TECHNISCHE UNIVERSITÄT MÜNCHEN

FAKULTÄT FÜR INFORMATIK

LEHRSTUHL FÜR COMPUTER GRAPHIK UND VISUALISIERUNG

Schattenalgorithmen im langwelligen Infrarotbereich

Andreas Herbert Klein

Vollständiger Abdruck der von der Fakultät für Informatik der Technischen Universität München zur Erlangung des akademischen Grades eines

Doktors der Naturwissenschaften (Dr. rer. nat.)

genehmigten Dissertation.

Vorsitzender		Univ.-Prof. Dr.-Ing. M. Althoff
Prüfer der Dissertation	1.	Univ.-Prof. Dr. R. Westermann
	2.	Prof. Dr. A. Nischwitz Hochschule München

Die Dissertation wurde am 27.10.2014 bei der Technischen Universität München eingereicht und durch die Fakultät für Informatik am 22.01.2015 angenommen.

Zusammenfassung

In der Luft- und Raumfahrt sowie in der Automobilindustrie werden Infrarotsensoren eingesetzt, um Umfeldmodelle für Navigations- und Tracking-Algorithmen zu erstellen. Dabei operieren Bildverarbeitungsalgorithmen auf Merkmalen in Infrarotbildern, wie z.B. Kanten. Um solche Algorithmen zu testen und zu validieren, werden neben realen auch synthetische Infrarotbilder eingesetzt, die über eine 3D-Infrarotbildgenerierung erzeugt werden. Bei Hardware-In-The-Loop (HIL) Simulationen werden die Infrarotbilder dabei direkt auf einen Infrarotsensor projiziert und setzen oft eine Bildgenerierung unter harten Echtzeitbedingungen mit 60 Bilder pro Sekunde voraus.

Die Basis für eine 3D-Infrarotbildgenerierung bildet eine thermische Simulation, bei der eine instationäre Wärmeleitung für eine 3D-Szene berechnet wird. Die resultierenden Oberflächentemperaturen werden anschließend zur Darstellung in Strahlungswerte umgewandelt. Allerdings ist die Berechnung einer 4D-Wärmeleitung sehr rechenaufwändig und im Allgemeinen nicht unter Echtzeitbedingungen durchführbar. Daher werden die Oberflächentemperaturen oft vorberechnet und in der 3D-Szene hinterlegt, was den Einsatz auf statische Geometrie beschränkt. Um die Berechnung weiter zu beschleunigen werden zudem oft wichtige thermische Effekte, wie z.B. thermische Schatten oder lokale Wärmequellen, vernachlässigt, die einen entscheidenden Beitrag zur Oberflächentemperatur liefern.

In dieser Dissertation präsentiere ich ein thermisches Modell zur Approximation des Temperaturverhaltens von Oberflächen, das für den Einsatz in einer echtzeitfähigen 3D-Infrarotbildgenerierung ausgelegt ist. Dieses thermische Modell bildet die Basis für zwei neu entwickelte Methoden zur Darstellung von thermischen Effekten, die für Szenen mit dynamischer 3D-Geometrie geeignet sind. Die erste Methode ermöglicht eine echtzeitfähige Berechnung von thermischen Schatten, die durch eine direkte Einstrahlung eines Wärmestrahlers entstehen. Dazu wird eine Ausgleichstemperatur dynamisch, über eine Interpolation von zwei vorberechneten Temperaturen mittels eines Schattenfaktors, berechnet. Um die Geschwindigkeit dieser Methode zu erhöhen, stelle ich ein Clustering-Algorithmus für die Beschleunigung der Schattenberechnung vieler Lichtquellen vor. Die zweite Methode berücksichtigt zusätzlich die indirekte Strahlung der Atmosphäre und ermöglicht lokale Strahlungsquellen. In diesem Ansatz wird die Ausgleichstemperatur dynamisch über ein Strahlungsgleichgewicht bestimmt. Beide Methoden können dazu genutzt werden, um die Genauigkeit einer echtzeitfähigen 3D-Infrarotbildgenerierung zu erhöhen.

Abstract

Infrared sensors are used in the aerospace and automotive sector to create environment models for navigation and tracking algorithms. These algorithms use image processing which operate on features of the infrared images, such as edges. In order to test and validate the image processing algorithms, synthetical infrared images are used that are generated with a 3D infrared image generation system. In hardware in the loop (HIL) simulations, these images are projected onto an infrared projector and require an image generation under hard real-time conditions of 60 frames per seconds.

The basis for an infrared image generation system is a thermal simulation that calculates an unsteady-state heat conduction for the geometry of a 3D scene. The resulting surface temperatures are then converted to radiance values. However, the calculation of a 4D heat conduction is computational expensive and usually not suitable for real-time simulations. Therefore, the surface temperatures are often precomputed, which restricts the 3D scene to static geometry. In order to accelerate the computation further, important thermal effects, such as thermal shadows or local heat sources, are often neglected. However, these thermal effect provide an important contribution for the surface temperatures.

In this thesis, I present a thermal model that is suitable for approximating the thermal behavior of a surface for the use in real-time 3D infrared image generation systems. This thermal model is the basis for two novel approaches for rendering thermal effects. These effects are suitable for scenes with dynamic 3D geometry. The first approach calculates the thermal shadow resulting from the direct irradiation of a heat source. For this purpose, an equilibrium temperature is calculated by an interpolation of two precomputed temperatures with a shadow factor. In order to increase the performance of this approach, I present a clustering algorithm for accelerating the shadow computation for many lights. The second approach for thermal effects incorporates an indirect irradiation resulting from the atmosphere. In this approach, the equilibrium temperature is calculated by assuming a radiation equilibrium. Both methods can be used to increase the fidelity of a real-time 3D image generation system.

Danksagungen

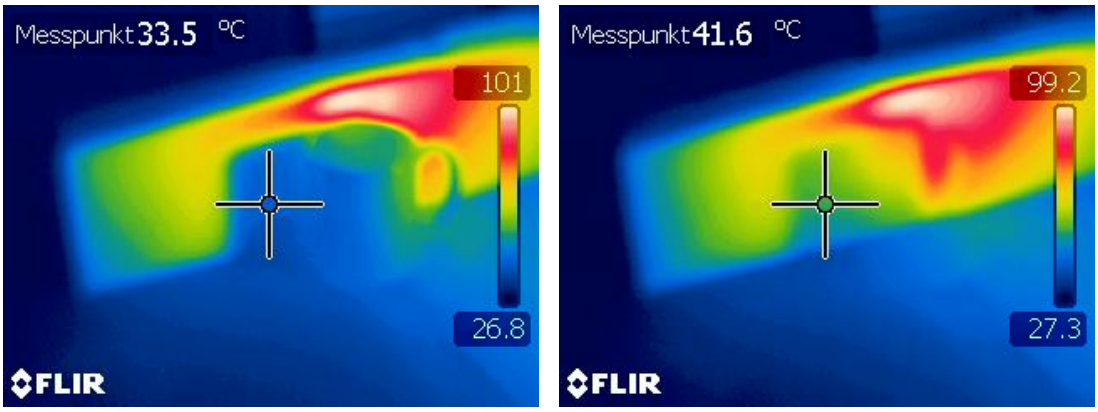
Ich möchte mich bei allen Personen bedanken, die diese Arbeit möglich gemacht haben. Zuerst möchte ich meinen Betreuer, Prof. Dr. Alfred Nischwitz, danken, der mir die Möglichkeit gegeben hat in diesem Bereich zu forschen. Ich bin sehr dankbar für seine Beiträge zu meiner Arbeit, sein Interesse und die zahlreichen Diskussionen. Des Weitern möchte ich mich bei Prof. Dr. Rüdiger Westermann bedanken, der mir die Möglichkeit zur Promotion und zum Austausch an seinem Lehrstuhl gegeben hat. Ich möchte mich weiter für die Beiträge und Ideen meiner Co-Autoren, Dr. Peter Schätz und Paul Obermeier von der MBDA Deutschland GmbH sowie Daniel Wiesenhütter von der Hochschule München, bedanken. Dabei möchte ich besonders Dr. Peter Schätz hervorheben, der mir die Daten für die Algorithmen zur Verfügung gestellt und mich bei der Umsetzung unterstützt hat. Zu guter Letzt möchte ich mich bei meinen Kollegen, Matthäus Chajdas und Dr. Tobias Pfaffelmoser von der TU München, Alexander Ederer und Siegfried Ippisch von der Hochschule München, für die zahlreichen, hilfreichen Diskussionen bedanken.

