DOI: 10.1111/cgf.70048
EUROGRAPHICS 2025 / A. Bousseau and A. Dai
(Guest Editors)

COMPUTER GRAPHICS *forum*
Volume 44 (2025), Number 2

Inverse Simulation of Radiative Thermal Transport

C. Freude[1], L. Lipp[1], M. Zezulka[1], F. Rist[2], M. Wimmer[1], and D. Hahn[1]

[1]TU Wien, Institute of Visual Computing & Human-Centered Technology, Austria
[2]King Abdullah University of Science and Technology (KAUST), Applied Mathematics and Computational Sciences, Saudi Arabia

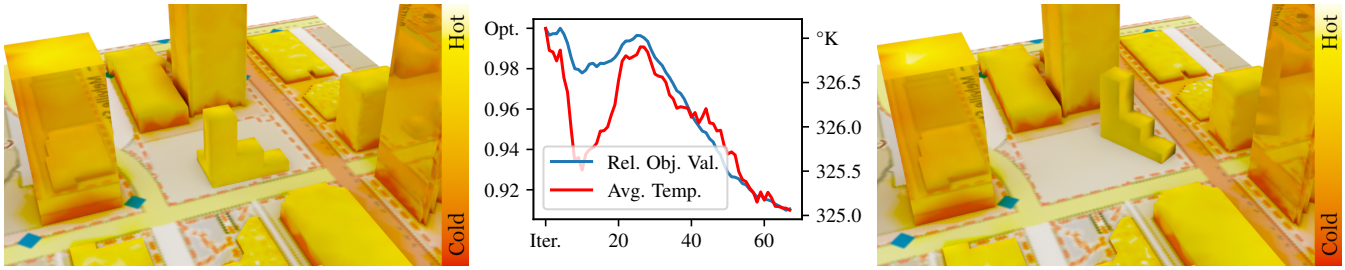


Figure 1: *Automatic optimization of the location and orientation of a building within a rectangular construction site in order to minimize its average surface temperature. The left image shows the initial plan, the plot shows the convergence of the relative objective-function value and the temperature, and the right image shows the optimized building layout. In addition to the visualization of the temperature distribution, we also display the city plan on the ground plane and indicate highly specular facades using a glossy material.*

Abstract

The early phase of urban planning and architectural design has a great impact on the thermal loads and characteristics of constructed buildings. It is, therefore, important to efficiently simulate thermal effects early on and rectify possible problems. In this paper, we present an inverse simulation of radiative heat transport and a differentiable photon-tracing approach. Our method utilizes GPU-accelerated ray tracing to speed up both the forward and adjoint simulation. Moreover, we incorporate matrix compression to further increase the efficiency of our thermal solver and support larger scenes. In addition to our differentiable photon-tracing approach, we introduce a novel approximate edge sampling scheme that re-uses primary samples instead of relying on explicit edge samples or auxiliary rays to resolve visibility discontinuities. Our inverse simulation system enables designers to not only predict the temperature distribution, but also automatically optimize the design to improve thermal comfort and avoid problematic configurations. We showcase our approach using several examples in which we optimize the placement of buildings or their facade geometry. Our approach can be used to optimize arbitrary geometric parameterizations and supports steady-state, as well as transient simulations.

CCS Concepts

• ***Computing methodologies*** → ***Ray tracing; Physical simulation;*** • ***Applied computing*** → *Physics; Computer-aided design;*

1. Introduction

Thermal simulations play a crucial role in urban planning and architectural design; however, they often do not receive the attention they deserve. As a result, in practice, issues can be overlooked that may be difficult to address later in the design process [DNW24]. State-of-the-art solutions allow for design evaluation but still require manual adjustments from the user to fix problems. In this paper, we introduce an inverse simulation method leveraging differentiable rendering. Our method allows for automated design optimization to reduce thermal loads and correct problematic configurations. We build upon the radiative thermal simulation by Freude et al. [FHR*23]: they efficiently simulate steady-state as well as transient radiative thermal transport, using photon tracing to precompute a transport matrix, which encodes radiative transport between all pairs of surface patches. Their approach enables designers to quickly and interactively evaluate the temperature distribution across the whole surface of a scene. However, once a problematic heat distribution has been identified, the design must be corrected *manually*, which is often non-trivial.

In this work, we present an approach to *automatically* optimize building-geometry parameters, introducing a novel differentiable heat transport simulation. We use gradient-based optimiza-

tion methods to minimize user-defined objective functions with respect to arbitrary geometric parameters, e.g., the location, orientation, or facade shape of a building. Furthermore, we also include heat transport by surface conduction in our simulation model. Figure 1 shows an example in which the placement of a building inside a city block is optimized to reduce its average surface temperature.

Building a differentiable simulation model for radiative heat transport poses two major challenges: first, as all materials emit radiation proportional to the fourth power of their temperature, the simulation model is inherently non-linear. While recent work has shown that automatically constructing a corresponding adjoint linear system is possible [BBC*22], currently popular automatic-differentiation tools do not include this functionality yet. We therefore derive the adjoint equation analytically. The second challenge is that radiative transport, including indirect radiation and potentially specular reflections, causes strong global coupling between temperatures throughout the scene. Simulating and differentiating this transport efficiently requires GPU-parallel Monte Carlo integration. Here, we present a differentiable photon-tracing approach, which allows us to compute the gradient of our minimization objective. We formulate this process such that we only need one additional photon-tracing pass regardless of the number of time steps taken during the simulation, so long as the parameterization of the scene geometry is time-independent.

Furthermore, we also show how the forward simulation can be accelerated by a low-rank compression of the transport matrix, in order to support scenes with a higher mesh resolution. Additionally, we implemented a *Blender* plugin for easy scene authoring, including optimization targets and free parameters, as well as solar irradiation from publicly available sky data sources. We verify the correctness of our approach on various test scenes and compare our gradients to finite-difference approximations. Additionally, we demonstrate the practical utility of our system on larger scenes, including concurrent optimization of multiple scene objects. In summary, our contributions are:

- a differentiable heat transport simulation method, supporting both steady-state and transient simulation, including a differentiable photon tracing step for the radiative transport component,
- a novel approximate edge sampling scheme for efficient discontinuity handling during differentiable photon tracing, and
- an evaluation of low-rank transport-matrix compression approaches for accelerating the transient forward simulation.

We first summarize related work in §2 and introduce the theory of heat transport and the finite-element discretization underlying our formulation in §3. We then describe our inverse thermal solver and our differentiable photon-tracing approach in §4, providing details and evaluating matrix-compression methods in §5. We present several examples in §6, including thermal optimization of object placements and mitigation of a problematic heat concentration by deforming a reflective facade.

2. Related Work

Heat transport can be classified into three distinct modes: conduction within the material, convection along a moving fluid, and radiation between surfaces. Heat conduction is one of the classical second-order partial differential equation (PDE) models and many simulation methods have been developed in this context over the years. Convection transports heat along with the motion of a fluid; while this transport mode is beyond the scope of this paper, approaches based on simplified flow models for urban design [LRPM22] could be incorporated in the future. The main challenge posed by *radiative* heat transport is the highly non-linear emissive term, i.e., any material emits radiation with a total power proportional to T^4, where T is the surface temperature (Stefan-Boltzmann law [Ste79; Bol84]).

The most widely used approach to simulating heat transfer, including radiation, is the finite element method (FEM), which discretizes the computational domain into small, often tetrahedral or triangular elements [LB13], and forms the basis of *radiosity* methods in computer graphics [Wie66]. Goral et al. [GTGB84] first applied this idea to compute diffuse-only light transport in complex scenes. Nevertheless, handling non-diffuse reflections in FEM is challenging. Conversely, Freude et al. [FHR*23] recently introduced a simulation method for radiative heat transport that utilizes a FEM discretization in combination with GPU-accelerated photon tracing, enabling both transient and steady-state simulation of surface temperatures including diffuse and specular reflections. Building upon their work, in this paper, we present the first differentiable heat simulation that includes radiative transport, especially the non-linear emissive term.

In general, Monte Carlo (MC) methods are a powerful tool for numerically approximating integrals via random sampling. This idea is the foundation for modern ray-tracing methods in physics-based rendering, i.e., solving the rendering equation [Kaj86] by tracing random rays that transport visible light through a 3D scene [Arv86; Jen96; PJH23]. In the last few years, differentiable rendering has received a lot of attention in the computer graphics community. Especially differentiable ray tracing has seen many applications, from shape or texture optimization to lighting design [LADL18; NVZJ19; ZMY*20; LHE*24]. One recurring problem in differentiable rendering is how to handle discontinuities caused by geometric occlusions (silhouette edges or hard shadows). This issue is typically addressed in rasterization by blurring the edges [LLCL19], whereas, for ray tracing, multiple solutions have been proposed. One option is to first detect and then separately sample these edges [LADL18], while other methods suggest to re-parameterize the discontinuous integrand near an edge [LHJ19; BLD20], directly start paths from them [ZRJ23], or apply Markov-Chain-Monte-Carlo perturbations on discontinuous edges [XBLZ24]. To reduce computational overhead compared to these methods, we build on the general edge-sampling idea, but make simplifying assumptions on the local irradiance field, which allows us to efficiently re-use primal samples and only trace one additional shadow ray for each primal sample that lies near a discontinuity edge. In the future, concurrent work on differentiable photon mapping [XLLX24] could also be considered for our application. Contrary to us, Liu and Hasegawa [LH23] address thermal inverse simulation with a grid-based differentiable immersed-boundary method.

Recently, exploiting the (generalized) mean value property of

harmonic functions [SC20], Monte Carlo (MC) methods have also been used to solve linear second-order PDEs. Earlier works, such as "Walk on Spheres" [Mul56; SC20] have focused on Dirichlet problems, while "Walk on Stars" and similar methods [SSJC23; MSCG24b] are capable of solving mixed boundary value problems; they have also been extended to exterior problems [SCJ*23]. Sawhney et al. [SSJC23] simulate diffusive-convective heat transfer between objects by solving Laplace's equation with mixed boundary conditions (Dirichlet and Neumann), while Miller et al. [MSCG24b] extend this approach to include Robin boundary conditions. Similarly, recent methods [BBC*23; PBC*23] couple a MC heat advection-diffusion model to a ray-tracing renderer to directly simulate and render thermal imaging. These MC-based PDE solvers have recently also been extended to differentiable simulation and inverse problems [YVJ24; MSCG24a; YWZZ24]. Applications include shape optimization [MSCG24a] and electrical impedance tomography [YVJ24]. Similarly, Mann et al. [MFS*22] developed an end-to-end differentiable model to optimize the energy absorption rate of photovoltaic installations. In all cases, external (temperature-independent) radiation sources may be included, as they fit within the underlying linear PDE model. However, currently, these MC methods cannot handle the full radiative coupling due to the non-linear emissive term, which we include in our differentiable simulation model.

3. Background

In this section, we briefly summarize relevant background and theory, starting with the well-known heat transport equation and describing our discrete simulation model. The main components of our discretization scheme are a diagonally lumped (thermal) mass matrix, surface conduction building on the cotangent formulation of the Laplace-Beltrami operator, and a radiative transport matrix following the formulation of Freude et al. [FHR*23].

