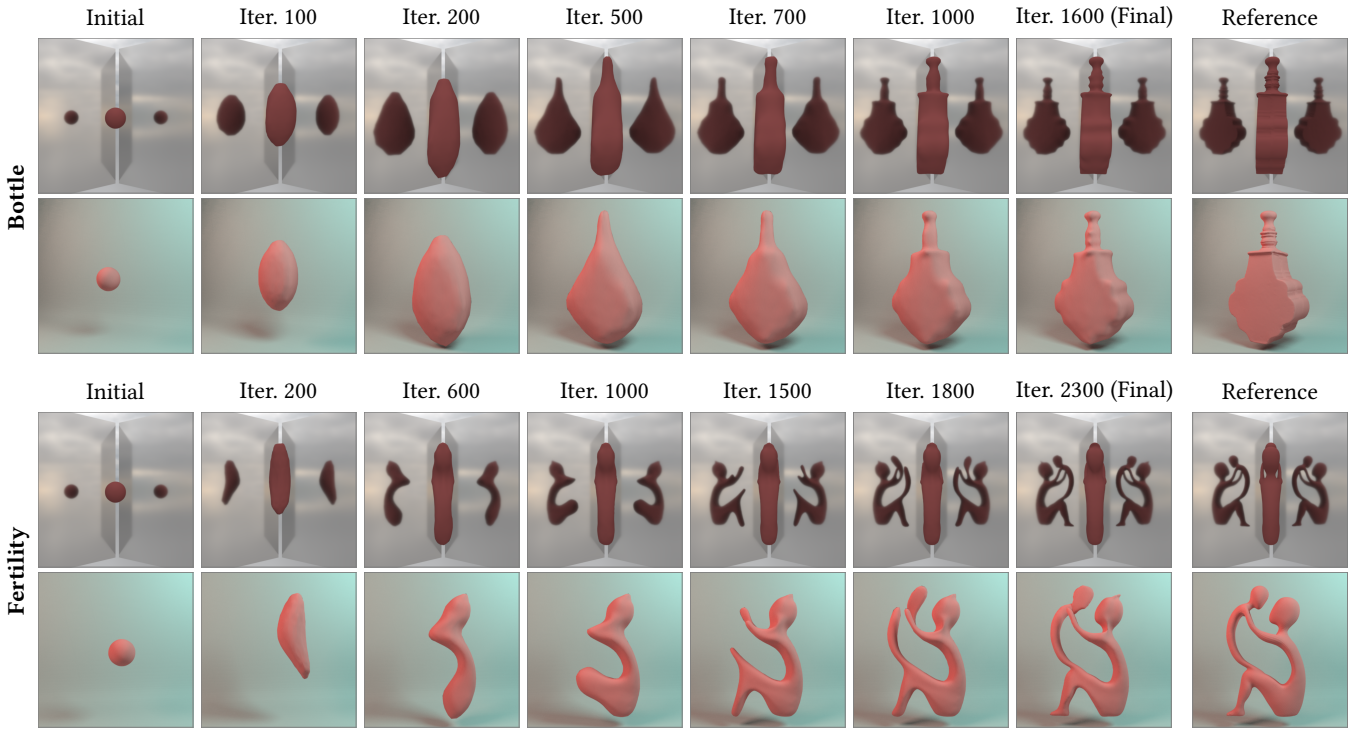


Fig. 14. **Geometry reconstruction results.** We reconstruct the shape of an object given a single reference image of the object and its two reflections (top right). The top row shows shape evolution in the original scene setup. The bottom row shows the object under different illumination conditions.

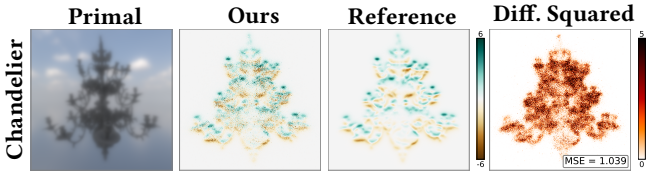


Fig. 15. **Challenging scene.** The Chandelier scene features an object with very complex geometry. Gradients are computed with 128 samples per pixel. Despite using as many samples as in Fig. 9, 10, and 11, our gradient image has significantly higher variance than in the previous experiments.

REFERENCES

Sai Bangaru, Michael Gharbi, Tzu-Mao Li, Fujun Luan, Kalyan Sunkavalli, Milos Hasan, Sai Bi, Zexiang Xu, Gilbert Bernstein, and Fredo Durand. 2022. Differentiable Rendering of Neural SDFs through Reparameterization. In *ACM SIGGRAPH Asia 2022 Conference Proceedings*. Article 22. https://doi.org/10.1145/3550469.3555397

Sai Bangaru, Tzu-Mao Li, and Frédo Durand. 2020. Unbiased Warped-Area Sampling for Differentiable Rendering. *ACM Trans. Graph.* 39, 6 (Nov. 2020). https://doi.org/10.1145/3414685.3417833

Alejandro Conty Estevez and Christopher Kulla. 2018. Importance Sampling of Many Lights with Adaptive Tree Splitting. *Proc. ACM Comput. Graph. Interact. Tech.* 1, 2 (Aug. 2018). https://doi.org/10.1145/3233305

Xi Deng, Fujun Luan, Bruce Walter, Kavita Bala, and Steve Marschner. 2022. Reconstructing Translucent Objects Using Differentiable Rendering. In *ACM SIGGRAPH 2022 Conference Proceedings*. Article 38. https://doi.org/10.1145/3528233.3530714

Xi Deng, Lifan Wu, Bruce Walter, Ravi Ramamoorthi, Eugene d'Eon, Steve Marschner, and Andrea Weidlich. 2024. Reconstructing Translucent Thin Objects from Photos. In *SIGGRAPH Asia 2024 Conference Papers*. Article 124. https://doi.org/10.1145/3680528.3687572

John Forrest, Stefan Vigerske, Ted Ralphs, John Forrest, Lou Hafer, jpfasano, Haroldo Gambini Santos, Jan-Willem, Matthew Saltzman, a-andre, Bjarni Kristjansson, h-i-gassmann, Alan King, Arevall, Bohdan Mart, Pierre Bonami, Ruan Luies, Samuel Brito, and to-st. 2024. *coin-or/Clp: Release releases/1.17.10.* https://doi.org/10.5281/zenodo.13347196

Andrew Gardner, Chris Tchou, Tim Hawkins, and Paul Debevec. 2003. Linear Light Source Reflectometry. *ACM Trans. Graph.* 22, 3 (July 2003). https://doi.org/10.1145/882262.882342

Bernd Gärtner. 1999. Fast and Robust Smallest Enclosing Balls. In *Algorithms - ESA' 99*, Jaroslav Nešetřil (Ed.). Springer Berlin Heidelberg, Berlin, Heidelberg, 325–338.

Gene H. Golub and Charles F. Van Loan. 1996. *Matrix Computations (3rd ed.).* Johns Hopkins University Press, USA, Chapter 8.7.2, 463–465.