Inhaltsverzeichnis

1. Einleitung

In der Luft- und Raumfahrt sowie in der Automobilindustrie werden Infrarotsensoren eingesetzt, um Umfeldmodelle für Navigations- und Tracking-Algorithmen zu erstellen. Dabei operieren Bildverarbeitungsalgorithmen auf Merkmalen in Infrarotbildern. Diese Merkmale sind z.B. Kanten. Um diese Bildverarbeitungsalgorithmen zu testen und zu validieren, werden synthetische Infrarotbilder eingesetzt, die über eine 3D-Infrarotbildgenerierung erzeugt werden. In Hardware-In-The-Loop (HIL) Simulationen projiziert ein Infrarotprojektor die Infrarotbilder direkt auf den Infrarotsensor. Dies setzt oft eine synthetische Bildgenerierung unter harten Echtzeitbedingungen mit 60 Bilder pro Sekunde voraus.

Als Basis für eine 3D-Infrarotbildgenerierung dient eine thermische Simulation, die eine instationäre Wärmeleitung für eine 3D-Szene berechnet. Die resultierenden Oberflächentemperaturen werden anschließend zur Darstellung in Strahlungswerte umgewandelt. Allerdings ist die Berechnung einer 4D-Wärmeleitung[1] sehr rechenaufwändig und im Allgemeinen nicht unter harten Echtzeitbedingungen durchführbar. Daher werden die Oberflächentemperaturen oft vorberechnet und in der 3D-Szene hinterlegt, was den Einsatz auf statische Geometrien beschränkt. Um die Berechnung weiter zu beschleunigen werden zudem oft wichtige thermische Effekte vernachlässigt. Diese Effekte liefern jedoch entscheidende Beiträge zur Oberflächentemperatur und beeinflussen damit die Merkmale in den Infrarotbildern.



(a) Durch die Abschattung nimmt die Temperatur der Oberfläche ab.

(b) Aufheizung der Oberfläche nach Entfernen der Tasse.

Abbildung 1.1.: Beispiel eines thermischen Schattens.

Einer dieser Effekte ist ein thermischer Schatten. Ein thermischer Schatten zeichnet sich dadurch aus, dass die Oberflächentemperatur durch eine Abschattung einer

[1] 3D-Geometrie und eine eindimensionale Zeit

Wärmequelle abnimmt. Diese Temperaturveränderung ist sowohl abhängig von der Oberflächenbeschaffenheit als auch von der Dauer der Abschattung. Abbildung 1.1 zeigt ein Beispiel eines thermischen Schattens. Eine Tasse wird mit einer Wärmelampe bestrahlt und schattet einen Teil einer schwarzen Kunststoffoberfläche ab. Durch die Abschattung verringert sich die Oberflächentemperatur. Wenn die Tasse entfernt wird, steigt die Temperatur der Kunststoffoberfläche wieder an und gleicht sich mit der Umgebung aus. Durch die Temperaturunterschiede, die ein thermischer Schatten verursacht, entstehen Intensitätssprünge im Infrarotbild. Diese Intensitätssprünge erzeugen Merkmale, die eine Bildverarbeitung beeinflussen können.

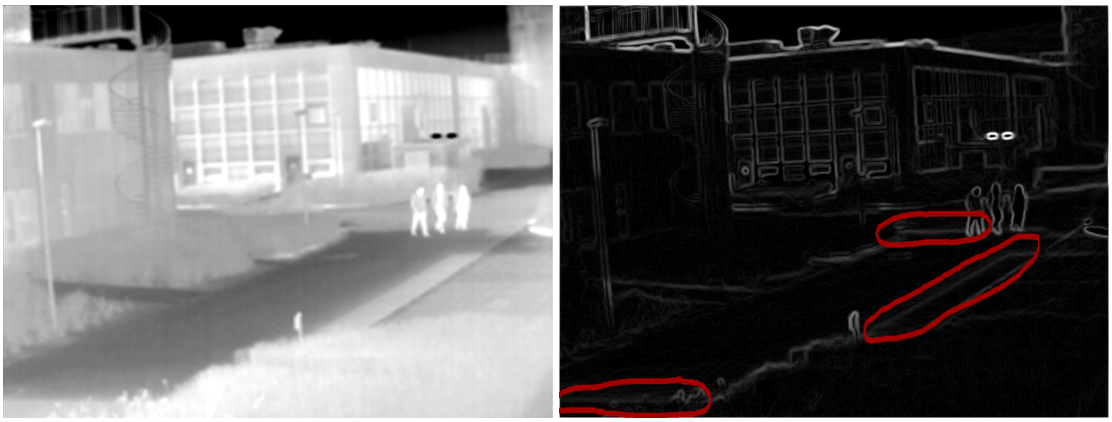


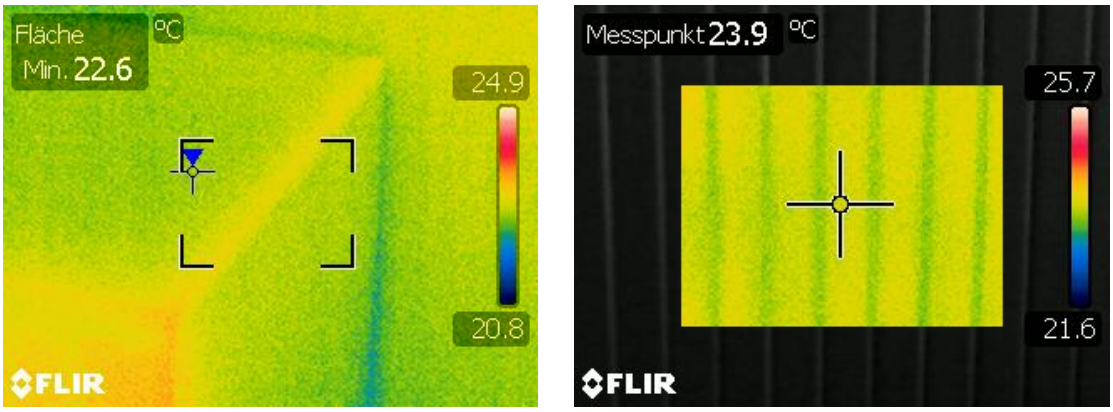
Abbildung 1.2.: Auswirkungen eines thermischen Schattens auf eine Kantenerkennung mit einen Sobel-Filter.

Abbildung 1.2 zeigt ein Beispiel einer solchen Beeinflussung. Das Gebäude auf der linken Seite schattet einen Teil der Straße ab. An den Schattenkanten entsteht ein Intensitätssprung im Bild, den ein Sobel-Filter als eine Kante detektiert.

Neben einer direkten Abschattung einer Wärmequelle ist die Umgebungsstrahlung, wie z.B. die atmosphärische Einstrahlung, eine weitere Ursache von thermischen Schatten. Diese entstehen durch einen unterschiedlich starken Einfluss der Umgebungsstrahlung auf Oberflächenpunkte. Beispielsweise sind Kanten von Objekten (Abbildung1.3b) zu einem größeren Anteil der Umgebung ausgesetzt als Vertiefungen (Abbildung 1.3a). Dadurch kühlt oder heizt sich die Oberflächentemperatur unterschiedlich auf.