The continuous volumetric heat transport equation assuming isotropic conduction and without convective transport is :

$$\rho c_p \frac{\partial T}{\partial t} - \kappa \nabla^2 T = 0 \quad \text{in } \Omega, \tag{1}$$

where T is the temperature, ρ is the mass density, c_p the heat capacity (at constant pressure), and κ the thermal conductivity [RMR*24]. We consider either standard Dirichlet boundary conditions, i.e., prescribing temperature data at the boundary $\Gamma_D \subset \partial\Omega$, i.e., $T = T_b$ on Γ_D, or radiative heat exchange

$$-\kappa \frac{\partial T}{\partial \mathbf{n}} = \varepsilon(\sigma T^4 - E) \quad \text{on } \Gamma_R, \tag{2}$$

where ε is the surface emissivity, E is the irradiance (i.e., incident flux per area over the entire wavelength spectrum), $\mathbf{n}$ is the outward unit surface normal, and σ is the Stefan-Boltzmann constant.

We derive the common finite-element discretization for this model as described in our appendix. Collecting per-vertex temperatures into the vector of unknowns $\mathbf{T}$, writing its time derivative as $\dot{\mathbf{T}}$, and denoting the component-wise fourth power as $\mathbf{T}^{\cdot 4}$, we can write the spatially discretized system as

$$\mathbf{M}\dot{\mathbf{T}} + \mathcal{L}\mathbf{T} + \mathcal{T}\,\mathbf{T}^{\cdot 4} = 0, \tag{3}$$

where $\mathbf{M}$ denotes the diagonally lumped (thermal) mass matrix, $\mathcal{L}$ is the Laplace-Beltrami operator discretizing the conduction term, and $\mathcal{T}$ is the radiative transport operator. We either solve for a steady state by setting $\dot{\mathbf{T}} = 0$, or apply backward Euler integration for time-dependent (transient) simulations. For each time step i (of size Δ_t) from $t_0 = 0$ to $t_{\max}$, we solve the following non-linear system for the next temperatures $\mathbf{T}_{i+1}$:

$$(\mathbf{M} + \Delta_t \mathcal{L})\mathbf{T}_{i+1} + \Delta_t \mathcal{T}\,\mathbf{T}_{i+1}^{\cdot 4} = \mathbf{M}\mathbf{T}_i, \tag{4}$$

respecting the (potentially time-dependent) Dirichlet boundary condition $\mathbf{T}_i|_{\Gamma_D} = \mathbf{T}_D(t_i)$, which also models external (solar) irradiation. The initial condition $\mathbf{T}_0$ could be specified by the user explicitly, or given as the solution to the steady-state problem

$$\mathcal{L}\mathbf{T}_0 + \mathcal{T}\,\mathbf{T}_0^{\cdot 4} = 0 \quad \text{s. t. } \mathbf{T}_0|_{\Gamma_D} = \mathbf{T}_D(t=0). \tag{5}$$

Finally, note that the irradiance E results from radiation being emitted by other surfaces and propagating according to the radiative transport equation. In practice, we construct the discrete radiative transport operator $\mathcal{T}$ in a parallel photon-tracing step on the GPU, using the publicly available implementation by Freude et al. [FHR*23]. In addition, in §5.2 we also discuss how to speed up the forward simulation by approximating the transport operator using *adaptive cross approximation* (ACA) [Beb00; Gra04]. We compute solar irradiation data using *Ladybug* [Lad24] in conjunction with *Radiance GENDAYMTX* [WS98] and freely available weather tapes from *EnergyPlus* [CPLW00], which provides data for many locations around the globe. In the future, more recent sky models developed for rendering [HW12; HW13; GGJ18; WVB*21] could also be used.

3.1. Problem Statement

In this paper, our goal is to solve optimization problems where the objective function measures the temperature distribution (in space and time) resulting from Eq. (3) (or the corresponding steady-state problem) and where the optimization parameters $\mathbf{p}$ define a static (i.e., time-independent) parameterization of the surface geometry. Formally, we aim to minimize an objective function Φ that measures the temperature at all time steps:

$$\begin{aligned} \mathbf{p}^* &= \arg\min_{\mathbf{p}} (\Phi + C(\mathbf{x}(\mathbf{p}))), \\ \Phi &= t_{\max}^{-1} \textstyle\sum_i \Delta_t \varphi(t_i, \mathbf{T}_i(\mathbf{p})), \end{aligned} \tag{6}$$

where all $\mathbf{T}_i(\mathbf{p})$ satisfy Eq. (4) for $i > 0$ and Eq. (5) for $i = 0$ (except for explicitly defined initial conditions). Here, φ denotes the part of the objective function that evaluates the resulting temperature at each time step t_i. Finally, C encapsulates all regularization and (soft) constraint terms that directly measure the geometric configuration, where $\mathbf{x}(\mathbf{p})$ refers to the list of all (parameterized) vertex positions. In order to find solutions to this problem efficiently, we use gradient-based optimization algorithms, which means we must compute the gradient of the objective function with respect to the optimization parameters, $\nabla_{\mathbf{p}}(\Phi + C)$. Note that for soft constraints and regularization terms, $\nabla_{\mathbf{p}} C$ is usually easy to compute because these terms do not include the simulated temperature data. Therefore, we focus on computing $\nabla_{\mathbf{p}}\Phi$ in the following section.

Our approach builds on the photon tracing method of Freude et al. [FHR*23] because of its guaranteed energy conservation in

the discrete setting and support for global illumination effects. We then formulate a differentiable counterpart, including an approximate edge-sampling scheme in order to minimize the computational overhead of differentiating discontinuous integrands. Our approach uses only one shadow ray, whereas alternative methods typically require multiple additional samples. Another advantage of the photon tracing framework is that it allows for an intuitive physical interpretation of our approximate edge sampling. In the context of this work, we consider design optimization for buildings that do not undergo time-dependent deformations, i.e., our parameterized geometry defines a static shape, rather than a motion. While we focus on optimizing the geometry of these buildings, in principle, material parameters, such as heat capacity or emissivity, could also be considered and would fit directly into our theoretical framework.

4. Differentiable Thermal Radiation

Having outlined the physical model of heat transport and our core optimization problem in the previous section, we now derive adjoint equations in order to compute the objective function gradient. In the following, we assume that the entire scene geometry is given as a triangle mesh, and each vertex position in that mesh is (or could be) an independent optimization parameter. In the following derivation, we consider the "full" parameterization $\mathbf{x} = \mathbf{p}$. In practice, restricting the gradients derived here to some arbitrary smooth parameterization $\mathbf{x}(\mathbf{p})$ can be easily done afterwards by projecting with the Jacobian of the chosen parameterization $(d\mathbf{x}/d\mathbf{p})$.

Following the derivation of Bradley [Bra24], the time-continuous adjoint of Eq. (3) is

$$\mathbf{M}^\mathsf{T}\dot{\lambda} = (\mathcal{L} + \operatorname{diag}(4\mathbf{T}^{\cdot 3})\mathcal{T})^\mathsf{T}\lambda + (\partial\varphi/\partial\mathbf{T})^\mathsf{T}, \tag{7}$$

where λ is the adjoint state. Applying the same backward Euler time discretization as in the forward simulation yields

$$\mathbf{S}_i^\mathsf{T}\lambda_i = \mathbf{M}^\mathsf{T}\lambda_{i+1} - \Delta_t(\partial\varphi/\partial\mathbf{T}_i)^\mathsf{T}, \tag{8}$$

where the system matrix $\mathbf{S}_i = \mathbf{M} + \Delta_t\mathcal{L} + \Delta_t \operatorname{diag}(4\mathbf{T}_i^{\cdot 3})\mathcal{T}$ corresponds to the Jacobian of Eq. (4). The boundary conditions are always $\lambda = 0$ on Γ_D, and the adjoint "initial" condition is $\lambda(t = t_{\max}) = 0$. Starting from the adjoint initial condition, we first solve Eq. (8) for $t_i = t_{\max} - \Delta_t$ and then proceed backward in time until all λ_i have been computed. The gradient of the objective function is then

$$\begin{aligned}\nabla_\mathbf{p}\Phi = {} & t_{\max}^{-1}\,\mu^\mathsf{T}(\partial_\mathbf{p}\mathcal{L}\mathbf{T}_0 + \partial_\mathbf{p}\mathcal{T}\mathbf{T}_0^{\cdot 4}) \\ & + t_{\max}^{-1}\sum\nolimits_i \Delta_t\left(\lambda_i^\mathsf{T}(\partial_\mathbf{p}\mathbf{M}\dot{\mathbf{T}}_i + \partial_\mathbf{p}\mathcal{L}\mathbf{T}_i + \partial_\mathbf{p}\mathcal{T}\mathbf{T}_i^{\cdot 4})\right),\end{aligned} \tag{9}$$

where $\mathbf{S}_0^\mathsf{T}\mu = \mathbf{M}^\mathsf{T}\lambda_0$ and $\mathbf{S}_0 = (\mathcal{L} + \operatorname{diag}(4\mathbf{T}_0^{\cdot 3})\mathcal{T})$. Here, we use the shorthand notation $\partial_\mathbf{p}(\cdot) = \partial(\cdot)/\partial\mathbf{p}$. Note that in the time-discrete setting, $\dot{\mathbf{T}}_i = (\mathbf{T}_{i+1} - \mathbf{T}_i)/\Delta_t$.

In case we only solve for steady-state temperatures $\mathbf{T}$, and therefore $\Phi = \varphi(\mathbf{T})$, these equations simplify to

$$\mathbf{S}^\mathsf{T}\lambda = -(\partial\varphi/\partial\mathbf{T})^\mathsf{T}, \text{ and} \tag{10}$$

$$\nabla_\mathbf{p}\Phi = \lambda^\mathsf{T}(\partial_\mathbf{p}\mathcal{L}\mathbf{T} + \partial_\mathbf{p}\mathcal{T}\mathbf{T}^{\cdot 4}). \tag{11}$$

Once we have completed the forward simulation (solving Eq. (4) or (5) respectively) as well as the adjoint simulation (solving Eq. (8) or (10) respectively), the remaining step to complete the gradient calculation according to Eq. (9) or (11) is to evaluate the products of the form $\mathbf{v}^\mathsf{T}(\partial\mathbf{A}/\partial\mathbf{p})\mathbf{u}$. Because $\mathbf{M}$ and $\mathcal{L}$ are sparse matrices resulting from a standard finite-element discretization, products involving these terms can be computed using the usual FEM assembly implementation and computing the required derivatives with respect to parameters locally, per element. Note that in our results, we omit these terms, as they typically have a much weaker influence on the resulting gradients than the changes to the radiative transport configuration. The main reason for this behavior is that we typically consider either rigid motions or (almost) isometric deformations that have no (or negligible) influence on the thermal mass or heat conduction.

Consequently, we are left with the most challenging terms to compute, which involve derivatives of the thermal transport operator $\partial\mathcal{T}/\partial\mathbf{p}$. Note that $\mathcal{T}$ is a dense matrix of size $n \times n$ (where n is the number of vertices in the scene). Furthermore, each vertex could, in principle, move independently during optimization, resulting in up to $3n$ parameters. Therefore $\partial_\mathbf{p}\mathcal{T}$ is a dense tensor of derivatives of size (up to) $n \times n \times 3n$; hence, we want to avoid storing this tensor in (GPU) memory.

For the forward simulation, we construct $\mathcal{T}$ via GPU-accelerated photon tracing as in Freude et al. [FHR*23], where each entry $\mathcal{T}_{kj}$ results from a summation over many photon paths (connecting the mesh vertices j and k). In our differentiable photon-tracing step, we consequently compute products of the form $\mathbf{v}^\mathsf{T}\partial_\mathbf{p}\mathcal{T}\mathbf{u}$ alongside the required derivatives also on a per-path basis, as we describe in the remainder of this section. We then directly accumulate the results of the per-path products into the final gradient.