Eric Heitz, Jonathan Dupuy, Stephen Hill, and David Neubelt. 2016. Real-Time Polygonal-Light Shading with Linearly Transformed Cosines. *ACM Trans. Graph.* 35, 4 (July 2016). https://doi.org/10.1145/2897824.2925895

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

James T. Kajiya. 1986. The Rendering Equation. *SIGGRAPH Comput. Graph.* 20, 4 (Aug. 1986). https://doi.org/10.1145/15886.15902

Bernhard Kerbl, Georgios Kopanas, Thomas Leimkühler, and George Drettakis. 2023. 3D Gaussian Splatting for Real-Time Radiance Field Rendering. *ACM Trans. Graph.* 42, 4 (July 2023). https://doi.org/10.1145/3592433

Thomas Larsson. 2008. Fast and Tight Fitting Bounding Spheres. In *Proceedings of the Annual SIGRAD Conference, Stockholm : (Linköping Electronic Conference Proceedings, Vol. 34).* 27–30. http://www.ep.liu.se/ecp/034/009/ecp083409.pdf

Tzu-Mao Li. 2019. *Differentiable Visual Computing.* Ph. D. Dissertation. Massachusetts Institute of Technology. Advisor(s) Durand, Frédo.

Tzu-Mao Li, Miika Aittala, Frédo Durand, and Jaakko Lehtinen. 2018. Differentiable Monte Carlo Ray Tracing through Edge Sampling. *ACM Trans. Graph.* 37, 6 (Dec. 2018). https://doi.org/10.1145/3272127.3275109

Guillaume Loubet, Nicolas Holzschuch, and Wenzel Jakob. 2019. Reparameterizing Discontinuous Integrands for Differentiable Rendering. *ACM Trans. Graph.* 38, 6 (Nov. 2019). https://doi.org/10.1145/3355089.3356510

R. Lougee-Heimer. 2003. The Common Optimization INterface for Operations Research: Promoting open-source software in the operations research community. *IBM Journal*

of Research and Development 47, 1 (2003), 57–66. https://doi.org/10.1147/rd.471.0057
Stephen R. Marschner. 1998. *Inverse Rendering for Computer Graphics.* Ph. D. Dissertation. Cornell University. Advisor(s) Greenberg, Donald.
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*.
Pierre Moreau, Matt Pharr, and Petrik Clarberg. 2019. Dynamic Many-Light Sampling for Real-Time Ray Tracing. In *High-Performance Graphics - Short Papers*, Markus Steinberger and Tim Foley (Eds.). The Eurographics Association. https://doi.org/10.2312/hpg.20191191
Baptiste Nicolet, Alec Jacobson, and Wenzel Jakob. 2021. Large Steps in Inverse Rendering of Geometry. *ACM Trans. Graph.* 40, 6 (Dec. 2021). https://doi.org/10.1145/3478513.3480501
Baptiste Nicolet, Felix Wechsler, Jorge Madrid-Wolff, Christophe Moser, and Wenzel Jakob. 2024. Inverse Rendering for Tomographic Volumetric Additive Manufacturing. *ACM Trans. Graph.* 43, 6 (Nov. 2024). https://doi.org/10.1145/3687924
Merlin Nimier-David, Thomas Müller, Alexander Keller, and Wenzel Jakob. 2022. Unbiased Inverse Volume Rendering with Differential Trackers. *ACM Trans. Graph.* 41, 4 (July 2022). https://doi.org/10.1145/3528223.3530073
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 (Aug. 2020). https://doi.org/10.1145/3386569.3392406
Merlin Nimier-David, Delio Vicini, Tizian Zeltner, and Wenzel Jakob. 2019. Mitsuba 2: A Retargetable Forward and Inverse Renderer. *ACM Trans. Graph.* 38, 6 (Dec. 2019). https://doi.org/10.1145/3355089.3356498
Matt Olson and Hao Zhang. 2006. Silhouette Extraction in Hough Space. *Computer Graphics Forum* 25, 3 (2006). https://doi.org/10.1111/j.1467-8659.2006.00946.x
Christoph Peters. 2021. BRDF Importance Sampling for Linear Lights. *Computer Graphics Forum* 40, 8 (2021). https://doi.org/10.1111/cgf.14379
H. Petryk and Zenon Mróz. 1986. Time Derivatives of Integrals and Functionals Defined on Varying Volume and Surface Domains. *Archives of Mechanics* 38 (Jan. 1986).
Matt Pharr, Wenzel Jakob, and Greg Humphreys. 2023. *Physically Based Rendering: From Theory to Implementation.* MIT Press.
Jürgen Richter-Gebert. 2011. *Homogeneous Coordinates.* Springer Berlin Heidelberg, Berlin, Heidelberg, 47–66. https://doi.org/10.1007/978-3-642-17286-1_3
Pedro V. Sander, Hugues Hoppe, John Snyder, and Steven J. Gortler. 2001. Discontinuity Edge Overdraw. In *Proceedings of the 2001 Symposium on Interactive 3D Graphics (I3D '01).* https://doi.org/10.1145/364338.364390
G. Taubin. 1991. Estimation of Planar Curves, Surfaces, and Nonplanar Space Curves Defined by Implicit Equations with Applications to Edge and Range Image Segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 13, 11 (1991), 1115–1138. https://doi.org/10.1109/34.103273
Panagiotis Tsiapkolis and Pierre Bénard. 2024. Patch Decomposition for Efficient Mesh Contours Extraction. *Computer Graphics Forum* 43, 4 (2024). https://doi.org/10.1111/cgf.15154
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 (July 2021). https://doi.org/10.1145/3450626.3459804
Delio Vicini, Sébastien Speierer, and Wenzel Jakob. 2022. Differentiable Signed Distance Function Rendering. *ACM Trans. Graph.* 41, 4 (July 2022). https://doi.org/10.1145/3528223.3530139
Emo Welzl. 1991. Smallest Enclosing Disks (Balls and Ellipsoids). In *New Results and New Trends in Computer Science*, Hermann Maurer (Ed.). Springer Berlin Heidelberg, Berlin, Heidelberg, 359–370.
Robert Woodham. 1992. Photometric Method for Determining Surface Orientation from Multiple Images. *Optical Engineering* 19 (01 1992). https://doi.org/10.1117/12.7972479
Peiyu Xu, Sai Bangaru, Tzu-Mao Li, and Shuang Zhao. 2023. Warped-Area Reparameterization of Differential Path Integrals. *ACM Trans. Graph.* 42, 6 (Dec. 2023). https://doi.org/10.1145/3618330
Kai Yan, Christoph Lassner, Brian Budge, Zhao Dong, and Shuang Zhao. 2022. Efficient Estimation of Boundary Integrals for Path-Space Differentiable Rendering. *ACM Trans. Graph.* 41, 4 (July 2022). https://doi.org/10.1145/3528223.3530080
Cheng Zhang, Bailey Miller, Kai Yan, Ioannis Gkioulekas, and Shuang Zhao. 2020. Path-Space Differentiable Rendering. *ACM Trans. Graph.* 39, 4 (Aug. 2020). https://doi.org/10.1145/3386569.3392383
Cheng Zhang, Lifan Wu, Changxi Zheng, Ioannis Gkioulekas, Ravi Ramamoorthi, and Shuang Zhao. 2019. A Differential Theory of Radiative Transfer. *ACM Trans. Graph.* 38, 6 (Nov. 2019). https://doi.org/10.1145/3355089.3356522
Ziyi Zhang, Nicolas Roussel, and Wenzel Jakob. 2023. Projective Sampling for Differentiable Rendering of Geometry. *ACM Trans. Graph.* 42, 6 (Dec. 2023). https://doi.org/10.1145/3618385