1.1. Ziele und Herausforderungen

Das Ziel dieser Dissertation besteht darin, die Simulationsgenauigkeit einer echtzeitfähigen 3D-Infrarotbildgenerierung durch die Integration von thermischen Effekten zu erhöhen. Der Fokus wird auf die Entwicklung von Algorithmen für die Darstellung von thermischen Schatten gelegt. Dabei werden sowohl thermische Schatten durch eine direkte Abschattung einer Wärmequelle, als auch indirekte thermische Schatten durch eine Abschattung der Umgebungsstrahlung betrachtet.



(a) Kanten sind wärmer als ebene Flächen. (b) Vertiefungen sind kühler als ebene Flächen.

Abbildung 1.3.: Beispiele eines thermischen Schattens durch Umgebungsabschattung.

Ein wichtiger Punkt ist es dabei, eine Balance zwischen der Simulationsgenauigkeit und der Darstellungsgeschwindigkeit zu finden. Die bestehenden Verfahren zur Lösung einer 4D-Wärmeleitung, wie z.B. die Finite-Elemente-Methode, liefern zwar genaue Ergebnisse, sind aber für komplexe 3D-Szenen in der Regel nicht unter Echtzeitbedingungen zu lösen. Eine Herausforderung ist daher, eine Approximation für die 4D-Wärmeleitung zu finden, die eine geringe Abweichung zu einer Referenzlösung aufweist und echtzeitfähig ist.

Eine einfache Möglichkeit um thermische Effekte zu realisieren, besteht darin, die Oberflächentemperaturen vorzuberechnen und in die 3D-Szene mittels Texturen zu platzieren. Bei thermischen Schatten schränkt diese Vorgehensweise die Szenen auf statische Geometrien ein, da sich der Schatten ändert sobald sich ein Objekt bewegt. Ein weiteres Ziel besteht daher darin, dass die Algorithmen zur Darstellung von thermischen Schatten einen Einsatz von Szenen mit dynamischer Geometrie erlauben.

Thermische Schatten sind für die Echtzeit 3D-Computergrafik eine Herausforderung, da ein thermischer Schatten eine Schattenhistorie der Vergangenheit ist. Anders interpretiert sind thermische Schatten nichts anderes als eine Beleuchtungsrechnung mit vielen, zeitlich versetzten Lichtquellen. Dabei ist allerdings die zeitliche Reihenfolge der Lichtquellen entscheidend. Die Herausforderung besteht darin, Beschleunigungsverfahren zu entwickeln um Schatten für viele Lichtquellen unter Echtzeitbedingungen zu berechnen.

1.2. Beiträge

In dieser Dissertation werden Lösungen für eine echtzeitfähige Darstellung von thermischen Schatten präsentiert. Folgende Beiträge werden gemacht:

In Kapitel 3 wird ein thermisches Modell für die Approximation des Temperaturver-

haltens einer Oberfläche entwickelt. Die Idee besteht darin, dass Temperaturverhalten mit zwei Exponentialfunktionen zu repräsentieren. Diese Approximation wird durch eine Messung mit einer Infrarotkamera und durch eine numerische Simulation überprüft. Eine Fehlerabschätzung schließt dieses Kapitel ab.

In Kapitel 4 wird ein Algorithmus zur Berechnung von thermischen Schatten, resultierend aus einer direkten Abschattung einer Wärmequelle, vorgestellt. Dieser Algorithmus benutzt das thermische Modell aus Kapitel 3, um sich einer Ausgleichstemperatur anzunähern. Diese Ausgleichstemperatur wird über vorberechnete Strahlungs- und Schattentemperaturen bestimmt. Des Weiteren wird genauer auf die Implementierung eingegangen und eine Fehler- sowie eine Geschwindigkeitsanalyse durchgeführt.

In Kapitel 5 werden Verfahren vorgestellt, um die Schattenberechnung für 3D-Szenen zu beschleunigen. Im ersten Abschnitt wird ein Clustering-Verfahren für Lichtquellen präsentiert, das die Berechnung von Schatten für viele Lichtquellen beschleunigt. Dabei wird ein Minimum-Distanz Clustering eingesetzt um Cluster von Lichtquellen zu finden. Diese Cluster werden anschließend als ausgedehnte Lichtquelle interpretiert. Im zweiten Abschnitt wird ein Algorithmus vorgestellt, der einen weichen Schatten nur an den Schattenkanten erzeugt. Dazu werden die Schattenkanten mit Hilfe eines variablen Erosionsoperators erodiert und zwei Ansätze für die Bestimmung des Schattenfaktors implementiert.

In Kapitel 6 wird ein Algorithmus vorgestellt, der die Darstellung von thermischen Schatten, um eine Behandlung der indirekten Umgebungsstrahlung, erweitert. Des Weiteren werden lokale Wärmequellen berücksichtigt. Dazu wird eine Ausgleichstemperatur über die Strahlungsbeiträge bestimmt und die Oberflächentemperatur mit dem vorgestellten thermischen Modell angepasst. Dabei wird zuerst der Algorithmus gezeigt und anschließend genauer auf die Implementierung eingegangen. Am Ende des Kapitels wird eine Fehler- und Geschwindigkeitsanalyse durchgeführt.

1.3. Publikationsliste

Die in dieser Arbeit vorgestellten Forschungsergebnisse wurden ursprünglich in den folgenden Peer-Reviewed Journal- und Konferenzbeiträgen veröffentlicht:

1. A. Klein, A. Nischwitz, P. Schätz, P. Obermeier: "Incorporation of thermal shadows into real-time infrared three-dimensional image generation", Opt. Eng., 53(5), 053113 (2014). doi:10.1117/1.OE.53.5.053113. [49]
2. D. Wiesenhütter, A. Klein, A. Nischwitz: “LightCluster - Clustering Lights to Accelerate Shadow Computation”, Journal of WSCG, Vol.21, No.1, Pages 31-40, ISSN 1213-6972, 2013. [99]
3. A. Klein, A. Nischwitz, P. Obermeier: “Contact Hardening Soft Shadows using Erosion”, 20th WSCG International Conference on Computer Graphics, Visualization and Computer Vision. Communication proceedings, pp. 53-58, Union Agency, 2012. [48]

2. Grundlagen

In diesem Kapitel werden die, für die Arbeit notwendigen, Grundlagen beschrieben. Zuerst wird dabei auf die Wärmeübertragung eingegangen. Anschließend werden die Grundlagen der Grafikpipeline und der Hardware-Architektur einer Grafikkarte erläutert. Der nächste Abschnitt stellt die Grundlagen der Beleuchtungsrechnung in der Computergrafik und der Infrarotbildgenerierung dar. Zum Abschluss wird auf OpenSceneGraph und die numerische Stabilität eingegangen.

2.1. Wärmeübertragung

Die Wärmeübertragung beschäftigt sich mit dem Energietransport, der durch ein Temperaturungleichgewicht entsteht. Die dabei transportierte Energie wird als Wärme bezeichnet und ist als die Menge an Wärmestrom $\dot{Q}$ $[W]$ in einen Zeitraum $[t_0, t_1]$ definiert [57]:

$$Q = \int_{t_0}^{t_1} \dot{Q}(t)dt \tag{2.1}$$

Die Wärme fließt dabei immer entgegen des Temperaturgradienten ∇T eines ortsabhängigen Temperaturfelds $T(x, y, z)$. Der Temperaturgradient ist definiert als:

$$\nabla T = (\frac{\partial T}{\partial x}, \frac{\partial T}{\partial y}, \frac{\partial T}{\partial z}) \tag{2.2}$$

Beim Wärmetransport wird nach Marek und Nitsche [57] zwischen einem stoffgebunden und nicht-stoffgebunden Transport unterschieden. Zu den stoffgebunden Transporten gehört die Wärmeleitung und die Konvektion mit Fluiden. Nicht stoffgebunden ist hingegen die Wärmestrahlung. Abbildung 2.1 stellt die Transportarten grafisch dar. Bei der Wärmeübertragung gelten die Gesetze der Energie-, Masse- und Impulserhaltung.

Abbildung 2.1.: Transportarten der Wärmeübertragung.