4.1. Differentiable Photon Tracing

The transport operator $\mathcal{T}$ follows from discretizing the radiative transport term in Eq. (20). It consists of an emissive part $\sigma\varepsilon T^4$ (on the diagonal) and an absorptive part εE, which is computed by photon tracing: We emit N photons around each vertex j with power $p_l = \sigma\varepsilon_j A_j/N$, where A_j is the vertex-associated area. Note that due the sign convention in Eq. (20) and (3), we then add p_l to the diagonal $\mathcal{T}_{jj}$. We trace the photon through the scene, including indirect scattering, until it is absorbed near some other vertex k, where we subsequently subtract p_l from $\mathcal{T}_{kj}$. Please refer to Freude et al. [FHR*23] for details on this construction, which they derive from a path-integral formulation. In summary, the discrete form of this path integral results in

$$\mathcal{T}_{kj} = \delta_{jk}\sigma\varepsilon_j A_j - \sum\nolimits_l p_l[j \to k], \tag{12}$$

where the Kronecker delta $\delta_{jk} = 1$ if $j = k$, 0 otherwise; and $p_l[j \to k]$ refers to any photon that was emitted near vertex j and absorbed near vertex k (including via indirect scattering paths).

While this photon-tracing approach might not appear amenable to differentiation on first glance, we use the key ideas of path-space differentiable rendering [ZMY*20] to find the required derivatives. Specifically, we first compute the continuous part, as suggested in §6.1 of Zhang et al. [ZMY*20], by always differentiating the geometric term (which is implied but not explicitly evaluated during photon tracing) without differentiating any sampling probabilities.

Algorithm 1 Differentiable thermal simulation: blue text indicates code that runs on the GPU.

Input: objective function φ, scene data (mesh and materials).
Input: initial and boundary conditions. // $\mathbf{T}_0$ can be given explicitly or computed as a steady-state solution
Initialize $\mathcal{T} = 0$ // Begin photon tracing.
for each mesh vertex j **do**
  trace N photons (in cos-weighted random directions),
  at each photon-surface intersection (near vertex k):
  reflect or absorb randomly proportional to emissivity ε_k.
  if photon absorbed near vertex k **then**
    accumulate absorption in transport operator:
    $\mathcal{T}_{kj}$ $-= \sigma A_j \varepsilon_j / N$. // Note the sign convention in Eq. (20): positive for emission, negative for absorption.
  Emissive diagonal term:
  $\mathcal{T}_{jj}$ $+= \sigma A_j \varepsilon_j$.
Assemble $\mathcal{L}$ // conductivity-scaled cotan-Laplacian
and $\mathbf{M}$ // diagonally-lumped thermal mass ($\mathbf{M}_{jj} = A_j\, \rho_j\, c_{pj}\, h_j$).
if $\mathbf{T}_0$ not explicitly given **then**
  solve Eq. (5) for $\mathbf{T}_0$,
  $\Phi = t_{\max}^{-1} \Delta_t \varphi(0, \mathbf{T}_0)$.
else
  $\Phi = 0$.
// Transient forward simulation – skip for steady-state only:
Initialize $\mathbf{T}_i = \mathbf{T}_0$.
for time $t_i = 0$ **to** $t_{\max}$ (time step Δ_t) **do**
  solve Eq. (4) for $\mathbf{T}_{i+1}$.
  Evaluate objective function: Φ $+= t_{\max}^{-1} \Delta_t \varphi(t_{i+1}, \mathbf{T}_{i+1})$.
// Adjoint simulation:
Initialize $\lambda = 0$.
// Note for steady-state only: solve Eq. (10), skip for-loop.
for time $t_i = t_{\max} - \Delta_t$ **to** 0 (time step $-\Delta_t$) **do**
  compute objective derivative: $\partial\varphi / \partial \mathbf{T}_i$.
  Solve Eq. (8) for λ_i.
if $\mathbf{T}_0$ not explicitly given **then**
  solve $\mathbf{S}_0^\mathsf{T} \mu = \mathbf{M}^\mathsf{T} \lambda_0$ for μ.
Initialize $\nabla_\mathbf{p} \Phi = 0$ // Begin differentiable photon tracing.
for each mesh vertex j **do**
  re-trace all photon paths
  for each path segment **do**
    compute $\widetilde{d_\mathbf{p} \mathcal{T}_{kj}}$, i.e, the one term of the transport derivative corresponding to this segment, Eq. (14) and (15), where $\mathbf{p}$ refers to the six vertices of the two triangles containing the segment's start and end points.
    If applicable, add edge derivatives, §4.2, to $\widetilde{d_\mathbf{p} \mathcal{T}_{kj}}$, where $\mathbf{p}$ also includes the shadowed vertex.
    // Accumulate the objective function gradient, Eq. 16:
    for time $t_i = 0$ **to** $t_{\max}$ **do**
      $\nabla_\mathbf{p} \Phi$ $+= (\lambda_i)_k \widetilde{d_\mathbf{p} \mathcal{T}_{kj}} \, (\mathbf{T}_i)_j^4$
Output: temperature data for all time steps $\{\mathbf{T}_i\}$, objective function value Φ and gradient $\nabla_\mathbf{p} \Phi$.

Consider a photon p_l emitted near vertex j and absorbed near vertex k, travelling along a path consisting of n_l segments (so $n_l = 1$ refers to direct transport from j to k). Following Pharr et al. [PJH23], but accounting for the absorption at k, resulting in one fewer BRDF term than path segments, the throughput along this path is

$$\eta_l = \prod_{i=1}^{n_l - 1} f_i \prod_{i=1}^{n_l} g_i, \tag{13}$$

where f_i is the material's bidirectional reflectance distribution function (BRDF) at intermediate path vertices, and $g_i = |\cos\theta \cos\theta'| / d_i^2$ is the geometric term per path segment of length d_i (assuming visibility). Here θ and θ' are the exitant and incident angle relative to the local surface normal at the two endpoints of each segment. When constructing the transport operator during photon tracing, we sample paths such that dividing by the product of all sampling probabilities cancels out this throughput term, resulting in Eq. (12). This means that $\eta_l / (\prod P_i) = 1$ for sampling probabilities P_i. Therefore, we can consider photon tracing, Eq. (12), as a specific discretization of the path integral and follow Zhang et al. [ZMY*20] in differentiating the throughput but not the sampling probabilities.

We now consider the general case, where the parameter vector $\mathbf{p}$ contains per-vertex displacements for all mesh vertices. (Projecting the resulting per-vertex gradient to other differentiable parameterizations of mesh deformation later is straightforward). The derivative of Eq. (12) with respect to $\mathbf{p}$ is then

$$\frac{d\mathcal{T}_{kj}}{d\mathbf{p}} = \delta_{kj} \sigma \varepsilon_j \frac{dA_j}{d\mathbf{p}} - \sum_l \left(\frac{d\eta_l}{d\mathbf{p}} + \frac{dA_j}{d\mathbf{p}} \frac{1}{A_j} \right) p_l[j \to k], \tag{14}$$

where the derivative of the vertex-associated area follows directly from the triangles adjacent to the vertex. Note that A_j always refers to the area around the emitting vertex, so $dA_j/d\mathbf{p}$ is only non-zero at the vertices of the triangle containing the photon path's starting point. Finally, the derivative of the throughput follows from the product rule:

$$\frac{d\eta_l}{d\mathbf{p}} = \sum_{i=1}^{n_l - 1} \frac{\eta_l}{f_i} \frac{df_i}{d\mathbf{p}} + \sum_{i=1}^{n_l} \frac{\eta_l}{g_i} \frac{dg_i}{d\mathbf{p}}. \tag{15}$$

Following the "material-form differential path integral" idea [ZMY*20], we always keep the barycentric coordinates of each path vertex constant as the vertices of the containing triangles are displaced by $\mathbf{p}$ to compute the derivatives $df/d\mathbf{p}$ and $dg/d\mathbf{p}$. Note that for each segment of the path, we only need to compute those derivatives relating to the local BRDF and geometric term, which are non-zero only at the vertices of the triangles containing the segment endpoints. In this way, we compute one term of the sum over photon paths and sum over path segments at a time, resulting in one summand $\widetilde{d_\mathbf{p} \mathcal{T}_{kj}}$ of the derivative of the transport matrix entry kj with respect to those parameters that affect only the currently processed segment. We then immediately evaluate the products $\mathbf{v}^\mathsf{T} \widetilde{d_\mathbf{p} \mathcal{T}_{kj}} \mathbf{u}$, where $\mathbf{v}$ represents all adjoint states, $\mathbf{u}$ all temperatures (coefficient-wise raised to the fourth power), and accumulate the result into the objective gradient vector

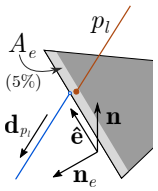
$$\nabla_\mathbf{p} \Phi \leftarrow \nabla_\mathbf{p} \Phi + \mathbf{v}^\mathsf{T} \widetilde{d_\mathbf{p} \mathcal{T}_{kj}} \mathbf{u}. \tag{16}$$

In summary, we accumulate these updates into $\nabla_\mathbf{p} \Phi$ independently for each path segment, and for each pair of adjoint state and temperature (i.e., for each time step and/or the (initial) steady-state).

4.2. Approximate Edge Sampling

For the discontinuous part, where Zhang et al. [ZMY*20] suggest a separate boundary integration step, we employ an approximate edge sampling that re-uses photons sampled in the primal pass, which we explain in the following. In this way, we avoid the computational overhead incurred by other methods that rely on separate edge samples of auxiliary rays.

The main approximation we make in our approach is to assume that in a small neighborhood of a discontinuous edge, the local irradiance remains constant across the edge. In a nutshell, we assume that the local derivative of the integral computing the absorbed radiative power, when offsetting the edge in a direction orthogonal to the edge by a small distance τ is of the form $(d/d\tau)\int_0^\tau c\,dx = c$ for some constant c representing the local irradiance. Consequently, we first detect photons that intersect a triangle close to one of this triangle's bounding edges based on the barycentric coordinates of the intersection point (in our results we always use a 5% margin, i.e., the minimal barycentric coordinate must be ≤ 0.05). For each such near-edge photon, we check if the edge casts a shadow onto any other triangle (if so we will refer to it as the occluding edge). For this check, we trace an additional shadow ray starting at the closest point on the edge (plus a tiny offset to avoid intersecting the same triangle again) in the same direction as the incident photon.