A DERIVATION OF THE BOUNDARY TERM

Our goal is to evaluate the derivative of the outgoing radiance L_o with respect to a scene parameter θ, i.e. to compute

$$\frac{\partial}{\partial\theta}L_o(\omega_o,\mathbf{p}) = \frac{\partial}{\partial\theta}\int_{\mathcal{H}^2} L_i(\omega_i,\mathbf{p})f_s(\omega_i,\omega_o,\mathbf{p})\mathrm{d}\omega_i. \tag{8}$$

For readability we use $F(\omega_i) = L_i(\omega_i,\mathbf{p})f_s(\omega_i,\omega_o,\mathbf{p})$.

For each **visible** triangle T_i in a scene, we define a spherical polygon P_i which is a projection of the visible part of T_i onto the hemisphere $\mathcal{H}^2$. We use P_* to refer to the part of the hemisphere that is not covered by any triangle, i.e. we use $P_* = \mathcal{H}^2\backslash(\cup P_i)$. The set $\mathcal{P} = \{P_*, P_0, P_1, ...\}$ covers the entire hemisphere $\mathcal{H}^2$ with disjoint spherical polygons. This allows us to rewrite Eq. 8 as

$$\int_{\mathcal{H}^2} F(\omega_i)\mathrm{d}\omega_i = \sum_{P\in\mathcal{P}}\int_P F(\omega_i)\mathrm{d}\omega_i.$$

Now, since every P is a regular surface and since function F is sufficiently smooth on each P, we can use the generalization of Reynolds transport theorem to surface domains [Petryk and Mróz 1986] and write that

$$\begin{aligned}\frac{\partial}{\partial\theta}\int_P F(\omega_i)\mathrm{d}\omega_i = \int_P \frac{\partial}{\partial\theta}F(\omega_i)\mathrm{d}\omega_i - \\ 2\int_P F(\omega_i)V_nK_m\mathrm{d}\omega_i + \oint_{\partial P} FV_\mu \mathrm{d}l,\end{aligned} \tag{9}$$

where we use ∂P to refer to the boundary line of P and K_m to denote mean curvature of the surface. $\mathbf{V}$ stands for the velocity of ∂P with respect to change in parameter θ. V_n denotes the component of $\mathbf{V}$ normal to the surface P, and $V_\mu = \mathbf{V}^T\mu$, where μ is the direction tangential to P and normal to ∂P. In Eq. 9, $\mathrm{d}l$ is the arc length of the curve.

Since spherical polygons move only along the hemispherical surface, $V_n = 0$ and Eq. 9 simplifies to

$$\frac{\partial}{\partial\theta}\int_P F(\omega_i)\mathrm{d}\omega_i = \int_P \frac{\partial}{\partial\theta}F(\omega_i)\mathrm{d}\omega_i + \oint_{\partial P} FV_\mu \mathrm{d}l.$$

Summing these equations for all the spherical polygons brings us to the expression

$$\frac{\partial}{\partial\theta}\int_{\mathcal{H}^2} F(\omega_i)\mathrm{d}\omega_i = \int_{\mathcal{H}^2} \frac{\partial}{\partial\theta}F(\omega_i)\mathrm{d}\omega_i + \sum_{P\in\mathcal{P}}\oint_{\partial P} FV_\mu \mathrm{d}l. \tag{10}$$

The first term in Eq. 10 is the interior term (Eq. 2). In the following, we prove that the second term equals the boundary term (Eq. 3).

We use $\partial\mathcal{H}^2$ to refer to the boundary of the hemisphere $\mathcal{H}^2$. Every **visible** edge e, when projected to the hemisphere, forms an arc $A(e)$. We assume that all the arcs are consistently oriented, i.e. that $\det(\mathbf{n}, \mathbf{m}_0(A(e)), \mathbf{m}_1(A(e))) > 0$, where $\mathbf{m}_0(A(e))$ and $\mathbf{m}_1(A(e))$ are the two endpoints of the arc. The set $\mathcal{A} = \cup A(e)$ is the union of projections of all visible edges.

Each boundary ∂P consists of multiple arc segments which either belong to $\partial\mathcal{H}^2$ or they are one of the arcs in $\mathcal{A}$. Since $\mathcal{P}$ covers the whole hemisphere, every arc in $\mathcal{A}$ appears in the sum $\sum_{P\in\mathcal{P}}\oint_{\partial P} fV_\mu \mathrm{d}l$ twice but with different signs. With this in mind the sum can be computed as

$$\sum_{P\in\mathcal{P}}\oint_{\partial P} FV_\mu \mathrm{d}l = \oint_{\partial\mathcal{H}^2} FV_\mu \mathrm{d}l + \sum_{A\in\mathcal{A}}\int_A (F^+V_\mu^+ - F^-V_\mu^-)\mathrm{d}l,$$

where $F^+V_\mu^+$ and $F^-V_\mu^-$ are values of F and V_μ to two sides of the boundary. In our case, $V_\mu^+ = V_\mu^-$. Moreover, because of the cosine foreshortening term, $F = 0$ on $\partial\mathcal{H}^2$, and the sum can be further simplified:

$$\sum_{P\in\mathcal{P}} \oint_{\partial P} FV_\mu \mathrm{d}l = \sum_{A\in\mathcal{A}} \int_A \Delta F V_\mu \mathrm{d}l.$$

Here we used $V_\mu := V_\mu^+ = V_\mu^-$ and $\Delta F := F^+ - F^-$.

Next, we notice that ΔF is non-zero only when the function $F(\omega_i) = L_i(\omega_i, \mathbf{p}) f_s(\omega_i, \omega_o, \mathbf{p})$ is discontinuous across the arc. Since the BRDF f_s is a continuous function, it implies that incoming radiance L_i should be discontinuous across the arc and, as a consequence, across the corresponding edge. This happens either for silhouette edges, where L_i is discontinuous because of the visibility term, or for sharp edges, where normals are discontinuous. Since we assume that our meshes do not have sharp edges, we conclude that

$$\begin{aligned}\sum_{P\in\mathcal{P}} \oint_{\partial P} FV_\mu \mathrm{d}l = \sum_{A\in\mathcal{SA}} \int_A \Delta F V_\mu \mathrm{d}l = \\ \sum_{A\in\mathcal{SA}} \int_A \Delta L_i(\omega_i, \mathbf{p}) f_s(\omega_i, \omega_o, \mathbf{p}) V_\mu \mathrm{d}l,\end{aligned} \tag{11}$$

where we use $\mathcal{SA}$ to refer to the silhouette arcs of the mesh.