2.1.1. Wärmeleitung

Die Wärmeleitung bezeichnet eine Übertragung von Wärme, die durch diffusive Transportprozesse entsteht, wie z.B. die Wechselwirkung auf atomarer oder molekularer Ebene [57, 77]. Diese Wärmeübertragung wird durch einen Temperaturgradienten im Festkörper oder Fluid verursacht.

Die Wärmestromdichte, die durch eine Wärmeleitung entsteht, wird durch den Fourier'schen Wärmeleitungssatz beschrieben [57, 77]:

$$\dot{q}(\vec{x}) = -\lambda(\vec{x}) \cdot \nabla T(\vec{x}) \tag{2.3}$$

Mit der Wärmeleitfähigkeit λ $[\frac{W}{mK}]$ und der Ortskoordinate $\vec{x}$.

2.1.2. Konvektion

Eine Konvektion beschreibt den massegebunden Wärmeaustausch in einem strömenden Fluid. Dabei wird zwischen einer freien und einer erzwungenen Konvektion unterschieden [57, 77]. Bei einer freien Konvektion sind das Strömungs- und das Temperaturfeld gekoppelt. Hingegen sind sie bei einer erzwungenen Konvektion entkoppelt.

Die Wärmestromdichte an der Oberfläche eines Körpers, die durch eine Konvektion entsteht, wird durch das Newton'sche Abkühlungsgesetz beschrieben [57, 77]:

$$\dot{q}_W = \alpha(T_W - T_\infty) \tag{2.4}$$

Mit dem Wärmeübergangskoeffizienten α $[\frac{W}{m^2K}]$, sowie der Wandtemperatur T_w und der Fluidtemperatur T_∞.

2.1.3. Wärmestrahlung

Eine Wärmestrahlung bezeichnet den Wärmeaustausch, der durch eine elektromagnetische Strahlung entsteht. Dies wird durch das Stefan-Bolzmann Strahlungsgesetz beschrieben [57, 77]:

$$\dot{q}_{12} = \Sigma_{12}(T_1^4 - T_2^4) \tag{2.5}$$

Mit dem Austauschkoeffizienten Σ_{12} $[\frac{W}{m^2K^4}]$, sowie den Temperaturen der Oberflächen T_1 und T_2. Im Austauschkoeffizienten fließen unter anderem die Emissionsgrade der Oberflächen sowie die Orientierung der Oberflächen zueinander ein. Beispielsweise ist der Austauschkoeffizient von zwei planparallelen Platten mit großer Ausdehnung nach Polifke und Kopitz [77]:

$$\Sigma_{12} = \frac{\sigma}{\frac{1}{\epsilon_1} + \frac{1}{\epsilon_2} - 1} \tag{2.6}$$

2.1.4. Instationäre Wärmeleitung

Im Rahmen dieser Arbeit soll ein ort- und zeitabhängiges Temperaturfeld $T(\vec{x}, t)$ mit $\vec{x} = (x, y, z)$ über eine instationäre Wärmeleitung berechnet werden. In der Wärmelehre wird dies mathematisch mit Hilfe einer Fourier'schen Wärmeleitungsgleichung beschrieben [57, 77]:

$$\frac{\partial T}{\partial t} - a \cdot \Delta T = 0 \Leftrightarrow \frac{\partial T}{\partial t} - a \cdot \left(\frac{\partial^2 T}{\partial x^2} + \frac{\partial^2 T}{\partial y^2} + \frac{\partial^2 T}{\partial z^2} \right) = 0 \tag{2.7}$$

Wobei a die Temperaturleitfähigkeit und Δ der Laplace-Operator ist. Die Temperaturleitfähigkeit a $[\frac{m^2}{s}]$ ist definiert als:

$$a := \frac{\lambda}{\rho c} \tag{2.8}$$

Mit der Wärmeleitfähigkeit λ, der Dichte ρ $[\frac{kg}{m^3}]$, der Wärmekapazität c $[\frac{J}{kgK}]$.

Zusätzlich gelten folgende Anfangs- und Randbedingungen [57, 77]:

Anfangsbedingung: Die Anfangsbedingung gibt die anfängliche Temperatur zum Zeitpunkt $t = 0$ vor:

$$T(\vec{x}, t = 0) = T_0(\vec{x}) \tag{2.9}$$

Dirichlet'sche Randbedingung (1. Art): Die Dirichlet'sche Randbedingung gibt eine Temperatur T_w am Rand $x = x_0$ vor:

$$T(x = x_0, y, z, t) = T_W(y, z, t) \tag{2.10}$$

Neumann'sche Randbedingung (2. Art): Mit der Neumann'schen Randbedingung wird eine Wärmestromdichte am Rand definiert:

$$\dot{q_W} = -\lambda \left. \frac{\partial T}{\partial x} \right|_{x=x_0} \tag{2.11}$$

Mit der Wärmeleitfähigkeit λ.

Spezialfall Adiabaise:

$$\left. \frac{\partial T}{\partial x} \right|_{x=x_0} = 0 \tag{2.12}$$

Newton'sche Randbedingung (3. Art): Der Wärmeübergang von einer Oberfläche an ein Fluid wird mit der Newton'schen Randbedingung beschrieben:

$$\alpha \cdot (T_W - T_\infty) = \lambda \left.\frac{\partial \theta}{\partial x}\right|_{x=x_0} \tag{2.13}$$

Mit dem Wärmeübergangskoeffizienten α, der Wärmeleitfähigkeit λ, sowie der Wandtemperatur T_W und der Fluidtemperatur T_∞.

Im Folgenden werden Lösungsansätze am Beispiel einer unendlich ausgedehnten, ebenen Platte gezeigt. Für Lösungsansätze weiterer geometrischer Grundobjekte wird auf Polifke und Kopitz [77] verwiesen.

2.1.4.1. Dimensionslose Formulierung

Mit Hilfe einer Entdimensionierung kann die Lösung von instationären Wärmeleitungsproblemen vereinfacht werden. Dazu werden folgende dimensionslose Kennzahlen eingeführt [77, 10, 57]:

Fourier-Zahl: Die Fourier-Zahl beschreibt eine dimensionslose Zeit.

$$F_o := \frac{at}{L^2} \tag{2.14}$$

Mit der charakteristischen Länge L und der Temperaturleitfähigkeit a.

Biot-Zahl: Die Biot-Zahl definiert das Verhältnis des Wärmeleitwiderstandes R_λ zum Wärmeübergangswiderstand R_α:

$$Bi := \frac{R_\lambda}{R_\alpha} = \frac{\alpha L}{\lambda} \tag{2.15}$$

Mit der charakteristischen Länge L, der Wärmeleitfähigkeit λ und dem Wärmeübergangskoeffizient α.

Dimensionslose Ortskoordinate: Eine Ortskoordinate x kann entdimensioniert werden, indem sie durch die charakteristische Länge L geteilt wird:

$$\xi := \frac{x}{L} \tag{2.16}$$

Dimensionslose Temperatur: Die Temperatur kann entdimensioniert werden, indem sie normiert wird:

$$\theta := \frac{T - T_\infty}{T_0 - T_\infty} \tag{2.17}$$

Mit $T_0 = T(r, t = 0)$ und $T_\infty = T(r, t \to \infty)$

Für eine dimensionslose Formulierung ist es weiter notwendig, die Anfangs- und Randbedingungen zu entdimensionieren. Dies kann mit Hilfe der dimensionslosen Kennzahlen geschehen.