If we find a shadowed triangle, there is a discontinuous change in the power density from the occluding edge to the shadowed triangle, proportional to the local irradiance. We estimate this irradiance as $E_l = p_l/A_e$, where p_l is the photon's power and A_e is the area of the trapezoidal strip of the triangle as illustrated in the inset image, i.e., $A_e = rh_e(1-r/2)l_e$, with $r = 5\%$, h_e the height of the triangle orthogonal to the occluding edge, and l_e the length of that edge. We now need to find the direction $\mathbf{d}$ in which the incident photon path needs to shift in order to move across the occluding edge. Clearly, this direction must be orthogonal to the direction the photon is traveling in ($\mathbf{d}_{p_l}$) and also orthogonal to the occluding edge ($\hat{\mathbf{e}}$, where both $\mathbf{d}_{p_l}$ and $\hat{\mathbf{e}}$ are unit vectors). Consequently, this direction is $\mathbf{d} = \hat{\mathbf{e}} \times \mathbf{d}_{p_l} / \|\hat{\mathbf{e}} \times \mathbf{d}_{p_l}\|$. Finally, the differential irradiance transfer from the occluding to the shadowed object is then $dE_l = \mathbf{d}\,|\mathbf{d}\cdot\mathbf{n}_e|\,l_e\,E_l$, where the projection term $|\mathbf{d}\cdot\mathbf{n}_e|$ diminishes the effect if the incident photon arrives at a shallow angle ($\mathbf{n}_e$ is the outward unit normal vector to the edge in the plane of the occluding triangle). This differential irradiance is an additional derivative contribution to $\widetilde{d_\mathbf{p}\mathcal{T}_{kj}}$, where j refers to the vertex where the photon was emitted and k refers to the shadowed vertex (i.e., the vertex closest to the intersection point of the shadow ray on the shadowed triangle). The parameters $\mathbf{p}$ where this contribution is non-zero are the displacements of the endpoints of the occluding edge, as well as of the vertex where the photon path segment started (origin vertex) and the shadowed vertex.

Finally, we distribute the differential irradiance to the parameters moving either the occluding edge, the origin of the photon's current path segment, or the shadowed object. Due to energy conservation, these contributions must sum to zero. Consequently, we distribute the differential irradiance among the origin and shadowed vertex based on their relative distance to the occluding edge: Let d_{oe} denote the distance from the photon's origin (start of the path segment) to the occluding edge and d_{es} the distance from the occluding edge to the shadowed object, as illustrated in the inset image. For the occluding edge, we have $\widetilde{d_\mathbf{p}\mathcal{T}_{kj}} = -dE_l$, and consequently for the origin: $\widetilde{d_\mathbf{p}\mathcal{T}_{kj}} = dE_l\,d_{es}/(d_{oe}+d_{es})$, and $\widetilde{d_\mathbf{p}\mathcal{T}_{kj}} = dE_l\,d_{oe}/(d_{oe}+d_{es})$ for the shadowed vertex. With these edge derivative terms, we again directly update the objective gradient as in Eq. (16). Figure 2 compares different configurations of our edge sampling scheme on gradient descent optimization. Reducing the edge-detection margin from the default $r = 5\%$ (dark blue) to $r = 0.1\%$ (medium blue) delivers similar results when the overall number of traced photons is sufficiently high. Please also refer to Fig. 12 for a similar comparison with a lower number of photons. Reducing the margin even further ($r = 10^{-6}$, light blue) introduces too much gradient noise and leads to unsatisfactory convergence behaviour, while disabling edge sampling altogether fails to converge (red). Per-vertex and parameter-space gradients are shown in Fig. 13.

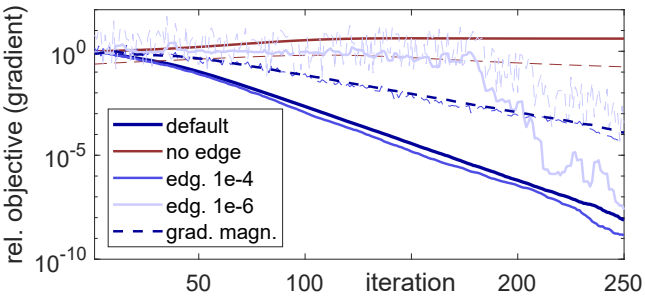


Figure 2: *Ground-truth recovery test as in Fig. 5: ablation study on gradient descent using different edge-detection margins in our edge sampling scheme. Default (5%, dark blue), 0.1% (medium blue), 10^{-6} (light blue), without edge sampling (red). Dashed lines show corresponding normalized gradient magnitudes.*

In summary, we re-use regular photons that intersect triangles near an edge in order to avoid detecting and sampling edges that cause shadows explicitly. In this way, we only need to trace one additional ray to find the occluded object and compute the differential power transfer based on the photon's original power.

5. Implementation Details

So far, we have considered the parameters $\mathbf{p}$ to refer directly to vertex positions in the mesh. In order to control the free parameters for the optimization on a per-scene basis, we implement various parameterization functions for the scene geometry, such that vertex positions $\mathbf{x}$ are specified as $\mathbf{x}(\mathbf{p})$, and we project the objective gradient by the Jacobian of the parameterization: $\nabla_\mathbf{p}\Phi = (d\mathbf{x}/d\mathbf{p})^\mathsf{T}\nabla_\mathbf{x}\Phi$. We implement this parameterization and gradient projection on the CPU after receiving $\nabla_\mathbf{x}\Phi$ from the differentiable photon tracing shader. In this way, the parameterization can be easily exchanged for different scenes without affecting the GPU implementation. For most results, we parameterize the geometry such that individual objects can translate relative to the ground plane

and rotate around the vertical axis. Additionally, we implement soft constraints on the vertex positions $C(\mathbf{x})$. In particular, we confine objects within a pre-defined polygon on the ground plane. We also prevent collisions, using object's circumcircles in the ground plane as collision proxies.

We have implemented a plugin for *Blender* [Ble24] that allows us to specify optimization settings and constraints as well as material parameters and the sky dome representing the main heat source. To transfer the scene between Blender and our framework we use the *GLTF* format. After optimization in our framework, we export the scene back into Blender again, including per-vertex temperatures, for visualization. We rendered most of our results using the *Cycles* rendering system. For the visualization of the per-vertex surface temperature distribution, we applied linear interpolation and a custom colormap.

In the remainder of this section, we discuss derivatives for ideally specular materials and options to speed up the forward simulation using matrix compression.

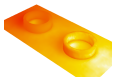
5.1. BRDF Derivatives

We use a mixture of ideally diffuse and ideally specular materials; in the former case, the bidirectional reflectance distribution function (BRDF) is constant and its derivatives vanish. In the latter case (ideally specular reflection) the BRDF is technically a Dirac distribution, which in practice means that we only sample the ideally mirrored outgoing direction during path construction. Xing et al. [XHL*23] suggest handling this constraint during differentiation by allowing the path vertex at the specular reflection to shift in the plane orthogonal to the surface such that the constraint always remains satisfied and include an additional term in the derivative of the path throughput accounting for this shift, instead of differentiating the BRDF itself. We follow this approach, however, in our experiments (Fig. 11) we found that the additional term has relatively small influence compared to the derivative of the throughput on the final results, while introducing more numerical noise to the gradients. Consequently, we do not include this constraint-derivative term in most of our results, although it could be added if required for specific applications in the future. Nevertheless, we are able to consistently reduce heat islands due to specular reflections in our optimization, as shown in Fig. 8 and 16.

5.2. Transport Matrix Compression

The adjoint problem and gradient calculation are linear, which allows us to implement the summation in Eq. (16) on a per-ray basis in a matrix-free way. The forward problem, Eq. (4) however, contains a strong non-linearity in the radiative component, $\mathcal{T}\,\mathbf{T}^{\cdot 4}$, where storing and re-using the transport operator $\mathcal{T}$ is preferable over repeated photon tracing during the non-linear solver. Nevertheless, this operator is (potentially) dense and grows quadratically with the number of vertices (degrees of freedom) in the mesh. In order to speed up the forward solver, we have investigated approaches for compressing the transport matrix via a low-rank approximation. In particular, we implement the *adaptive cross approximation* (ACA) algorithm, initially presented by Bebendorf [Beb00], as well as the improved ACA+ algorithm [Gra04]. The general

Table 1: *Overview of transport operator compression results. Columns: scene name, number of vertices n, total runtime for uncompressed reference solution t_u, total runtime with compression t_c, relative temperature error 'err.', rank of compression r, type of compression algorithm ('+' indicates ACA+, otherwise standard ACA, selected by which produced the lower error).*

Scene	n	t_u [s]	t_c [s]	err.	r	(+)
FHR-5	1067	2.25	2.09	0.7%	90	+
(Fig. 5 in	3769	25.40	16.20	1.2%	200	+
[FHR*23])	5089	47.17	35.16	1.6%	250	+
	10483	204.49	175.89	4.3%	220	+
Rings	777	2.13	1.96	4.3%	25	
	3687	39.77	9.01	5.4%	90	+
	9927	382.92	99.23	4.4%	180	+
	14535	992.33	99.23	5.8%	200	
Blocker	214	0.40	1.00	1.4%	20	
(Fig. 5)	2168	50.66	30.56	1.2%	80	+
	5180	306.55	148.57	2.2%	150	+
	6772	552.62	335.65	2.5%	300	+
Reflector	307	0.39	0.47	0.2%	25	+
	707	1.81	1.51	0.2%	60	+
	2035	18.43	10.63	0.2%	200	+
	6803	312.12	170.64	0.3%	650	+
	8403	597.45	257.68	0.8%	550	

idea of ACA is to approximate a $m \times n$ matrix $\mathbf{A} \approx \tilde{\mathbf{A}}$ by a sum of outer products of columns and rows selected from the input matrix, i.e., $\tilde{\mathbf{A}} = \mathbf{UV} = \sum_{k=1}^{r} \mathbf{u}_k \mathbf{v}_k$, where r is the rank of the approximation, and $\mathbf{u}_k$ and $\mathbf{v}_k$ are the selected columns and rows of $\mathbf{A}$ respectively (the latter must be divided by the pivot element where the selected row intersects the corresponding selected column). Note that in our case $n = m$ where n is the number of vertices in the mesh. The goal of the ACA algorithm is, therefore, to choose an appropriate subset of r columns and rows from the input matrix, where a common expectation is that $r \approx \log(n)$ [Gra04].

To make this selection, we consider the residual $\mathbf{R}$ and iteratively expand the selection $\mathbf{U}, \mathbf{V}$ until error threshold ϵ is reached:

$$\|\mathbf{R}\| = \left\|\mathbf{A} - \tilde{\mathbf{A}}\right\| \leq \epsilon, \tag{17}$$

where $\|\cdot\|$ is the Frobenius norm. A key feature of ACA is that this residual matrix is never fully computed, but updated one row or column at a time, while iteratively adding one column or row to $\mathbf{U}$ or $\mathbf{V}$ respectively, depending on where the largest error in the known part of $\mathbf{R}$ currently occurs. Similarly, the input matrix $\mathbf{A}$ does not need to be fully computed, so long as the selected rows or columns can be retrieved as needed. The ACA+ variant (see Grasedyck [Gra04] for details) differs from standard ACA primarily in the formulation of the row and column selection criteria.