The final step is computing V_μ. Let arc A be a projection of a silhouette edge e with endpoints $\mathbf{w}_0$ and $\mathbf{w}_1$. For other points on the edge, we use $\mathbf{w}_t = \mathbf{w}_0 + t(\mathbf{w}_1 - \mathbf{w}_0)$. Velocity of the point $\mathbf{w}_t$ with respect to parameter θ is denoted as $\mathbf{v}_t = \partial_\theta \mathbf{w}_t$. The projection of $\mathbf{w}_t$ onto the hemisphere can be computed as

$$\mathbf{m}_t = \frac{\mathbf{w}_t}{\|\mathbf{w}_t\|} = \frac{\mathbf{w}_0 + t(\mathbf{w}_1 - \mathbf{w}_0)}{\|\mathbf{w}_0 + t(\mathbf{w}_1 - \mathbf{w}_0)\|}.$$

To compute the velocity $\mathbf{V}$ of $\mathbf{m}_t$, we differentiate the expression above with respect to θ:

$$\mathbf{V} = \frac{\partial}{\partial\theta}\mathbf{m}_t = \frac{\mathbf{v}_t}{\|\mathbf{w}_t\|} - \frac{\mathbf{w}_t(\mathbf{v}_t^T\mathbf{w}_t)}{\|\mathbf{w}_t\|^3}.$$

The next step is deriving an expression for μ. We start from computing $\partial_t \mathbf{m}_t$:

$$\partial_t \mathbf{m}_t = \frac{\mathbf{w}_1 - \mathbf{w}_0}{\|\mathbf{w}_t\|} - \frac{\mathbf{w}_t(\mathbf{w}_t^T(\mathbf{w}_1 - \mathbf{w}_0))}{\|\mathbf{w}_t\|^3}$$

and define μ as

$$\mu = \mathbf{m}_t \times \frac{\partial_t \mathbf{m}_t}{\|\partial_t \mathbf{m}_t\|} = \frac{\mathbf{m}_t \times (\mathbf{w}_1 - \mathbf{w}_0)}{\|\mathbf{w}_t\|\|\partial_t \mathbf{m}_t\|} - 0.$$

Now we can project $\mathbf{V}$ onto μ:

$$\begin{aligned}V_\mu = \mathbf{V}^T\mu = \frac{\mathbf{v}_t^T\mu}{\|\mathbf{w}_t\|} - 0 = \frac{\mathbf{v}_t^T(\mathbf{m}_t \times (\mathbf{w}_1 - \mathbf{w}_0))}{\|\mathbf{w}_t\|^2\|\partial_t \mathbf{m}_t\|} = \\ = \frac{\mathbf{v}_t^T(\mathbf{w}_t \times (\mathbf{w}_1 - \mathbf{w}_0))}{\|\mathbf{w}_t\|^3\|\partial_t \mathbf{m}_t\|} = \frac{\mathbf{v}_t^T(\mathbf{w}_0 \times \mathbf{w}_1)}{\|\mathbf{w}_t\|^3\|\partial_t \mathbf{m}_t\|}.\end{aligned}$$

Finally, substituting the expression for V_μ into Eq. 11 and replacing $\mathrm{d}l$ with $\|\partial_t \mathbf{m}_t\|\mathrm{d}t$, we obtain

$$\int_A FV_\mu \mathrm{d}l = \int_0^1 \Delta L_i(\omega_i, \mathbf{p}) f_s(\omega_i, \omega_o, \mathbf{p}) \frac{\det(\mathbf{v}_t, \mathbf{w}_0, \mathbf{w}_1)}{\|\mathbf{w}_t\|^3} \mathrm{d}t. \tag{12}$$

Summing up expressions from Eq. 12 for each silhouette arc, we obtain the boundary term from Eq. 3.

B UNREJECTABLE MESH PATCHES

During the hierarchy construction, for each mesh patch with triangle planes $\mathcal{P}$, we attempt to compute a vector $\mathbf{Z}$ such that $\mathbf{Z}^T\mathbf{q} > 0$ $\forall \mathbf{q} \in \mathcal{P}$ (Sec. 4.4). If no such vector is found, the node is marked accordingly and never rejected during hierarchy traversal.

If the mesh patch is connected, this choice is optimal as there is always a silhouette edge. Indeed, for any shading point $\mathbf{p}$ with homogeneous coordinates $\mathbf{Z} = [\mathbf{p}, 1]^T$, there must be two planes $\mathbf{q}_0$ and $\mathbf{q}_1$ such that $(\mathbf{Z}^T\mathbf{q}_0)(\mathbf{Z}^T\mathbf{q}_1) \le 0$. Since the mesh patch is connected, there must be two planes $\mathbf{q}_0(e)$ and $\mathbf{q}_1(e)$ adjacent to the same edge e s.t. $(\mathbf{Z}^T\mathbf{q}_0(e))(\mathbf{Z}^T\mathbf{q}_1(e)) \le 0$, implying that e is a silhouette edge for $\mathbf{p}$.

C POLYGON-QUADRIC INTERSECTION

Cond. (3) of the rejection test (see Sec. 3.4) requires us to establish whether the quadric Q intersects interior of the polygon C given that Q does not intersect the boundary of the polygon. In this section, we explain how to efficiently check for the intersection.

First, we notice that every point of the polygon C can be represented as $\mathbf{q}(\alpha_1, \alpha_2) = \alpha_1\mathbf{q}_1 + \alpha_2\mathbf{q}_2 + \mathbf{q}_3 \in \mathbb{R}^4$, where $\mathbf{q}_3 = \mathbf{Z}$, and $\mathbf{q}_1, \mathbf{q}_2$ are two linearly independent vectors orthogonal to $\mathbf{Z}$. Each vertex v_i of the polygon C can now be represented with a vector of coefficients $v_i^* = [v_{i,1}^*, v_{i,2}^*, 1]^T$ s.t. $v_i = v_{i,1}^*\mathbf{q}_1 + v_{i,2}^*\mathbf{q}_2 + \mathbf{q}_3$. We use C^* to refer to the polygon with vertices v_i^*.

A point $\mathbf{q}(\alpha_1, \alpha_2)$ lies on the quadric Q when α_1, α_2 satisfy

$$[\alpha_1, \alpha_2, 1] C_Q [\alpha_1, \alpha_2, 1]^T = 0,$$

where $C_Q[i, j] = \mathbf{q}_i^T Q \mathbf{q}_j$; or in other words, when point $\mathbf{a}$ with homogeneous coordinates $[\alpha_1, \alpha_2, 1]^T$ lies on the conic with matrix C_Q. Now, the problem of intersecting polygon C with quadric Q is reduced to the problem of intersecting conic C_Q with polygon C^*.

Since Q does not intersect the boundary of the polygon C, the conic C_Q does not intersect the boundary of C^*. A curve that intersects interior of a polygon but not its boundary must have a bounded component entirely contained within the polygon. Ellipses are the only conics that have a bounded component. Since an ellipse has only one component overall, it means that for a conic to intersect the interior of the polygon without intersecting the boundary, the conic must lie entirely within the polygon. Thus, to check for intersection of C^* and C_Q, it is sufficient to find some point lying on C_Q and verify that it is in the interior of C^*.