Anfangsbedingung:

$$\theta(\xi, Fo = 0) = \theta_0 \tag{2.18}$$

Dirichlet'sche Randbedingung (1. Art):

$$\theta(\xi = 1, Fo) = \theta_w \tag{2.19}$$

Neumann'sche Randbedingung (2. Art):

$$\left.\frac{\partial\theta}{\partial\xi}\right|_{\xi=1} = -\frac{\dot{q}_w L}{\lambda(T_0 - T_\infty)} \tag{2.20}$$

Spezialfall Adiabasie (für $\xi = 0$ analog):

$$\left.\frac{\partial\theta}{\partial\xi}\right|_{\xi=1} = 0 \tag{2.21}$$

Newton'sche Randbedingung (3. Art):

$$Bi \cdot \theta(\xi = 1, Fo) + \left.\frac{\partial\theta}{\partial\xi}\right|_{\xi=1} = 0 \tag{2.22}$$

2.1.4.2. Lösungen

Eine analytische Lösung zur instationären Wärmeleitung einer unendlich ausgedehnten, ebenen Platte, in dimensionsloser Formulierung mit den Randbedingungen, liefert die Fourier-Reihe [57, 77]:

$$\theta(\xi, Fo) = \sum_{k=1}^{\infty} C_k cos(\delta_k \xi) e^{-\delta_k^2 Fo} \tag{2.23}$$

Wobei C_k die Koeffizienten und δ_k der Eigenwert der Ordnung k ist. Die Eigenwerte werden über die transzendente Gleichung ermittelt:

$$tan(\delta) = \frac{Bi}{\delta} \tag{2.24}$$

Die Koeffizienten C_k sind wie folgt definiert:

$$C_k = \frac{\int_0^1 \theta(0,\xi) cos(\delta_k \xi) d\xi}{\int_0^1 cos^2(\delta_k \xi) d\xi} \tag{2.25}$$

Wobei $\theta(0,\xi)$ die anfängliche Temperaturverteilung des Festkörpers ist.

In der Praxis wird die Fourier-Differenzialgleichung jedoch mit numerischen Verfahren gelöst, wie z.B. der Finite-Volumen oder der Finite-Elemente Methode. Die Grundlagen hierzu werden allerdings im Rahmen dieser Arbeit nicht erläutert. Stattdessen wird auf Polifke und Kopitz [77] verwiesen.

2.2. Grafikpipeline

Im folgenden Abschnitt wird auf die Grundlagen der OpenGL 4.3 Grafikpipeline eingegangen. Im Anschluss werden die Grundlagen der Hardware Architektur einer Grafikkarte erläutert.

Abbildung 2.2 zeigt einen Überblick der OpenGL 4.3 Grafikpipeline. Die Grafikpipeline kann in die Stufen Vertex-Verarbeitung, Tesselierung, Primitiven-Verarbeitung, Rasterisierung, Fragment-Verarbeitung und Pixel-Verarbeitung unterteilt werden (siehe [90, 19, 4]). Im Folgenden wird genauer auf die einzelnen Stufen eingegangen.

Abbildung 2.2.: Überblick über die Grafikpipeline von OpenGL 4.3.

2.2.1. Vertex-Verarbeitung

Bei der Vertex-Verarbeitungsstufe werden zuerst Vertex-Daten aus Buffern, wie z.B. Vertex-Buffer Objects oder Vertex-Array Objects, zusammengesetzt [90]. Anschließend wird der Vertex an einen Vertex-Shader übergeben, welcher z.B. eine Koordinatentransformation oder eine Vertex-Beleuchtungsrechnung durchführt. Diese Vertex-Shader sind frei programmierbar. Nach der Vertex-Verarbeitung wird das Vertex an die Tesselierungsstufe übergeben.

2.2.2. Tesselierung

Die Tesselierungsstufe erlaubt es, sogenannte Patches von Vertex-Daten in Primitive zu unterteilen. Dazu werden zwei programmierbare Shader und eine nichtprogrammierbare Stufe eingesetzt. Eine Tesselierung läuft wie folgt ab: Zuerst wird über den Tessellation-Control-Shader ein innerer und ein äußerer Tesselierungsfaktor für einen Patch erzeugt. Anschließend unterteilt die nichtprogrammierbare Stufe die Patches in Primitive. Dabei wird ein baryzentrisches Koordinatensystem genutzt [90]. Im Tessellation-Evaluation-Shader werden anschließend die Koordinaten aus dem baryzentrischen Koordinatensystem in das Zielkoordinatensystem umgewandelt.

2.2.3. Primitiven-Verarbeitung und Rasterisierung

Das Ziel der Primitiven-Verarbeitungsstufe besteht darin, anhand der Vertex-Daten geometrische Primitive, wie z.B. Dreiecke oder Linien, zu erzeugen. Der Anwender kann auf diesen Prozess über einen Geometry-Shader zugreifen und Primitive erzeugen, verwerfen oder modifizieren. Nach dem die Primitven erzeugt sind, wird ein Viewport-Clipping durchgeführt und die Primitive über ein Rasterisierungsverfahren zu Fragmenten rasterisiert. Bei der Rasterisierung werden auch die Vertex-Attribute interpoliert [90].

2.2.4. Fragment-Verarbeitung

In der Fragment-Verarbeitungsstufe werden die, von der Rasterisierung erzeugten, Fragmente verarbeitet. Die Fragmente werden über einen Fragment-Shader modifiziert. In der Regel wird ein Fragment-Shader zur Fragment-Beleuchtung und für das Texture-Mapping verwendet. Anschließend wird der Z- und Stencil-Test durchgeführt und das Fragment gegebenenfalls verworfen oder an die Pixel-Verarbeitungsstufe weitergegeben.

2.2.5. Pixel-Verarbeitung

Das Ziel der Pixel-Verarbeitungsstufe liegt darin, die generierten Pixel in den Framebuffer zu übertragen. Der Anwender kann diesen Prozess über OpenGL-Befehle steuern. Unter anderem können Farbraumkonvertierungen in sRGB vorgenommen oder die Farbmischung, über eine Blending-Funktion, angepasst werden [90].

2.2.6. Hardware Architektur einer Grafikkarte

Das Pipeline-Modell einer Grafikkarte eignet sich nach Owens et al. [68] besonders für einen Aufgaben- und Datenparallelismus. Die Aufgabenparallelität entsteht dadurch,

dass einzelne Stufen der Pipeline, wie z.B. die Vertex- und die Fragment-Stufe, parallel bearbeitet werden können. In der Vergangenheit wurden dazu eigene Hardwareeinheiten für die Vertex- und die Fragment-Stufe eingesetzt.

Eine moderne Graphics Processing Unit (GPU) arbeitet vorwiegend nach dem Prinzip der Datenparallelität [68]. Dabei wird das gleiche Programm parallel für mehrere Eingabedaten ausgeführt (Single Program Multiple Data (SPMD)). Der Wandel von der Aufgaben- zur Datenparallelität ist bedingt durch die Einführung einer Unified-Shader Architektur, bei der alle programmierbaren Stufen auf der gleichen Hardwareeinheit basieren. Diese Architektur besitzt den Vorteil, dass dem Anwender nun alle programmierbaren Stufen einer Grafikkarte für die gleiche Aufgabe zur Verfügung stehen. Dadurch können Geschwindigkeitsengpässe durch ungleiche Lastverteilungen vermieden werden [68].

Im Folgenden wird eine typische GPU Hardware Architektur am Beispiel einer NVIDIA GeForce GTX 680 Grafikkarte beschrieben. Die GPU ist hierarchisch in Cluster, Multiprozessoren und Streamprozessoren gegliedert [64, 63]. Sie besitzt vier Cluster, bestehend aus je zwei Multiprozessoren und einer Rasterisierungs-Engine. Des Weiteren besitzt die GPU vier Speichercontroller mit einem 128 KByte großen L2-Cache und 32 Raster-Operation-Units (ROPs). Der GDDR5-Speicher ist über einen 256 Bit breiten Speicherbus mit der GPU verbunden [64].

Jeder Multiprozessor besteht wiederum aus 192 Streamprozessoren, 4 Raster-Operation-Units (ROPs), 16 Textureinheiten, sowie 32 Special-Function-Units und 32 Load-Store-Units. Des Weiteren besitzt es einen 64 KByte großen L1-Cache [64]. Die eigentlichen Berechnungen finden auf Streamprozessoren statt. Diese besitzen eine Arithemtic-Logic-Unit (ALU) und eine Floating-Point-Unit (FPU). Die Hardware eines Streamprozessors implementiert den IEEE-754 Floating-Point Standard und ist in der Lage mit 64-Bit Floating Point Zahlen zu rechnen [63].