In our case, the system matrix in each non-linear solver iteration is $\mathbf{S}_i = \mathbf{M} + \Delta_t \mathcal{L} + \Delta_t \operatorname{diag}(4\mathbf{T}_i^3)\mathcal{T}$, and the inner linear solver requires matrix-vector product evaluations for this matrix. Note that $\mathbf{M}$ is a diagonal and $\mathcal{L}$ is a sparse matrix, so we only need to compress the (potentially dense) transport operator $\mathcal{T}$. This operator is

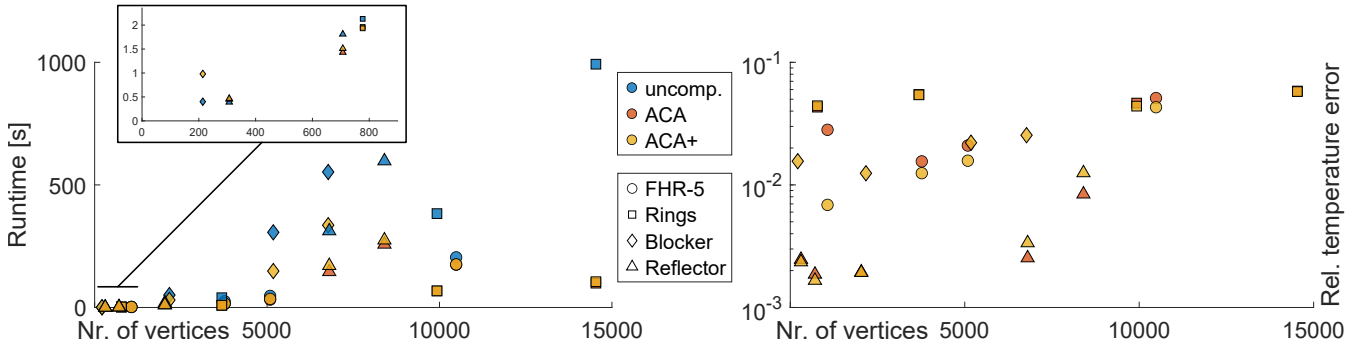


Figure 3: *Comparison of matrix compression methods: adaptive cross approximation (ACA) and ACA+ [Gra04] (marker color) on four different test scenes (marker shapes) and varying mesh resolution. Left: total runtime for uncompressed vs. compressed variants for the forward simulation, including matrix assembly, compression, and non-linear system solving. Right: relative errors in the final temperature distribution compared to the uncompressed reference solution. All errors are on the order of a few percent or less, while the maximal error is 5.8%. Please also refer to Table 1 for an overview of these results.*

composed of emissive and absorptive parts, describing how radiative power emitted near any one vertex is transported to any other vertex. We can therefore decompose $\mathcal{T}$ into these two components, $\mathcal{T} = \mathcal{T}_{\text{em}} + \mathcal{T}_{\text{ab}}$, where the emissive part is a diagonal matrix. As (non-zero) diagonal matrices have full rank, the diagonal dominance resulting from $\mathcal{T}_{\text{em}}$ would cause significant error for a low-rank approximation. We therefore apply ACA(+) to compress only the absorptive part of the transport operator: $\mathcal{T}_{\text{ab}} \approx \mathbf{U}_{\text{ab}}\mathbf{V}_{\text{ab}}$. Using the shorthand notation $\mathbf{D}_i = \text{diag}(4\mathbf{T}_i^3)$ and the decomposition of the transport operator, we can write the linear system matrix as $\mathbf{S}_i = \mathbf{M} + \Delta_t\mathcal{L} + \Delta_t\mathbf{D}_i(\mathcal{T}_{\text{em}} + \mathcal{T}_{\text{ab}})$. Approximating the absorptive part by ACA(+), the required matrix-vector products $\mathbf{S}_i\,\mathbf{x}$ for an arbitrary vector $\mathbf{x}$ can then be computed as

$$\mathbf{S}_i\,\mathbf{x} \approx (\mathbf{M} + \Delta_t\mathcal{L} + \Delta_t\mathbf{D}_i\mathcal{T}_{\text{em}})\,\mathbf{x} + \Delta_t\mathbf{D}_i\mathbf{U}_{\text{ab}}\mathbf{V}_{\text{ab}}\,\mathbf{x}, \tag{18}$$

where the first term on the right-hand side is sparse and the second term contains the "thin" matrices $\mathbf{U}_{\text{ab}}$ and $\mathbf{V}_{\text{ab}}$ (size $n \times r$ and $r \times n$ respectively), which can be computed efficiently as a sequence of matrix-vector products from right to left. Apart from the reduced memory cost for the transport matrix, this formulation significantly increases performance for matrix-vector products (assuming $r \ll n$). In our implementation, we use a wrapper class to represent this compressed system matrix and implement its matrix-vector product operation, building on the open linear algebra library Eigen, in particular their *EigenBase*<...> class. In this way, we can immediately use Eigen's iterative linear solvers [GJ*10] with the ACA-compressed format.

In Fig. 3, we compare the total runtimes and errors for various test scenes, and we also summarize these results in Table 1. Depending on the complexity of the scene and the mesh resolution, the transient forward simulation can achieve up to 10× speed up, with a relative temperature error of a few percent compared to the uncompressed reference solution. While we currently only implement ACA(+) on the CPU side, in the future, it may be possible to integrate ACA(+) into the ray-tracing shaders so as to evaluate only the columns and rows requested by the compression algorithm.

6. Results

In this section, we first present some geometrically simple test cases to verify the correctness of our approach before demonstrating the applicability of our method to various design tasks. For basic test cases, we emit 256k photons per triangle by default; for larger scenes the triangle and total photon numbers are listed in Tbl. 2.

6.1. Validation Tests

We first compare the optimization convergence of a basic gradient-descent method using our gradients to a finite-difference (FD) approximation in Fig. 4. Here, the optimization parameters allow the temperature sensor to move and rotate freely until a constant target temperature is achieved at the sensor. Initially, our gradients and the FD approximation converge equally well. Later on, our gradients allow the optimizer to converge more precisely to the target temperature, whereas the approximation error in finite differencing prevents convergence. Similarly, in the appendix, Fig. 10, we compare gradients to FD approximations at varying FD-step sizes: the error between our gradient and the FD approximation for the optimal step size is below 1%. Instead of optimizing towards a uniform constant temperature (which cannot be achieved exactly), in Fig. 5 we aim to recover a ground-truth temperature distribution. We compare gradient descent and Adam optimization on this test case, where the blocking object (3/4-disk) must be translated and rotated (in-plane horizontally) to return to its original place. In Fig. 2 we also show that this test case fails without considering discontinuous edges in the gradient calculation, but converges accurately when using our edge sampling scheme. In Fig. 6, we repeat the ground-truth recovery test for a time-dependent simulation with the number of photons for both forward and adjoint simulation reduced by a factor of 80. The increase in noise due to the smaller number of sampled photons causes the FD-based optimization to fail, while our gradients robustly lead the gradient-descent optimizer to the correct solution. Finally, in Fig. 11, we test the effect of additional gradient terms for ideally specular reflections, as suggested by Xing et al. [XHL*23], on a simple 2D test geometry, observing only a slight improvement with these additional terms.

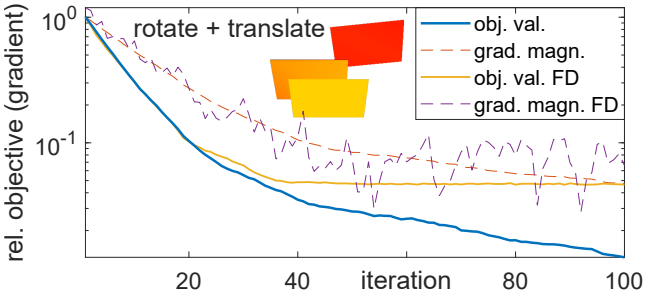


Figure 4: *Optimization convergence using our gradients compared to using finite-difference approximation for constant-step-size gradient descent. Finite-difference step is twice the descent step size. The task is to translate and rotate the receiving plate such as to reach a predefined constant temperature (note that all plates consist of only two triangles).*

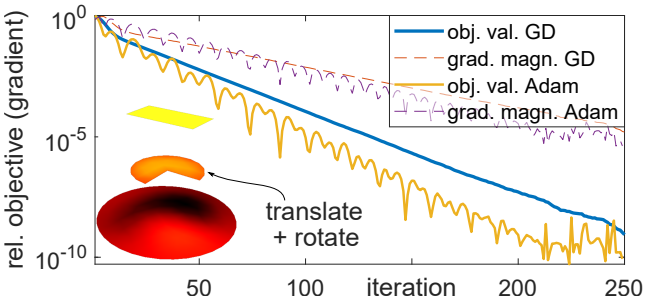


Figure 5: *Ground-truth recovery test with gradient descent (step size* 0.001*) compared to Adam (step size* 0.1*), optimizing in-plane translation and rotation of a blocking object. The blocking* 3/4*-disk is initially rotated by* 50° *around the vertical axis and translated by* 1/2 *radius to the left, and must then return to the central position shown in the inset image.*

Consequently, our current implementation does not include these terms for 3D scenes, although they could be added if required.

6.2. Example Applications

Having validated our method on basic test cases, we now move on to potential applications and more complex scenes, such as optimizing the location and rotation of buildings, trees, or the shape of a highly specular reflective facade. We create these scenes in *Blender* [Ble24] and then transfer the scene data into our software for optimization. Once the optimal configuration has been found, we then export all data back into Blender for high-quality rendering and overlaying of the color-coded temperature distribution on the surface textures. Please refer to Table 2 for an overview of basic scene properties and timings, as well as Table 3 for more detailed scene parameters.

City Block In Figure 1, we show a small city block. In this scenario, a new building is to be constructed inside a vacant area between already existing buildings. The goal is to find the position and orientation of the new building which minimizes its average surface temperature. For this optimization, we parameterize the horizontal location (XY-plane) and the vertical rotation (Z-axis) of the building. Additionally, we ensure that the optimization does not place the building outside of the construction site by a constraint that penalizes positions outside the bounds of the property. During optimization, we use the average solar irradiation over one day and compute steady-state solutions. As a result, the optimization moves the building closer to the two taller buildings on the right and adjusts its orientation to decrease its average temperature.

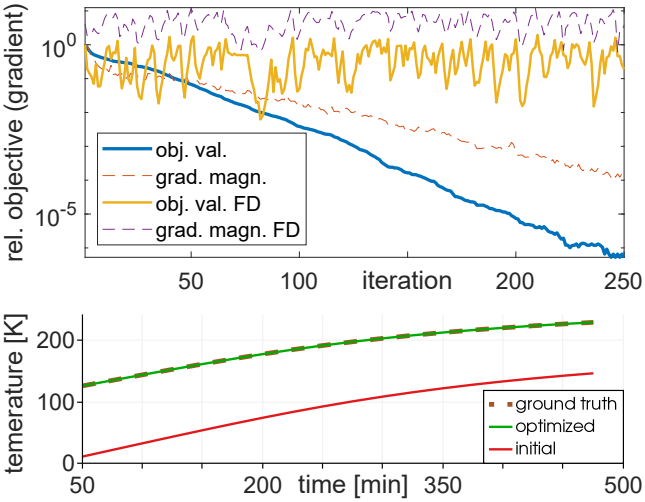
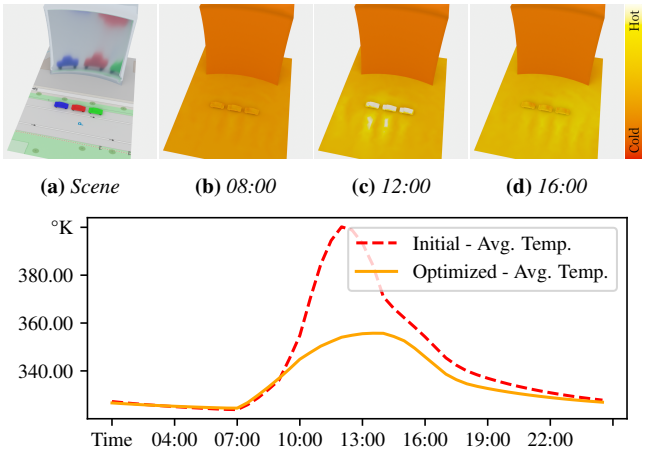