To find a point on the conic C_Q we choose a line with homogeneous coordinates $\mathbf{l} \in \mathbb{R}^3$ and a point $\mathbf{r}$ lying on it, i.e. $\mathbf{r}^T\mathbf{l} = 0$. We intersect C with $\mathbf{l}$ and with $\mathbf{l}' = C_Q\mathbf{r}$. If an intersection with either one of the lines is found, we use it as an example of a point on the conic. In the following, we prove that if the conic has real points, at least one intersection must exist. For that, assume that neither $\mathbf{l}$ nor $\mathbf{l}'$ intersect the conic C_Q.

First, we prove that $\mathbf{l}$ and $\mathbf{l}'$ are two distinct lines. Assume that it is not true, and there is $\alpha \neq 0$ s.t. $\mathbf{l} = \alpha\mathbf{l}'$. Then, since $\mathbf{r}^T\mathbf{l} = 0$ it also holds that $\alpha\mathbf{r}^T\mathbf{l}' = 0$, which implies that $\mathbf{r}^T C_Q \mathbf{r} = 0$. The last equality contradicts with the assumption that C_Q does not intersect $\mathbf{l}$.

Since the two lines are distinct, there exists a single point $\mathbf{q}$ (up to scaling) s.t. $\mathbf{q}^T\mathbf{l} = 0$ and $\mathbf{q}^T\mathbf{l}' = 0$. The point $\mathbf{q}$ can be a finite point if

$\mathbf{l}$ and $\mathbf{l}'$ intersect, as well as a point at infinity if the lines are parallel. Moreover, $\mathbf{q} \neq \mathbf{r}$ since we previously proved that $\mathbf{r}^T\mathbf{l}' = \mathbf{r}^T C_Q \mathbf{r} \neq 0$.

Any point $\mathbf{p}$ on the line $\mathbf{l}$ can be expressed as $\mathbf{p} = \lambda\mathbf{r} + \mu\mathbf{q}$. The line $\mathbf{l}$ intersects the conic C_Q iff there exist μ and λ s.t. $(\lambda\mathbf{r} + \mu\mathbf{q})C_Q(\lambda\mathbf{r} + \mu\mathbf{q}) = 0$ and $\lambda^2 + \mu^2 \neq 0$. This is equivalent to requiring that

$$\det \begin{pmatrix} \mathbf{q}^T C_Q \mathbf{q} & \mathbf{q}^T C_Q \mathbf{r} \\ \mathbf{q}^T C_Q \mathbf{r} & \mathbf{r}^T C_Q \mathbf{r} \end{pmatrix} \geq 0.$$

Since $\mathbf{l}$ does not intersect the conic, and since $\mathbf{q}^T C_Q \mathbf{r} = \mathbf{q}^T \mathbf{l}' = 0$, it holds that $(\mathbf{q}^T C_Q \mathbf{q})(\mathbf{r}^T C_Q \mathbf{r}) < 0$.

Similarly, any point on $\mathbf{l}'$ can be expressed as $\lambda\mathbf{s} + \mu\mathbf{q}$, for some point $\mathbf{s} \neq \mathbf{q}$ lying on $\mathbf{l}'$. Since $\mathbf{l}'$ does not intersect C_Q, it follows that

$$\det \begin{pmatrix} \mathbf{q}^T C_Q \mathbf{q} & \mathbf{q}^T C_Q \mathbf{s} \\ \mathbf{q}^T C_Q \mathbf{s} & \mathbf{s}^T C_Q \mathbf{s} \end{pmatrix} < 0.$$

Now we notice that

$$\det \begin{pmatrix} \mathbf{q}^T C_Q \mathbf{q} & \mathbf{q}^T C_Q (\beta\mathbf{s} + \gamma\mathbf{r}) \\ \mathbf{q}^T C_Q (\beta\mathbf{s} + \gamma\mathbf{r}) & (\beta\mathbf{s} + \gamma\mathbf{r})^T C_Q (\beta\mathbf{s} + \gamma\mathbf{r}) \end{pmatrix} =$$
$$\beta^2 \det \begin{pmatrix} \mathbf{q}^T C_Q \mathbf{q} & \mathbf{q}^T C_Q \mathbf{s} \\ \mathbf{q}^T C_Q \mathbf{s} & \mathbf{s}^T C_Q \mathbf{s} \end{pmatrix} + \gamma^2 (\mathbf{q}^T C_Q \mathbf{q})(\mathbf{r}^T C_Q \mathbf{r}) < 0.$$

Therefore, for any β and γ, the line that goes through $\mathbf{q}$ and $(\beta\mathbf{s} + \gamma\mathbf{r})$ does not intersect C_Q. Equivalently, any line that goes through $\mathbf{q}$ does not intersect the conic C_Q, implying that there are no real points that belong to C_Q. With this contradiction, we conclude our derivation and state that either $\mathbf{l}$ or $\mathbf{l}'$ intersect the conic C_Q.

D COMPUTING BOUNDING QUADRICS

In Sec. 4.2, we discuss that to construct the bounding quadrics Q_0 and Q_1, it is necessary to find extrema of $\lambda(\mathbf{q}(t))$, where

$$\lambda(\mathbf{q}) = -\frac{\mathbf{q}^T Q_f \mathbf{q}}{\mathbf{q}^T Q_o \mathbf{q}}, \quad \mathbf{q}(t) = (1-t)\mathbf{q}_0 + t\mathbf{q}_1,$$

and $\mathbf{q}_0$, $\mathbf{q}_1$ are two planes forming an edge. We assume that Q_o is a rank-1 positive semidefinite matrix s.t. $\mathbf{q}(t)^T Q_o \mathbf{q}(t) > 0$, $\forall t \in [0, 1]$.

As noted before, $\lambda(\mathbf{q}(t))$ is a differentiable function of t and it achieves its extremal values either at $t = 0$, at $t = 1$, or at a critical point $t^* \in (0, 1)$ if such a point exists. Explicitly computing t^* is possible, but the resulting expressions are complicated and hard to analyze. Thus, we directly evaluate $\lambda^* = \lambda(\mathbf{q}(t^*))$.

For that, we consider the function

$$g(\lambda, \mathbf{q}) = \mathbf{q}^T Q_\lambda \mathbf{q}, \text{ where } Q_\lambda = Q_f + \lambda Q_o.$$

Given the choice of the offset quadric, for any $\mathbf{q} = \mathbf{q}(t)$, $t \in (0, 1)$, the function $g(\lambda, \mathbf{q})$ is a strictly monotonically increasing linear function of λ that equals zero at $\lambda(\mathbf{q}(t))$. Therefore, the zero set of the function $g(\lambda, \mathbf{q})$ for $t \in (0, 1)$ is

$$\Lambda_0 = \left\{\lambda \mid \exists t \in (0,1) \text{ s.t. } g(\lambda, \mathbf{q}(t)) = 0\right\} = \{\lambda(\mathbf{q}(t)) \mid t \in (0,1)\}.$$

This observation leads us to the conclusion that computing λ^* is equivalent to computing an extremum (if it exists) of the set Λ_0.