2.3. Beleuchtungsrechnung

Eine Beleuchtungsrechnung bildet die Basis für eine Bildgenerierung. Sie bestimmt, wie viel Strahlung von einem Oberflächenpunkt zum Betrachter über elektromagnetische Wellen ausgesandt wird. Kajia [46] benutzt die Radiometrie um die Rendering-Gleichung aufzustellen. Mit der Notation $p \to q := \frac{q-p}{\|q-p\|}$ nach Eisemann et al. [28] ist die Rendering-Gleichung wie folgt definiert:

$$L_o(p,\omega) = L_e(p,\omega) + \int_S f_r(p,\omega,p \to q) L_o(p,q \to p) G(p,q) V(p,q) dq \tag{2.26}$$

Wobei p ein Punkt auf einer Oberfläche, ω eine Richtung, f_r eine bidirektionale Reflektanzverteilungsfunktion (BRDF), V eine Sichtbarkeitsfunktion ist. Der geometrische Term G ist definiert als:

$$G(p,q) = \frac{cos(n_p, p \to q)cos(n_q, q \to p)}{\|p-q\|^2} \tag{2.27}$$

Mit dem Normalenvektor n. Für eine globale Beleuchtungsrechnung gilt es die Rendering-Gleichung zu lösen. Dazu gibt es in der Literatur Standardverfahren, wie z.B. Path Tracing oder Monte Carlo Methoden, auf die im Rahmen der Arbeit nicht weiter eingegangen wird. Stattdessen wird auf Pharr und Humphreys [74] verwiesen.

In der Echtzeit-Computergrafik wird stattdessen oft nur eine lokale Beleuchtungsrechnung durchgeführt. Im Gegensatz zur globalen Beleuchtung, ist die lokale Beleuchtung eines Oberflächenpunkts unabhängig von anderen Oberflächenpunkten. Das bedeutet vor allem auch, dass keine indirekte Beleuchtung durch eine Reflexion aus der Umgebung berücksichtigt wird. Auch Schatten, die durch eine Verdeckung einer Lichtquelle entstehen, bleiben unberücksichtigt. Diese globalen Beleuchtungseffekte können aber in ein lokales Beleuchtungsmodell integriert werden. Auf diese Methode wird in dieser Arbeit zurückgegriffen.

2.3.1. Schatten

Ein wichtiger Beitrag für eine realistische Beleuchtung sind Schatten, die durch eine Verdeckung einer Strahlungsquelle entstehen. Ein Schatten gibt wichtige Hinweise auf die Lage und die Form eines Objekts (siehe [28, 4]). Es wird zwischen einem Halbschatten (Penumbra) und einem Kernschatten (Umbra) unterschieden. Bei einem Kernschatten wird die Strahlungsquelle vollständig verdeckt. Hingegen ist die Strahlungsquelle im Halbschatten noch teilweise sichtbar. Des Weiteren zeichnet sich ein Halbschatten dadurch aus, dass dieser mit der Entfernung zwischen Schattenspender und Schattenempfänger wächst (Abbildung 2.3). Ein Halbschatten wird in der Literatur häufig auch als weicher Schatten bezeichnet (siehe [28]).

Nach Eisemann et al. [28] kann mit der Rendering-Gleichung (Gleichung 2.26) ein Schatten, für eine direkte Beleuchtung einer Lichtquelle, wie folgt definiert werden:

$$L_o(p,\omega) = L_e(p,\omega) + \int_L f_r(p,\omega,p \to l)G(p,l)dl \cdot \frac{1}{|L|}\int_L L_e(l, l \to p)V(p,l)dl \tag{2.28}$$

Mit einer Lichtquelle L. In der Praxis wird jedoch angenommen, dass die emittierte Radianz homogen über die Fläche der Lichtquelle ist. Dadurch kann die Sichtbarkeitsfunktion wie folgt vereinfacht werden [28]:

$$V_L(p) = \frac{1}{|L|}\int_L V(p,l)dl \tag{2.29}$$

Die Lösung der Sichtbarkeitsfunktion $V_p(p)$ für einen Punkt p wird im Folgenden Schattenfaktor genannt und besitzt den Wertebereich von $[0,1]$. Null bedeutet dabei

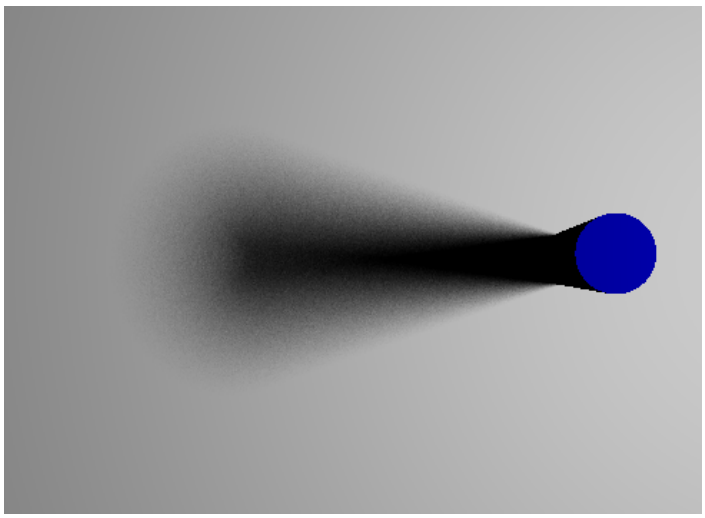


Abbildung 2.3.: Beispiel für einen Schatten, der durch eine Säule entsteht. Die Halbschattenbreite wächst mit der Entfernung zwischen Schattenspender und Schattenempfänger.

ein Kernschatten und Eins eine volle Beleuchtung. Ein Halbschatten wird durch einen Wert zwischen Null und Eins angegeben.

2.3.1.1. Ray Tracing

Eine Möglichkeit, um einen Schattenfaktor zu bestimmen, ist das Ray Tracing [98, 18]. Die Idee besteht darin, von jedem Oberflächenpunkt aus die Lichtquelle mit Strahlen abzutasten und die Strahlen auf einen Schnittpunkt mit einer Geometrie zu testen. Wenn ein Schnittpunkt mit einer Geometrie vorhanden ist, bevor die Lichtquelle erreicht wird, ist dieser Abtastpunkt der Lichtquelle verdeckt. Der Schattenfaktor ergibt sich über das Verhältnis der sichtbaren Abtastpunkte zu der Gesamtanzahl der Abtastpunkte [18]. Abbildung 2.4 stellt diesen Vorgang grafisch dar.

Der Vorteil bei dieser Vorgehensweise besteht darin, dass ein Aliasing im Schatten, durch die Verfolgung der Strahlen für jeden Oberflächenpunkt, reduziert wird. Allerdings ist ein Schnittpunkttest eines Strahls mit der Geometrie aufwändig. Dieser Aufwand kann durch Beschleunigungsstrukturen reduziert werden (siehe [74]). Für Halbschatten sind jedoch eine große Anzahl an Strahlen je Oberflächenpunkt notwendig, um einen weichen Graustufenübergang zu erzeugen. Dadurch eignet sich Ray Tracing besonders für nicht-echtzeit Anwendungen.