Figure 6: *Ground-truth recovery as in Fig. 5, here for a time-dependent simulation instead of the steady-state solution and* 80× *fewer traced photons. Top: our method recovers the original rotation and position of the blocking* 3/4*-disk, whereas optimizing with finite-difference approximations fails to converge due to sampling noise. Bottom: the average temperature of the* 3/4*-disk over time before and after optimization (matching the ground-truth data).*

Table 2: *Overview of the mesh size (top), and corresponding optimization step timings (bottom) for the example scenes* City Block *(Fig. 1),* Hot Spot *(Fig. 7–8), and* Village with Trees *(Fig. 9).*

Example		Fig. 1	Fig. 7–8	Fig. 9
Count	Vertices	5014	2974	2488
	Triangles	8964	4734	3114
	Photons (Millions)	2349.85	620.49	816.31
Seconds	Transp. Matrix (avg.)	0.93	0.45	0.33
	Opt. Step (avg.)	12.84	8.30	4.89
	Total Opt. Time	872.81	1161.70	880.81
	Opt. Method	Adam	GD	GD
	Nr. of Steps	64	140	180

Hot Spot In our second example scene, shown in Figure 7, we consider three cars parked in front of a tall building with a concave and highly reflective (e.g., glass) facade. This facade focuses the sun's energy directly onto the cars, causing their temperature to spike around midday. This example is inspired by the *20 Fenchurch Street* skyscraper in London, which similarly created problematic



(a) *Scene* **(b)** *08:00* **(c)** *12:00* **(d)** *16:00*

(e) *Average temperature of the three cars over time.*

Figure 7: *Transient simulation of a building with a highly specular reflective, concave facade, which creates a problematic hot spot around three cars parked in front of it (a). The images (b)–(d) show the temperature distribution (heat map) of the scene at three selected times during the day; note how the radiation is focused on the three cars and the temperature peaks around noon. To mitigate this concentrated thermal radiation, we optimize the building's facade (shape) to move the hot spot away from the cars (see Figure 8). We plot the average temperature of the three cars over the course of 24 hours (e) before and after this optimization.*

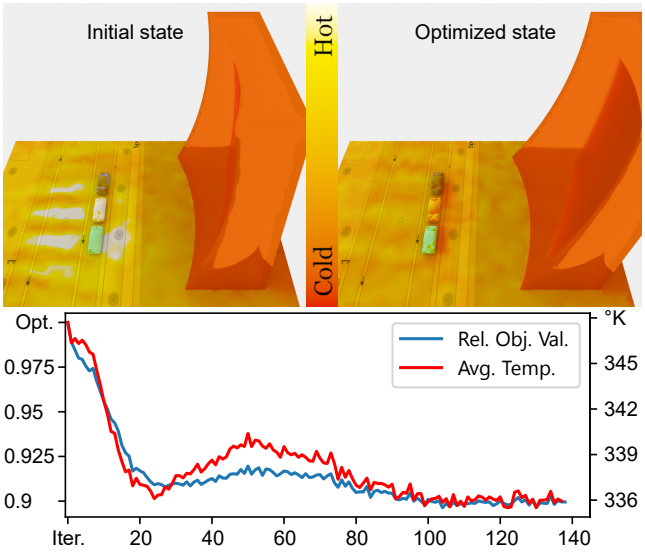


Figure 8: *Optimization of the heat spot problem shown in Fig. 7. In the initial state (top left) the tall building reflects the solar irradiation onto three cars parked in front of it. The temperature is visualized as a heat map overlay. In the optimized result (top right), the facade is skewed back to remove the focal point from the cars. The plot shows the convergence of the objective function relative to the initial state and the average temperature (in Kelvin) of the three cars, which has been reduced by* 12 *degrees.*

hot spots in the streets that damaged a car. To resolve this problematic scenario, we parameterize the facade of the building such that it can be skewed. The goal of the optimization is to adjust this skewness such as to minimize the temperature of the cars. During optimization, we simulate steady-state temperatures due to full-day average solar irradiation. The optimization (Fig. 8) skews the building's facade to the right, which successfully moves the focal point away from the cars, thus reducing their average temperature. We then compare the time evolution of the cars' temperature before and after optimization in Fig. 7.

Village with Trees In Figure. 9, we optimize the position (XY-plane) and the vertical rotation (Z-axis) of eight trees surrounding a small village to minimize the temperature of the buildings. Currently, our simulation does not model the evaporative cooling of the trees. We instead mimic the cooling effect of trees by considering them as black bodies with a constant moderate temperature, causing them to act as heat sinks in the scene. To prevent the trees from moving into the buildings during optimization, we implement a collision constraint that penalizes overlapping circumcircles for all pairs of objects. As a result of the optimization of all 24 parameters, the trees move closer to the buildings, which significantly reduces the average temperature of the buildings.

7. Discussion and Limitations

In summary, this paper presents a differentiable heat transport simulation, considering both surface conduction and radiative exchange between all scene surfaces. We compute both time-dependent and steady-state temperature distributions in virtual scenes, taking solar irradiation and temperature-dependent emission into account. Deriving the adjoint simulation and a differentiable photon-tracing method, including a novel approximate edge sampling scheme, allows us to efficiently compute derivatives of objective functions with respect to arbitrary parameterizations of the scene geometry.

Our forward simulation builds upon the publicly available framework by Freude et al. [FHR*23] and inherits a few limitations regarding the physical model. Specifically, the solar irradiation data source provides no data during the night. We, therefore, set a constant and uniform temperature for the night sky to model ambient nightly thermal radiation. While the simulation is directly derived from the well-known continuous heat transfer model, for accurate predictions, proper calibration of all parameters of heat sources and building materials would be necessary. We see such calibration and evaluation via real-world measurements as an interesting opportunity for future work.

Employing gradient-based optimization generally approaches a local minimum near the initial parameter values. In some cases, the momentum added by the Adam algorithm can increase the robustness to spurious local minima. Furthermore, noise in the transport operator and its derivatives computed by MC photon tracing can lead to noisy gradients, which sometimes have a detrimental effect on the convergence behavior. Nevertheless, our examples show that our approach delivers cleaner gradient data than a finite-difference

Table 3: *Material properties, simulation settings and optimization parameters used for the examples in Fig. 1, 8–7, and 9. Emissivity ϵ, diffuse reflectivity r_d, specular reflectivity r_s, mass density ρ [kg / m^3], specific heat capacity c_p [J / (kg K)], conductivity κ, shell thickness h [m], temperature T: free (determined by the simulation), constant (fixed uniform value) or data (fixed, possibly time-dependent, per-vertex values); optimization parameters:* ***T*** *(translation along axes),* ***R*** *(rotation around an axis),* ***Skew*** *(along an axis), or fixed (static geometry); optimization target: none or cold (minimization of T); and active optimization constraints: Collision (penalizing intersecting circumcircles), Polygon (penalizing parts outside of a polygonal boundary), or free (no constraint).*

Figure	Scene Element	ϵ	r_d	r_s	ρ	c_p	κ	h	T	Parameters	Target	Constraints
1	Ground	0.8	0.2	0.0	2000	1000	0.5	20.0	free	fixed	none	free
	Two Skyscrapers	0.2	0.1	0.7	2000	1000	0.1	0.5	free	fixed	none	free
	Other Buildings	0.8	0.2	0.0	2000	1000	0.1	0.5	free	fixed	none	free
	Center Building	0.8	0.2	0.0	2000	1000	0.1	0.5	free	**T(xy), R(z)**	**cold**	**Polygon**
	Sky Dome	1.0	0.0	0.0	1.0	1.0	0.0	1.0	**Data**	fixed	none	free
7, 8	Ground	0.8	0.2	0.0	100	200	0.5	2.0	free	fixed	none	free
	Building	0.0	0.05	0.95	2000	500	1.0	2.0	free	**Skew(x)**	none	free
	Car	0.6	0.3	0.1	100	100	0.1	0.2	free	fixed	**cold**	free
	Sky Dome	1.0	0.0	0.0	1.0	1.0	0.0	1.0	**Data**	fixed	none	free
9	Ground	0.5	0.5	0.0	1000	500	0.2	5.0	free	fixed	none	free
	Trees	1.0	0.0	0.0	200	100	0.0	0.1	**const.**	**T(xy), R(z)**	none	**Collision**
	Houses	0.6	0.4	0.0	2000	1000	3.0	1.0	free	fixed	**cold**	**Collision**
	Sky Dome	1.0	0.0	0.0	1.0	1.0	0.0	1.0	**Data**	fixed	none	free

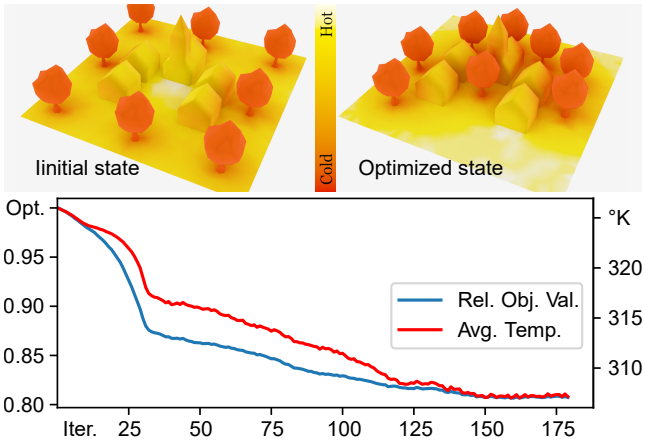


Figure 9: *In this small village, we optimize the positions of trees to reduce the average temperature of the buildings. Initially (top left) the trees are evenly spaced around the buildings. The optimization then moves the trees closer to the buildings to cool them down (top right). The plot shows the relative objective value, and the average temperature (in Kelvin) of the small chapel during the optimization.*

approximation. In the future, initial sampling of the objective function to find good parameters to start from, as well as more sophisticated optimization schemes, could help to find the global minimum. In our current implementation, we compute only the radiative transport operator on the GPU, while the rest of the solver runs on the CPU. In the future, a GPU-parallel solver could provide further performance improvements. Finally, we currently simulate heat conduction and radiation. However, depending on the application, convection may play an important part, for instance in the design of convective cooling devices. Therefore, coupling or unifying our approach with a fluid flow simulation could broaden the applicability of our inverse thermal simulation framework in the future. Overall, to our knowledge, we present the first differentiable thermal radiation model built on GPU-accelerated photon tracing, enabling gradient-based optimization of the scene geometry to support the thermal design process.

Acknowledgments

This work has received funding from the Austrian Science Fund (FWF) project F 77 (SFB "Advanced Computational Design"), the Vienna Science and Technology Fund (WWTF) project ICT22-028 ("Toward Optimal Path Guiding for Photorealistic Rendering"). Map overlays in Fig. 1, 7, 16, and 8, as well as the building geometries in Fig. 1, and 15 are based on data from *OpenStreetMap* [Ope17].

References

[Arv86] ARVO, J. "Backward ray tracing". *In SIGGRAPH '86 Developments in Ray Tracing* (1986) 2.

[BBC*22] BLONDEL, M., BERTHET, Q., CUTURI, M., et al. "Efficient and Modular Implicit Differentiation". *Advances in Neural Information Processing Systems*. Vol. 35. 2022, 5230–5242 2.

[BBC*23] BATI, M., BLANCO, S., COUSTET, C., et al. "Coupling Conduction, Convection and Radiative Transfer in a Single Path-Space: Application to Infrared Rendering". *ACM Transactions on Graphics* 42.4 (2023), 1–20. DOI: 10.1145/3592121 3.