For a fixed λ, $g(\lambda, \mathbf{q}(t))$ is a quadratic function of t:

$$g(\lambda, \mathbf{q}(t)) = t^2(\mathbf{q}_1 - \mathbf{q}_0)^T Q_\lambda (\mathbf{q}_1 - \mathbf{q}_0) + 2t\mathbf{q}_0^T Q_\lambda (\mathbf{q}_1 - \mathbf{q}_0) + \mathbf{q}_0^T Q_\lambda \mathbf{q}_0.$$

The set Λ_0 only contains λ for which this parabola has real roots in $(0, 1)$. A value λ is an extremum of Λ_0 if and only if there is **no** neighborhood $E = (\lambda - \epsilon, \lambda + \epsilon)$ s.t. $\forall \lambda' \in E$ the function $g(\lambda', \mathbf{q}(t))$ has real roots. Considering that $g(\lambda, \mathbf{q}(t))$ is a continuous function, it implies that if for some λ the parabola $g(\lambda, \mathbf{q}(t))$ has two real roots, this λ cannot be an extremum of Λ_0. Now we know that if λ is an extremum of Λ_0 then $g(\lambda, \mathbf{q}(t))$ has a single root $t^* \in (0, 1)$. The opposite also holds: assume that for some λ^*, the parabola $g(\lambda, \mathbf{q}(t))$ has a single root $t^* \in (0, 1)$. This implies that either $g(\lambda^*, \mathbf{q}(t)) \leq 0$ $\forall t$ or $g(\lambda^*, \mathbf{q}(t)) \geq 0$ $\forall t$. We will start from the first case. Taking into account that $g(\lambda, \mathbf{q}(t))$ is a strictly increasing function of λ, we state that for any $\lambda' < \lambda^*$ and any t it holds that $g(\lambda', \mathbf{q}(t)) < 0$. This means that $\lambda' \notin \Lambda_0$ and that $\lambda^* = \min \Lambda_0$. In the second case, similar derivations can be used to show that $\lambda^* = \max \Lambda_0$.

Similar reasoning can be used to explain that if for some λ parabola $g(\lambda, \mathbf{q}(t))$ has a leading coefficient equal to zero, this λ is not an extremum. Thus, in the following, we only consider the case where $(\mathbf{q}_1 - \mathbf{q}_0)^T Q_\lambda (\mathbf{q}_1 - \mathbf{q}_0) \neq 0$.

A parabola has a single root when its discriminant is equal to zero. In our case, it should hold that:

$$(\mathbf{q}_0^T Q_\lambda (\mathbf{q}_1 - \mathbf{q}_0))^2 - \left(\mathbf{q}_0^T Q_\lambda \mathbf{q}_0\right)\left((\mathbf{q}_1 - \mathbf{q}_0)^T Q_\lambda (\mathbf{q}_1 - \mathbf{q}_0)\right) = 0 \Leftrightarrow$$
$$(\mathbf{q}_0^T Q_\lambda \mathbf{q}_1)^2 = (\mathbf{q}_0^T Q_\lambda \mathbf{q}_0)(\mathbf{q}_1^T Q_\lambda \mathbf{q}_1) \tag{13}$$

Eq. 13 is equivalent to

$$(\mathbf{q}_0^T Q_f \mathbf{q}_1)^2 + 2\lambda(\mathbf{q}_0^T Q_f \mathbf{q}_1)(\mathbf{q}_0^T Q_o \mathbf{q}_1) + \lambda^2(\mathbf{q}_0^T Q_o \mathbf{q}_1)^2 =$$
$$(\mathbf{q}_0^T Q_f \mathbf{q}_0)(\mathbf{q}_1^T Q_f \mathbf{q}_1) + \lambda^2(\mathbf{q}_0^T Q_o \mathbf{q}_0)(\mathbf{q}_1^T Q_o \mathbf{q}_1) + \tag{14}$$
$$\lambda((\mathbf{q}_0^T Q_f \mathbf{q}_0)(\mathbf{q}_1^T Q_o \mathbf{q}_1) + (\mathbf{q}_0^T Q_o \mathbf{q}_0)(\mathbf{q}_1^T Q_f \mathbf{q}_1)).$$

Since Q_o is a rank-1 symmetric positive semidefinite matrix, it can be expressed as $Q_o = \mathbf{N}\mathbf{N}^T$ for some vector $\mathbf{N}$. This implies that $(\mathbf{q}_0^T Q_o \mathbf{q}_1)^2 = (\mathbf{N}^T \mathbf{q}_0)^2 (\mathbf{N}^T \mathbf{q}_1)^2 = (\mathbf{q}_0^T Q_o \mathbf{q}_0)(\mathbf{q}_1^T Q_o \mathbf{q}_1)$. The coefficient in front of λ^2 in Eq. 14 vanishes, and the quadratic equation for λ simplifies to a linear equation. Solving for λ we find that the extremum of Λ_0 equals

$$\lambda^* = \frac{(q_f^{01})^2 - q_f^{00} q_f^{11}}{q_f^{00} q_0^{11} + q_o^{00} q_f^{11} - 2 q_f^{01} q_o^{01}},$$

where $q_f^{ij} = \mathbf{q}_i^T Q_f \mathbf{q}_j$ and $q_o^{ij} = \mathbf{q}_i^T Q_o \mathbf{q}_j$.

The next step is to establish conditions under which the root t^* belongs to $(0, 1)$. For a parabola $at^2 + bt + c$ with a single root, this root is attained at $t^* = -\frac{b}{2a}$. Thus, we require that $-\frac{b}{2a} \in (0, 1)$:

$$0 < -\frac{\mathbf{q}_0^T Q_{\lambda^*} (\mathbf{q}_1 - \mathbf{q}_0)}{(\mathbf{q}_1 - \mathbf{q}_0)^T Q_{\lambda^*} (\mathbf{q}_1 - \mathbf{q}_0)} < 1. \tag{15}$$

We first consider the case that $(\mathbf{q}_1 - \mathbf{q}_0)^T Q_{\lambda^*} (\mathbf{q}_1 - \mathbf{q}_0) > 0$ which is equivalent to saying that the parabola $g(\lambda^*, \mathbf{q}(t))$ opens upwards. Since there is only one root, it holds that $g(\lambda^*, \mathbf{q}(t)) > 0$ $\forall t \neq t^*$.