2.3.1.2. Shadow Mapping

Ein Standardverfahren für die Schattenberechnung in der Echtzeit Computergrafik ist das Shadow Mapping [100]. Die Idee von Shadow Mapping besteht darin, den

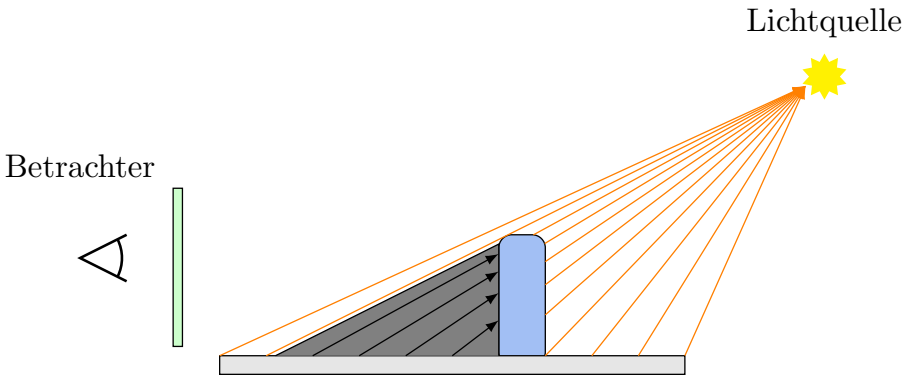
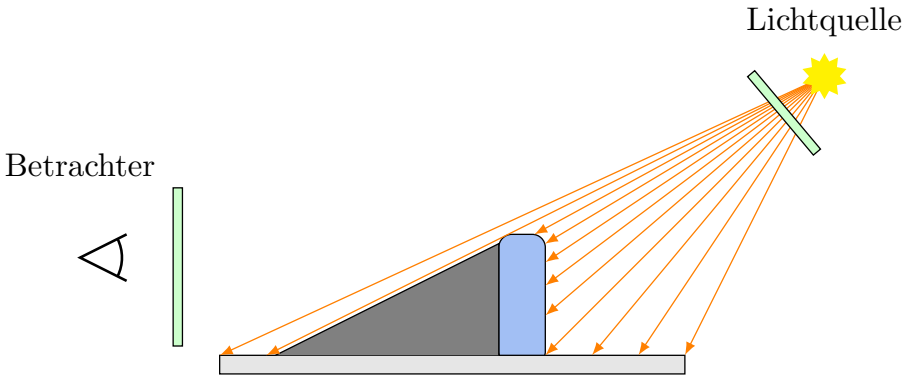


Abbildung 2.4.: Schatten mit Ray Tracing. Von jedem Oberflächenpunkt aus werden Strahlen zur Lichtquelle verfolgt und überprüft, ob ein Strahl verdeckt ist.

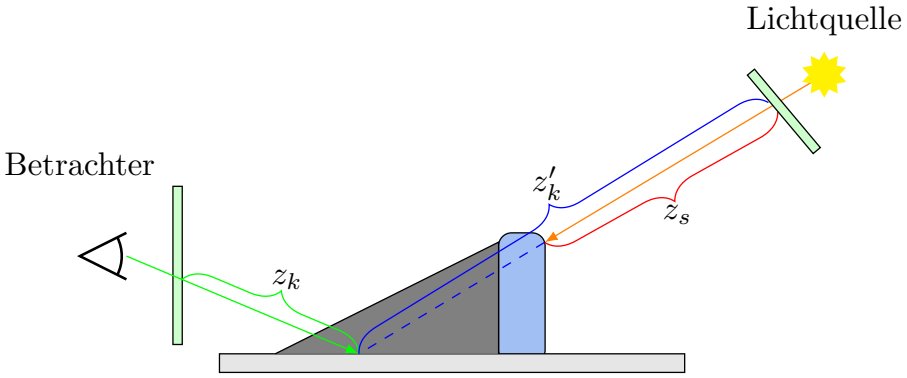
Sichtbarkeitstest umzukehren. Anstatt zu prüfen, ob eine Lichtquelle von einem Oberflächenpunkt sichtbar ist, wird getestet, ob der Oberflächenpunkt von der Lichtquelle aus sichtbar ist. Abbildung 2.5a verdeutlicht diesen Vorgang.

Abbildung 2.5b zeigt wie das Sichtbarkeitsproblem mit Hilfe von Shadow Mapping gelöst wird. Die Szene wird zunächst aus Sicht der Lichtquelle abgetastet und die Tiefenwerte z_s der Szenengeometrie in eine 2D-Textur (Shadow Map) diskretisiert. Diese Tiefenwerte werden anschließend mit den Tiefenwerten z_k aus der Betrachtersicht verglichen, die vor dem Vergleich in das Koordinatensystem der Lichtquelle transformiert werden (z'_k). Ein Punkt auf einer Oberfläche ist genau dann im Schatten, wenn der Tiefenwert in der Shadow Map kleiner als der transformierte Tiefenwert aus Betrachtersicht ist.

Der Vorteil von Shadow Mapping besteht darin, dass der Sichtbarkeitstest durch eine Hardware-Rasterisierung mit einer Grafikkarte gelöst werden kann. Allerdings verursacht Shadow-Mapping Aliasing-Artefakte. Aufgrund der Diskretisierung einer Szene in eine 2D-Textur aus Tiefenwerten entsteht eine Unterabtastung, die zu Block-Artefakten im Schatten führen kann. Des Weiteren entsteht ein projektives und ein perspektivisches Aliasing [28]. Das projektive Aliasing ist von der Ausrichtung einer Oberfläche abhängig und dann am größten, wenn eine Oberfläche fast parallel zur Lichtrichtung ist. Nach Eisemann et al. [28] kann projektives Aliasing durch eine höhere Abtastrate in diesen Bereichen reduziert werden, wie z.B. durch Adaptive Shadow Maps [31]. Das perspektivische Aliasing entsteht durch die perspektivische Projektion, die eine höhere Abtastrate im Bereich der Near-Plane besitzt [28]. Bereiche, die an der Far-Plane liegen, werden mit einer niedrigeren Abtastrate behandelt. Durch eine Unterabtastung dieser Bereiche entstehen Artefakte im Schatten. Das perspektivische Aliasing kann durch eine Anpassung der Projektion durch ein Fitting-Verfahren (z.B. LISPM [101]) oder durch den Einsatz mehrerer Shadow Maps (z.B. Cascaded Shadow Maps [29]) reduziert werden [28].



(a) Beim Shadow Mapping wird überprüft, ob ein Oberflächenpunkt von der Lichtquelle aus sichtbar ist.



(b) Der Sichtbarkeitstest beim Shadow Mapping erfolgt durch einen Vergleich der transformierten Tiefenwerte z'_p aus der Betrachtersicht und den Tiefenwerten z_s der Shadow Map. Ein Oberflächenpunkt ist im Schatten, wenn z_s kleiner als z'_k ist.

Abbildung 2.5.: Funktionsweise des Shadow Mappings

Weiche Schatten

In der Literatur gibt es eine Vielzahl von Verfahren für weiche Schatten mit Shadow Mapping. Im Folgenden werden nur die Grundlagen der, in dieser Arbeit verwendeten, Verfahren erläutert. Für eine detaillierte Diskussion anderer Verfahren wird auf Eisemann et al. [28] verwiesen.

Um weiche Schatten mit Shadow Mapping zu bestimmen, führen Reeves et al. [79] Percentage Closer Filtering (PCF) ein. Die Idee besteht darin, mehrere Schattentests innerhalb eines Fensters in der Shadow Map durchzuführen. Der Schattenfaktor ergibt sich über eine Mittelung der Ergebnisse des Schattentests.

Diese Vorgehensweise wurde von Fernando [30] mit Percentage Closer Soft Shadows (PCSS) für variable Halbschatten erweitert. Die Idee besteht darin, dass zuerst ein mittlerer Tiefenwert der Schattenspender durch eine Blockersuche bestimmt wird. Dazu wird die Shadow Map innerhalb eines Fensters abgetastet und Schattenspender

(a) Schatten mit konstanter Halbschattenbreite durch Percentage Closer Shadows.

(b) Schatten mit variabler Halbschattenbreite durch Percentage Closer Soft Shadows.