[Beb00] BEBENDORF, M. "Approximation of boundary element matrices". *Numerische Mathematik* 86 (2000), 565–589. DOI: 10.1007/PL00005410 3, 7.

[BLD20] BANGARU, S., LI, T.-M., and DURAND, F. "Unbiased warped-area sampling for differentiable rendering". *ACM Trans. Graph.* 39.6 (2020), 245:1–245:18. DOI: 10.1145/3414685.3417833 2.

[Ble24] BLENDER DEVELOPMENT TEAM. *Blender*. Version 4.2. July 2024. URL: https://www.blender.org/ 7, 9.

[Bol84] BOLTZMANN, L. "Ableitung des Stefan'schen Gesetzes, betreffend die Abhängigkeit der Wärmestrahlung von der Temperatur aus der electromagnetischen Lichttheorie". *Annalen der Physik* 258.6 (1884), 291–294 2.

[Bra24] BRADLEY, A. M. *PDE-constrained optimization and the adjoint method*. July 2024. URL: https://cs.stanford.edu/~ambrad/adjoint_tutorial.pdf 4.

[CPLW00] CRAWLEY, D., PEDERSEN, C., LAWRIE, L., and WINKELMANN, F. "EnergyPlus: Energy Simulation Program". *Ashrae Journal* 42 (2000), 49–56 3.

[DNW24] DE LUCA, F., NATANIAN, J., and WORTMANN, T. "Ten questions concerning environmental architectural design exploration". *Building and Environment* 261 (2024), 111697. DOI: 10.1016/j.buildenv.2024.111697 1.

[FHR*23] FREUDE, C., HAHN, D., RIST, F., et al. "Precomputed Radiative Heat Transport for Efficient Thermal Simulation". *Computer Graphics Forum* 42.7 (2023). DOI: 10.1111/cgf.14957 1–4, 7, 10, 13.

[GGJ18] GUIMERA, D., GUTIERREZ, D., and JARABO, A. "A Physically-Based Spatio-Temporal Sky Model". *XXVIII Spanish Computer Graphics Conference, CEIG 2018, Madrid, Spain, June 27-29, 2018*. Eurographics Association, 2018, 29–37. DOI: 10.2312/CEIG.20181150 3.

[GJ*10] GUENNEBAUD, G., JACOB, B., et al. *Eigen v3*. (Accessed 09/27/2024). 2010. URL: https://eigen.tuxfamily.org/dox/group__MatrixfreeSolverExample.html 8.

[Gra04] GRASEDYCK, L. "Adaptive Recompression of H-Matrices for BEM". *Computing* 74.3 (2004), 205–223. DOI: 10.1007/s00607-004-0103-1 3, 7, 8.

[GTGB84] GORAL, C. M., TORRANCE, K. E., GREENBERG, D. P., and BATTAILE, B. "Modeling the interaction of light between diffuse surfaces". *ACM SIGGRAPH Computer Graphics* 18.3 (1984), 213–222 2.

[HW12] HOSEK, L. and WILKIE, A. "An analytic model for full spectral sky-dome radiance". *ACM Trans. Graph.* 31.4 (2012), 95:1–95:9. DOI: 10.1145/2185520.2185591 3.

[HW13] HOSEK, L. and WILKIE, A. "Adding a Solar-Radiance Function to the Hošek-Wilkie Skylight Model". *IEEE Computer Graphics and Applications* 33.3 (2013), 44–52. DOI: 10.1109/MCG.2013.18 3.

[Jen96] JENSEN, H. W. "Global illumination using photon maps". *Proceedings of the Eurographics Workshop on Rendering techniques' 96*. 1996, 21–30. DOI: 10.1007/978-3-7091-7484-5_3 2.

[Kaj86] KAJIYA, J. T. "The Rendering Equation". *SIGGRAPH Comput. Graph.* 20.4 (1986), 143–150 2.

[Lad24] LADYBUG TOOLS LLC. *Ladybug*. Version 0.2.10. Jan. 2024. URL: https://www.ladybug.tools/ 3.

[LADL18] LI, T.-M., AITTALA, M., DURAND, F., and LEHTINEN, J. "Differentiable Monte Carlo ray tracing through edge sampling". *ACM Transactions on Graphics* 37.6 (2018), 1–11. DOI: 10.1145/3272127.3275109 2.

[LB13] LARSON, M. G. and BENGZON, F. *The Finite Element Method: Theory, Implementation, and Applications*. Springer Berlin Heidelberg, 2013. DOI: 10.1007/978-3-642-33287-6 2.

[LH23] LIU, M. and HASEGAWA, Y. "Adjoint-based shape optimization for radiative transfer using level-set function and volume penalization method". *International Journal of Heat and Mass Transfer* 210 (2023). DOI: 10.1016/j.ijheatmasstransfer.2023.124158 2.

[LHE*24] LIPP, L., HAHN, D., ECORMIER-NOCCA, P., et al. "View-Independent Adjoint Light Tracing for Lighting Design Optimization". *ACM Transactions on Graphics* 43.3 (2024), 1–16. DOI: 10.1145/3662180 2.

[LHJ19] LOUBET, G., HOLZSCHUCH, N., and JAKOB, W. "Reparameterizing discontinuous integrands for differentiable rendering". *ACM Transactions on Graphics* 38.6 (2019), 1–14. DOI: 10.1145/3355089.3356510 2.

[LLCL19] LIU, S., LI, T., CHEN, W., and LI, H. "Soft Rasterizer: A Differentiable Renderer for Image-based 3D Reasoning". *The IEEE International Conference on Computer Vision (ICCV)* (2019). DOI: 10.1109/ICCV.2019.00780 2.

[LRPM22] LIU, D., RIST, F., POTTMANN, H., and MICHELS, D. "UrbanFlow: Designing Comfortable Outdoor Areas". (2022). DOI: 10.48550/ARXIV.2204.01117. arXiv: 2204.01117 [cs.GR] 2.

[MFS*22] MANN, S., FADEL, E., SCHOENHOLZ, S. S., et al. "∂PV: An end-to-end differentiable solar-cell simulator". *Computer Physics Communications* 272 (2022), 108232. DOI: 10.1016/j.cpc.2021.108232 3.

[MSCG24a] MILLER, B., SAWHNEY, R., CRANE, K., and GKIOULEKAS, I. "Differential Walk on Spheres". *ACM Transactions on Graphics (SIGGRAPH Asia)* 43.6 (2024). DOI: 10.1145/3687913 3.

[MSCG24b] MILLER, B., SAWHNEY, R., CRANE, K., and GKIOULEKAS, I. "Walkin' Robin: Walk on Stars with Robin Boundary Conditions". *ACM Trans. Graph.* 43.4 (2024). DOI: 10.1145/3658153. URL: https://doi.org/10.1145/3658153 3.

[Mul56] MULLER, M. E. "Some Continuous Monte Carlo Methods for the Dirichlet Problem". *The Annals of Mathematical Statistics* 27.3 (1956), 569–589. URL: http://www.jstor.org/stable/2237369 3.

[NVZJ19] NIMIER-DAVID, M., VICINI, D., ZELTNER, T., and JAKOB, W. "Mitsuba 2: A Retargetable Forward and Inverse Renderer". *Transactions on Graphics (Proceedings of SIGGRAPH Asia)* 38.6 (Dec. 2019). DOI: 10.1145/3355089.3356498 2.

[Ope17] OPENSTREETMAP CONTRIBUTORS. *Planet dump retrieved from https://planet.osm.org*. https://www.openstreetmap.org. 2017 11.

[PBC*23] PENAZZI, L., BLANCO, S., CALIOT, C., et al. "Path integrals formulations leading to propagator evaluation for coupled linear physics in large geometric models". *Computer Physics Communications* 294 (2023), 108911. DOI: 10.1016/j.cpc.2023.108911 3.

[PJH23] PHARR, M., JAKOB, W., and HUMPHREYS, G. *Physically Based Rendering, fourth edition: From Theory to Implementation*. MIT Press, 2023. ISBN: 9780262048026. URL: https://pbr-book.org 2, 5.

[RMR*24] RABACK, P., MALINEN, M., RUOKOLAINEN, J., et al. *Elmer Models Manual*. CSC – IT Center for Science. 2024. URL: http://www.nic.funet.fi/pub/sci/physics/elmer/doc/ElmerModelsManual.pdf 3, 14.

[SC20] SAWHNEY, R. and CRANE, K. "Monte Carlo Geometry Processing: A Grid-Free Approach to PDE-Based Methods on Volumetric Domains". *ACM Trans. Graph.* 39.4 (2020). DOI: 10.1145/3386569.3392374 3.

[SCJ*23] SUGIMOTO, R., CHEN, T., JIANG, Y., et al. "A Practical Walk-on-Boundary Method for Boundary Value Problems". *ACM Transactions on Graphics* 42.4 (2023), 1–16. DOI: 10.1145/3592109 3.

[SSJC23] SAWHNEY, R., SEYB, D., JAROSZ, W., and CRANE, K. "Walk on Stars: A Grid-Free Monte Carlo Method for PDEs with Neumann Boundary Conditions". *ACM Trans. Graph.* (2023). DOI: 10.1145/3592398 3.

[Ste79] STEFAN, J. "Über die Beziehung zwischen der Wärmestrahlung und der Temperatur". *Sitzungsber. Kaiserl. Akad. Wiss. Math. Naturwiss. Cl. II. Abth.* 79.3 (1879), 391–428 2.

[Wie66] WIEBELT, J. *Engineering Radiation Heat Transfer*. Holt, Rinehart and Winston, 1966 2.

[WS98] WARD, G. and SHAKESPEARE, R. "Rendering with Radiance: The Art and Science of Lighting Visualization". (1998) 3.

[WVB*21] WILKIE, A., VÉVODA, P., BASHFORD-ROGERS, T., et al. "A fitted radiance and attenuation model for realistic atmospheres". *ACM Trans. Graph.* 40.4 (2021), 135:1–135:14. DOI: 10.1145/3450626.3459758 3.

[XBLZ24] XU, P., BANGARU, S., LI, T.-M., and ZHAO, S. "Markov-Chain Monte Carlo Sampling of Visibility Boundaries for Differentiable Rendering". *SIGGRAPH Asia 2024 Conference Papers*. 2024. DOI: 10.1145/3680528.3687622 2.

[XHL*23] XING, J., HU, X., LUAN, F., et al. "Extended Path Space Manifolds for Physically Based Differentiable Rendering". *SIGGRAPH Asia 2023 Conference Papers*. SA '23. ACM, Dec. 2023. DOI: 10.1145/3610548.3618195 7, 8, 13.

[XLLX24] XING, J., LI, Z., LUAN, F., and XU, K. "Differentiable Photon Mapping using Generalized Path Gradients". *ACM Trans. Graph.* 43.6 (2024). DOI: 10.1145/3687958 2.

[YVJ24] YILMAZER, E. F., VICINI, D., and JAKOB, W. "Solving Inverse PDE Problems using Monte Carlo Estimators". *Transactions on Graphics (Proceedings of SIGGRAPH Asia)* 43 (Dec. 2024). DOI: 10.1145/3687990 3.

[YWZZ24] YU, Z., WU, L., ZHOU, Z., and ZHAO, S. "A Differential Monte Carlo Solver For the Poisson Equation". *ACM SIGGRAPH 2024 Conference Papers*. 2024. DOI: 10.1145/3641519.3657460 3.