Our goal is to find conditions under which

$$0 < -\mathbf{q}_0^T Q_{\lambda^*} (\mathbf{q}_1 - \mathbf{q}_0) < (\mathbf{q}_1 - \mathbf{q}_0)^T Q_{\lambda^*} (\mathbf{q}_1 - \mathbf{q}_0).$$

For the first inequality, we get that

$$-\mathbf{q}_0^T Q_{\lambda^*} (\mathbf{q}_1 - \mathbf{q}_0) > 0 \quad \Leftrightarrow \quad \mathbf{q}_0^T Q_{\lambda^*} \mathbf{q}_0 > \mathbf{q}_0^T Q_{\lambda^*} \mathbf{q}_1$$

Considering that $\mathbf{q}_0^T Q_{\lambda^*}\mathbf{q}_0 = g(\lambda^*, \mathbf{q}(0))$ and that $t = 0 \neq t^*$, we state that $\mathbf{q}_0^T Q_{\lambda^*}\mathbf{q}_0 > 0$. Thus, the previous inequality holds iff

$$\mathbf{q}_0^T Q_{\lambda^*}\mathbf{q}_1 < 0 \text{ or } (\mathbf{q}_0^T Q_{\lambda^*}\mathbf{q}_0)^2 > (\mathbf{q}_0^T Q_{\lambda^*}\mathbf{q}_1)^2.$$

Using Eq. 13, we conclude that the first inequality holds iff

$$\mathbf{q}_0^T Q_{\lambda^*}\mathbf{q}_1 < 0 \text{ or } \mathbf{q}_0^T Q_{\lambda^*}\mathbf{q}_0 > \mathbf{q}_1^T Q_{\lambda^*}\mathbf{q}_1.$$

For the second inequality, we use similar reasoning:

$$-\mathbf{q}_0^T Q_{\lambda^*}(\mathbf{q}_1 - \mathbf{q}_0) < (\mathbf{q}_1 - \mathbf{q}_0)^T Q_{\lambda^*}(\mathbf{q}_1 - \mathbf{q}_0) \Leftrightarrow$$
$$\mathbf{q}_0^T Q_{\lambda^*}\mathbf{q}_1 < \mathbf{q}_1^T Q_{\lambda^*}\mathbf{q}_1 \Leftrightarrow$$

$$\mathbf{q}_0^T Q_{\lambda^*}\mathbf{q}_1 < 0 \text{ or } (\mathbf{q}_0^T Q_{\lambda^*}\mathbf{q}_1)^2 < (\mathbf{q}_1^T Q_{\lambda^*}\mathbf{q}_1)^2 \Leftrightarrow$$
$$\mathbf{q}_0^T Q_{\lambda^*}\mathbf{q}_1 < 0 \text{ or } \mathbf{q}_0^T Q_{\lambda^*}\mathbf{q}_0 < \mathbf{q}_1^T Q_{\lambda^*}\mathbf{q}_1$$

For the extremum to belong to $(0, 1)$ it is necessary that both inequalities hold, and therefore both conditions that we derived above have to be satisfied, i.e.

$$(\mathbf{q}_0^T Q_{\lambda^*}\mathbf{q}_1 < 0 \text{ or } \mathbf{q}_0^T Q_{\lambda^*}\mathbf{q}_0 > \mathbf{q}_1^T Q_{\lambda^*}\mathbf{q}_1) \text{ and}$$
$$(\mathbf{q}_0^T Q_{\lambda^*}\mathbf{q}_1 < 0 \text{ or } \mathbf{q}_0^T Q_{\lambda^*}\mathbf{q}_0 < \mathbf{q}_1^T Q_{\lambda^*}\mathbf{q}_1).$$

This condition is equivalent to $\mathbf{q}_0^T Q_{\lambda^*}\mathbf{q}_1 < 0$.

For the case when $(\mathbf{q}_1 - \mathbf{q}_0)^T Q_{\lambda^*}(\mathbf{q}_1 - \mathbf{q}_0) < 0$, similar derivations show that Ineq. 15 holds iff $\mathbf{q}_0^T Q_{\lambda^*}\mathbf{q}_1 > 0$.

Thus, we proved that λ^* is the extremum of the set Λ_0 iff

$$\left(\mathbf{q}_0^T Q_{\lambda^*}\mathbf{q}_1\right)\left((\mathbf{q}_1 - \mathbf{q}_0)^T Q_{\lambda^*}(\mathbf{q}_1 - \mathbf{q}_0)\right) < 0.$$

E DISTANCE FROM A QUADRIC TO A PLANE

In the following, we prove that distance $d(Q, \mathbf{q})$ from a (dual) quadric Q to a plane $\mathbf{q}$ can be approximated with

$$\tilde{d}^2(Q, \mathbf{q}) = \frac{9(\mathbf{q}^T Q\mathbf{q})^2}{4\mathbf{q}^T QAQ\mathbf{q}}, \text{ where } A = \text{diag}(0, 0, 0, 1).$$

We start from defining distance between two planes $\mathbf{q}_0$ and $\mathbf{q}_1$. The distance can be meaningfully defined only for parallel planes, i.e. only in the case when $\exists\lambda \in \mathbb{R}$ s.t. $\mathbf{q}_0 = \mathbf{q}_1 + \lambda\mathbf{i}$, where $\mathbf{i} = [0, 0, 0, 1]^T$. When $\mathbf{q}_0$ and $\mathbf{q}_1$ are parallel, the distance $d(\mathbf{q}_0, \mathbf{q}_1)$ is simply the offset between the planes. Equivalently, $d(\mathbf{q}_0, \mathbf{q}_1) = |\lambda|$, or $d(\mathbf{q}_0, \mathbf{q}_1)^2 = (\mathbf{q}_0 - \mathbf{q}_1)^T A(\mathbf{q}_0 - \mathbf{q}_1)$.

The distance between a quadric Q and a plane $\mathbf{q}$ is the distance from $\mathbf{q}$ to the closest parallel plane $\mathbf{q}'$ that lies on the quadric Q:

$$d(Q, \mathbf{q}) = \min_{\substack{\mathbf{q}'=\mathbf{q}+\lambda\mathbf{i} \\ \mathbf{q}'^T Q\mathbf{q}'=0}} d(\mathbf{q}', \mathbf{q}).$$

Solving for an analytical expression for $d(Q, \mathbf{q})$ is possible, but the resulting formula is too complex to work with. Instead, we follow the approach proposed by Taubin [1991], and approximate the zero set of implicit function $f(\mathbf{q}') = \mathbf{q}'^T Q\mathbf{q}'$ with zero set of $\tilde{f}$ – first-order approximation of f at $\mathbf{q}$.

Using Taylor series expansion, we get that $\tilde{f}(\mathbf{q}') = f(\mathbf{q}) + 2\mathbf{q}^T Q\mathbf{q}'$. The approximate distance from Q to $\mathbf{q}$ is defined as

$$\tilde{d}(Q, \mathbf{q}) = \min_{\substack{\mathbf{q}'=\mathbf{q}+\lambda\mathbf{i} \\ \tilde{f}(\mathbf{q}')=0}} d(\mathbf{q}', \mathbf{q}).$$

To get a closed-form expression for $\tilde{d}(Q, \mathbf{q})$, we find $\mathbf{q}'$ parallel to $\mathbf{q}$ s.t. $\tilde{f}(\mathbf{q}') = 0$. This is equivalent to finding λ^* for which $\tilde{f}(\mathbf{q} + \lambda^*\mathbf{i}) = 0$. Such λ^* is unique and equals to $\lambda^* = (3\mathbf{q}^* Q\mathbf{q})/(2\mathbf{q}^T Q\mathbf{i})$.

Finally, we derive

$$\tilde{d}^2(Q, \mathbf{q}) = d(\mathbf{q}, \mathbf{q} + \lambda^*\mathbf{i}) = (\lambda^*)^2 \mathbf{i}^T A\mathbf{i} =$$
$$= \frac{9(\mathbf{q}^T Q\mathbf{q})^2}{4\mathbf{q}^T Q\mathbf{i}(\mathbf{q}^T Q\mathbf{i})^T} = \frac{9(\mathbf{q}^T Q\mathbf{q})^2}{4\mathbf{q}^T QAQ\mathbf{q}}.$$

F V-SPHERE TEST BUG

The rejection test proposed by Li [2019] relies on Hough transforms. For a plane P with unit normal $\mathbf{n}$ and offset d, the Hough transform is a vector $H(P) = d\mathbf{n} \in \mathbb{R}^3$. For a point V, a **hollow** sphere with center $\frac{\mathbf{V}}{2}$ and radius $\frac{\|\mathbf{V}\|}{2}$ is called the *V-sphere*. Consider an edge e formed by planes P_0 and P_1. Assuming that e is not a silhouette when viewed from the origin O, the edge e is a silhouette for a point $\mathbf{V}$ if and only if the corresponding *V-sphere* intersects the line segment connecting $H(P_0)$ and $H(P_1)$ [Olson and Zhang 2006].

Li [2019] and Olson and Zhang [2006] use this property to detect mesh patches that do not contain silhouettes. For a set of edges $\mathcal{E}$ formed by the set of planes $\mathcal{P}$, the bounding box $\mathcal{B}$ encloses Hough transforms of these planes i.e. $H(P) \in \mathcal{B},\ \forall P \in \mathcal{P}$. If the V-sphere does not intersect the **solid** box $\mathcal{B}$, it is guaranteed that there are no silhouette edges in the set $\mathcal{E}$. In the original version of redner, however, the intersection is performed for a **solid** V-sphere and a **solid** box $\mathcal{B}$. Such rejection test rejects fewer nodes than possible. In Fig. 7, we compare the number of false accepts for the original version of redner and the version with fixed rejection test. We show that in some cases, using the correct rejection test can noticeably reduce the number of false accepts. We use edge sampling with the corrected rejection test for all of our comparisons.

G THE SMALLEST BOUNDING SPHERE

In this section, we prove that optimization problem (7) has a solution if and only if the smallest bounding sphere of the set $\mathcal{N}$ has radius $r_{\min} < 1$. Moreover, we show that in this case $\mathbf{d}$ can be computed as $\mathbf{c}_{\min}/\|\mathbf{c}_{\min}\|$, where $\mathbf{c}_{\min}$ is the center of the smallest bounding sphere. In the following, we extensively use the property that the set $\mathcal{N}$ lies on the unit sphere $\mathcal{S}^2$ centered at the origin.

To prove the first part of the claim, assume that problem (7) is feasible and has a solution $\mathbf{d}$ s.t. $\mathbf{d}^T\mathbf{d} = 1$ and $\mathbf{d}^T\mathbf{n} \geq \cos\alpha > 0$, $\forall\mathbf{n} \in \mathcal{N}$. We can then construct a sphere with center $\mathbf{d}\cos\alpha$ and radius $\sin\alpha < 1$ that is guaranteed to bound the set $\mathcal{N}$. Since the radius of this bounding sphere is less than 1, the radius of the smallest bounding sphere will be less than 1 as well. Now we prove the opposite. Assume that the smallest bounding sphere $\mathcal{S}^2_{\min}$ with radius less than 1 has its center at $\mathbf{c}$. This sphere contains the whole set $\mathcal{N}$. The intersection of the volume bounded by $\mathcal{S}^2_{\min}$ with $\mathcal{S}^2$ is a spherical cap centered around $\mathbf{c}$. Since radius of $\mathcal{S}^2_{\min}$ is strictly less than 1, the spherical cup is a strict subset of $\mathcal{S}^2$ and the circle in the intersection of $\mathcal{S}^2$ and $\mathcal{S}^2_{\min}$ has radius $r_{\text{cap}} < 1$. Thus, problem (7) has a feasible point $\mathbf{d} = \mathbf{c}/\|\mathbf{c}\|$ since $\mathbf{d}^T\mathbf{n} \geq \sqrt{1 - r_{\text{cap}}^2} > 0,\ \forall\mathbf{n} \in \mathcal{N}$. This concludes the first part of the proof.

To prove the second half of the claim, let the minimal bounding sphere for the set $\mathcal{N}$ have a radius r_{min} and $\mathbf{c}_{\text{min}}$ as its center. We prove that $\mathbf{d} = \mathbf{c}_{\text{min}}/\|\mathbf{c}_{\text{min}}\|$ is a solution of problem (7). Let $\cos\alpha = \min_{\mathbf{n}\in\mathcal{N}} \mathbf{d}^T\mathbf{n}$. Assume that there is another feasible vector $\mathbf{d}'$ s.t. $\cos\alpha' = \min_{\mathbf{n}\in\mathcal{N}} \mathbf{d}'^T\mathbf{n} > \cos\alpha$. We define $\mathbf{c}' = \|\mathbf{c}_{\text{min}}\|\mathbf{d}'$ and compute the distance from this point to all points $\mathbf{n}$ in the set $\mathcal{N}$:

$$\|\mathbf{c}' - \mathbf{n}\|^2 = \|\mathbf{d}' \times (\mathbf{c}' - \mathbf{n})\|^2 + (\mathbf{d}'^T(\mathbf{c}' - \mathbf{n}))^2 =$$
$$\|\mathbf{d}' \times \mathbf{n}\|^2 + (\|\mathbf{c}_{\text{min}}\| - \mathbf{d}'^T\mathbf{n})^2 = 1 - (\mathbf{d}'^T\mathbf{n})^2 + (\|\mathbf{c}_{\text{min}}\| - \mathbf{d}'^T\mathbf{n})^2 =$$
$$1 + \|\mathbf{c}_{\text{min}}\|^2 - 2\|\mathbf{c}_{\text{min}}\|\mathbf{d}'^T\mathbf{n} < 1 + \|\mathbf{c}_{\text{min}}\|^2 - 2\|\mathbf{c}_{\text{min}}\|\cos\alpha' <$$
$$1 + \|\mathbf{c}_{\text{min}}\|^2 - 2\|\mathbf{c}_{\text{min}}\|\cos\alpha = 1 + \|\mathbf{c}_{\text{min}}\|^2 - 2\|\mathbf{c}_{\text{min}}\| \min_{\mathbf{n}\in\mathcal{N}} \mathbf{d}^T\mathbf{n} =$$
$$\max_{\mathbf{n}\in\mathcal{N}}(1 + \|\mathbf{c}_{\text{min}}\|^2 - 2\|\mathbf{c}_{\text{min}}\|\mathbf{d}^T\mathbf{n}) = \max_{\mathbf{n}\in\mathcal{N}} \|\mathbf{c}_{\text{min}} - \mathbf{n}\|^2 = r_{\text{min}}^2$$

Thus, $\max_{\mathbf{n}\in\mathcal{N}} \|\mathbf{c}' - \mathbf{n}\|^2 < r_{\text{min}}^2$, implying that there exists a smaller bounding sphere. This contradiction concludes the proof.