[ZMY*20] ZHANG, C., MILLER, B., YAN, K., et al. "Path-space differentiable rendering". *ACM Transactions on Graphics* 39.4 (Aug. 2020). DOI: 10.1145/3386569.3392383 2, 4–6.

[ZRJ23] ZHANG, Z., ROUSSEL, N., and JAKOB, W. "Projective Sampling for Differentiable Rendering of Geometry". *Transactions on Graphics (Proceedings of SIGGRAPH Asia)* 42.6 (Dec. 2023). DOI: 10.1145/3618385 2.

Appendix

Spatial Discretization Following the standard finite-element approach, we first derive the weak form of Eq. (1) by multiplying with a test function v and integrating over the domain Ω, then integrating the Laplacian term by parts to obtain:

$$\int_\Omega v\rho c_p \frac{\partial T}{\partial t}\, d\mathbf{x} + \int_\Omega \nabla v \cdot \kappa \nabla T\, d\mathbf{x} - \int_{\partial\Omega} v\kappa \frac{\partial T}{\partial \mathbf{n}}\, d\tilde{\mathbf{x}} = 0. \quad (19)$$

In this paper, we consider thin-shell geometries and assume constant temperature orthogonal to the shell surface. Therefore, we split the volume integrals on the left-hand side into a surface integral over the outer shell surface Γ, while integrating along the thickness h directly. We set the test function v to zero on surfaces where the temperature is known (Dirichlet boundaries, or the inner shell surface, assuming no interior radiative transport), while substituting Eq. (2) for the remaining surfaces. We now have

$$\int_\Gamma h v\rho c_p \frac{\partial T}{\partial t}\, d\tilde{\mathbf{x}} + \int_\Gamma h\kappa \nabla v \cdot \nabla T\, d\tilde{\mathbf{x}} + \int_{\Gamma_R} v\varepsilon(\sigma T^4 - E)\, d\tilde{\mathbf{x}} = 0. \quad (20)$$

Applying a standard (piece-wise linear) finite-element discretization to the first two integrals yields the (thermal) mass matrix $\mathbf{M}$, which we lump to the diagonal, and the conduction (discrete Laplace-Beltrami) operator $\mathcal{L}$. For the radiative exchange term, we apply the discretization scheme described by Freude et al. [FHR*23], obtaining the radiative transport operator $\mathcal{T}$, which then results in the form stated in Eq. (3).

Further Test Cases In Fig. 11 and 10, we show two additional test cases, related to the ideally specular derivative formulation by Xing et al. [XHL*23] and a validation against finite-difference approximation for a basic test scene respectively. In Fig. 10, we compare our gradients, including approximate edge sampling, to finite-difference (FD) approximations over a range of FD step sizes. The optimization parameter in this case is the uniform scaling factor of the blocking plate, placed between the heat source (bottom) and temperature sensor (top). We observe the expected error behaviour, where large steps cause FD-approximation errors, while small steps suffer from rounding errors.

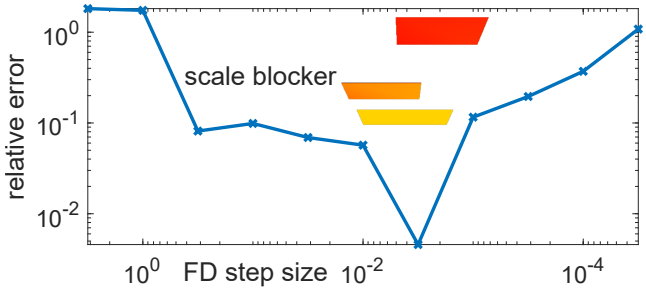


Figure 10: *Gradient error relative to finite-difference approximation of various step sizes for a simple test case where the blocker is scaled uniformly to adjust the resulting temperature on the receiver.*

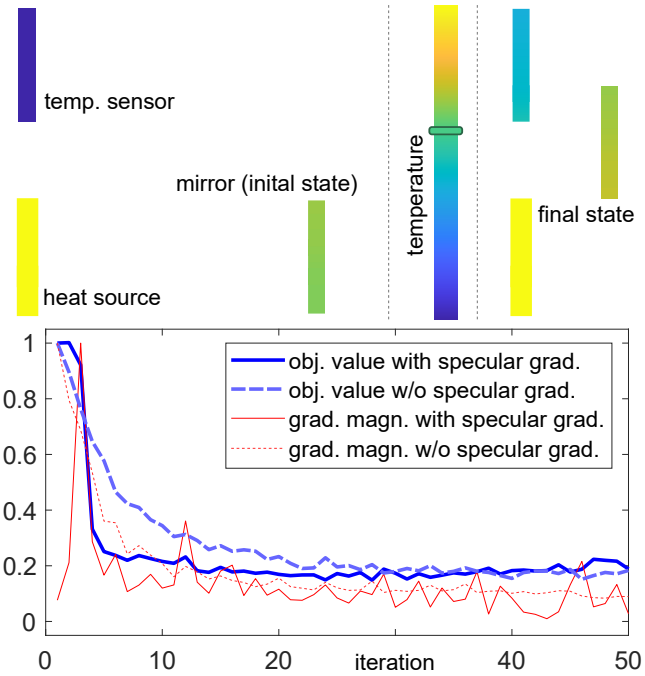


Figure 11: *Gradient-descent optimization placing 2D mirror between a source and sensor. We compare our gradients with (solid) and without (dashed) the specular constraint derivative suggested by Xing et al. [XHL*23]. The plot shows relative objective function values (blue) and gradient magnitudes (red). The marker on the temperature scale shows the target value for the sensor.*

In Fig. 12 we extend the ablation study of Fig. 2 to lower numbers of primary photons. Reducing both the primary photon count, as well as the edge-detection margin (and therefore the number of edge samples relative to the primary samples), shows a progressive degradation of convergence as the sampling noise increases. Furthermore, we validate our forward simulation model against a well-

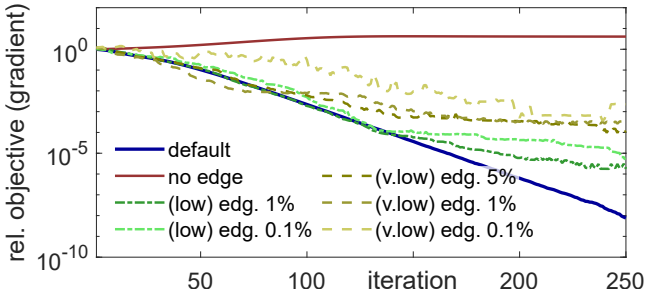


Figure 12: *Ablation study for lower numbers of primary photons. Default configuration (blue) and disabled edge sampling (red) are the same as in Fig. 2. By default we emit 256k primary photons per triangle, which we reduce to 32k ('low', dash-dots) and then to 3k ('v.low', dashed), while also testing different ratios for the edge-detection margin (default 5%).*

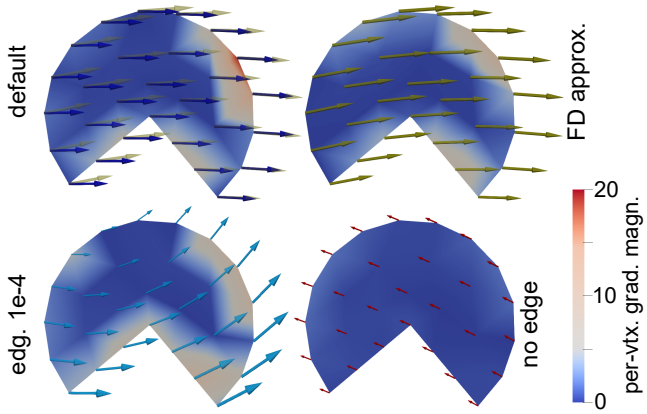


Figure 13: *Visualizing per-vertex (colors, linearly interpolated) and parameter-space gradients (mapped to vertex motion, arrows) corresponding to the initial state of the optimization in Fig. 2. Default settings (top left), finite difference approximation (top right), reduced edge-detection margin (bottom left), and disabled edge sampling (bottom right).*

established open-source finite element simulation tool in Fig. 14. Finally, Fig. 15 shows an additional rendering of the example in Fig. 1 without the reflective shader for easier readability of the temperature colormap, and Fig. 16 extends the heat-island example in Fig. 8 with three additional degrees of freedom allowing for sinusoidal deformations of the building's facade during optimization. In this example we only use 32k photons per triangle, resulting in noticeable noise in the convergence plot.

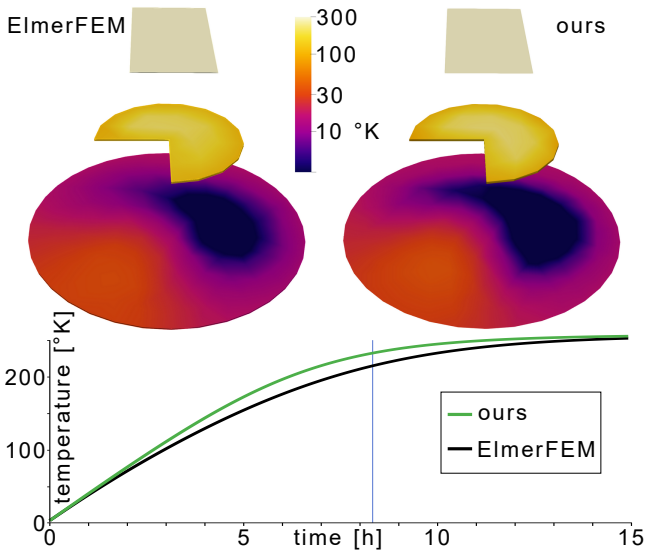


Figure 14: *Comparison of our transient simulation to the open-source finite element software* ElmerFEM *[RMR*24] on the example shown in Fig. 6. Even though we work with thin shells instead of volumetric elements, our radiative heating simulation closely matches this benchmark. The vertical marker on the plot indicates the time at which the snapshots above are taken.*

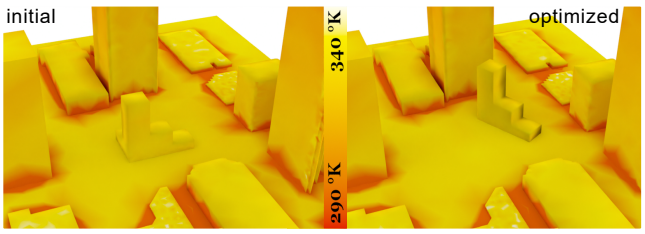


Figure 15: *Variant of Fig. 1 showing only the temperature colormap with a simple diffuse shader for comparability.*

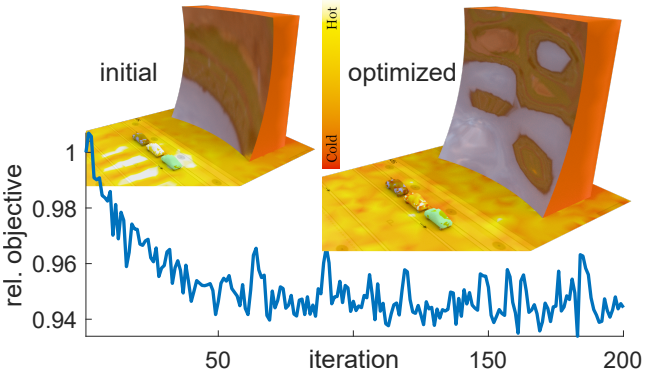


Figure 16: *Extending the parametrization of the building in Fig. 8 by additional sine waves with 2 to 5 periods across the facade area allowing for more elaborate deformation during optimization.*