Abbildung 2.6.: Vergleich der Schatten zwischen PCF und PCSS. Durch die konstante Halbschattenbreite wirken die Schatten von PCF unrealistisch.

gesucht. Wenn ein Schattenspender vorhanden ist, wird der Tiefenwert gespeichert und über alle gefundenen Schattenspender gemittelt. Mit Hilfe dieses mittleren Tiefenwerts kann die Halbschattenbreite über den Strahlensatz abgeschätzt werden. Die Halbschattenbreite ergibt sich wie folgt:

$$\omega_P = \omega_L \frac{z_r - z_b}{z_b} \tag{2.30}$$

Wobei z_r der Tiefenwert des Oberflächenpunkts, z_b die mittlere Blockertiefe und ω_L die Lichtgröße ist. Die Halbschattenbreite wird anschließend genutzt um das Fenster für PCF zu skalieren. Abbildung 2.6 zeigt den Unterschied zwischen weichen Schatten mit PCF und PCSS. Durch eine variable Halbschattenbreite wird eine realistischere Schattendarstellung erreicht.

Variance Shadow Maps (VSM) [52] benutzt ein statistisches Modell um einen weichen Schatten zu erzeugen. Dazu wird eine Shadow Map mit linearisierten Tiefenwerten gerendert, die zusätzlich den quadrierten Tiefenwert enthält. Anschließend wird die Shadow Map mit einem Box-Filter innerhalb eines Fenster gefiltert, um die Momente M_1 und M_2 zu berechnen. Über diese Momente kann der Erwartungswert $\mu = M_1$ und die Varianz $\sigma = M_2 - M_1^2$ bestimmt werden. Der Schattenfaktor wird anschließend über eine Chebyshev Ungleichung ermittelt:

$$p(z_k) = \frac{\sigma^2}{\sigma^2 + (z_k - \mu)^2} \geq Pr(z_k < z_s) \tag{2.31}$$

Wobei z_k der transformierte Tiefenwert aus der Betrachtersicht und z ein Tiefenwert ist. Der Vorteil von VSM besteht darin, dass die Momente M_1 und M_2 vorgefiltert

werden können. Dadurch kann ein weicher Schatten mit deutlich weniger Texturzugriffen, z.B. durch Summed-Area Tables [52], erfolgen. Allerdings kann bei einer hohen Varianz, wie z.B. an Stellen mit mehreren Schattenspendern, ein Light-Leaking entstehen [28]. Diese können jedoch mit mehreren VSMs für verschiedene Tiefenschichten reduziert werden [54].

Exponential Shadow Maps (ESM) [8, 86] benutzt eine Exponentialfunktion als Basisfunktion, um die Schattentestfunktion so zu modifizieren, dass die Anteile für den Schattenspender und Schattenempfänger separierbar sind. Dabei wird angenommen, dass $z_s - z_k < 0$ ist. Dieses Vorgehen ermöglicht eine Vorfilterung der Shadow Map. Die Filterfunktion ist gegeben durch [28]:

$$f(t, z_k) = e^{cz_k} \sum_{t_i \in K} k(t_i - t) e^{-c \cdot z_s(t_i)} \tag{2.32}$$

Mit einer Nachbarschaft t um z_k, einem Filterkern k und einem Faktor c. Der Vorteil gegenüber VSM besteht darin, dass ein Light-Leaking bei mehreren Schattenspendern nicht mehr auftritt. Allerdings kann es trotzdem zu Light-Leaking kommen, wenn die Bedingung $z_s - z_k < 0$ nicht erfüllt ist (siehe [28]).

2.3.2. Ambient Occlusion

Um eine indirekte Abschattung für ein lokales Beleuchtungsmodell zu realisieren, wurde Ambient Occlusion entwickelt. Die Idee besteht darin, den konstanten ambienten Lichtanteil des lokalen Beleuchtungsmodells dynamisch an die Geometrie bzw. die Umgebung anzupassen. Beispielsweise ist die ambiente Beleuchtung an den Stellen dunkler, die in einer Vertiefung liegen. Oder anders ausgedrückt, an den Stellen bei denen eine Hemisphäre um einen Oberflächenpunkt teilweise verdeckt ist.

Nach Loos und Sloan [55] ist Ambient Occlusion ein Spezialfall von Obscurance. Obscurance an einem Oberflächenpunkt p ist wie folgt definiert [55] :

$$A(p) = \frac{1}{\pi} \int_{\Omega} s(d(p, \omega)) cos\theta d\omega$$

Wobei Ω die Hemisphäre, s die Skalierungsfunktion, d die Distanz zum ersten Schnittpunkt von p in Richtung ω und θ der Winkel zwischen der Normale von p und der Richtung ω ist. Bei Ambient Occlusion ist die Skalierungsfunktion Null für alle Punkte kleiner ∞, andernfalls Eins [55].

Verfahren

In der Literatur gibt es eine Vielzahl von Verfahren für die Berechnung von Ambient Occlusion. Im Folgenden werden nur die in dieser Arbeit verwendeten Verfahren vor-

gestellt. Für eine vollständige Diskussion anderer Verfahren wird auf Ritschel et al. [81] verwiesen.

Ambient Occlusion kann im Screen Space mit Hilfe einer Tiefenkarte bestimmt werden. Die Idee besteht darin, eine Kugel um einen Oberflächenpunkt zu legen und mit Hilfe der Tiefenkarte zu überprüfen, wieviel Prozent der Kugel verdeckt ist [59]. Dazu werden Stichproben innerhalb der Kugel erzeugt, auf die Tiefenkarte zugegriffen und der Tiefenwert auf eine Verdeckung überprüft.

Bavoil et al. [11] führen Horizontal Based Ambient Occlusion ein, bei dem Ambient Occlusion über den Öffnungswinkel des sichtbaren Horizontes bestimmt wird. Um diesen Öffnungswinkel zu bestimmen, wird die Tiefenkarte in verschiedenen Richtungen mittels einen Ray Marching abgetastet. Dieses Verfahren liefert eine genaue Abschätzung des Ambient Occlusions da ein minimaler Horizont-Öffnungswinkel der umliegenden Geometrie gesucht wird. Allerdings erfordert das Ray Marching eine hohe Anzahl an Texturzugriffen.

Volumetric Obscurance [55] bestimmt Ambient Occlusion mit Hilfe eines Line Samplings. Jedem Line Sample wird dabei ein Volumenanteil einer Einheitskugel zugewiesen. Beim Sichtbarkeitstest wird nun überprüft, wieviel einer Linie sichtbar ist. Damit kann auf den verdeckten Volumenanteil der Kugel geschlossen werden. Der Vorteil dieser Vorgehensweise besteht darin, dass mit wenigen Line Samples bereits ein kontinuierlicher Grauwerteübergang erreicht wird. Des Weiteren können verdeckte Line Samples, durch eine Paarung der Abtastpunkte, mit wenigen Artefakten invalidiert werden (siehe [69]).

2.4. Infrarotbildgenerierung

Die Infrarotbildgenerierung beschreibt eine Bildgenerierung bei dem der Wellenlängenbereich für die Beleuchtungsrechnung in das Infrarotspektrum gesetzt wird. Dabei wird zwischen kurzwelligen ($1.4 - 3\mu m$), mittelwelligen ($3 - 8\mu m$) und langwelligen ($8 - 15\mu m$) Infrarot unterschieden [96]. Diese Arbeit befasst sich ausschließlich mit dem langwelligen Infrarotbereich.

Im Gegensatz zum visuellen Spektrum, bei dem die reflektierte Strahlung dominiert, ist die emittierte Strahlung im langwelligen Infrarotspektrum entscheidend. Nahezu jede Oberfläche emittiert eine Infrarotstrahlung [77]. Hingegen spielt die reflektierte Strahlung nur eine untergeordnete Rolle [12]. Eine Besonderheit bei der Infrarotbildgenerierung besteht darin, dass die Eigenstrahlung und die Strahlungsdämpfung der Atmosphäre, z.B. durch Gase, berücksichtigt werden muss. Die Eigenstrahlung wird mit einem Atmosphärenmodell, wie z.B. MODTRAN [6], berechnet und bei der Wärmebilanzrechnung berücksichtigt. Die Dämpfung hingegen, muss bei der Simulation eines Sensors berücksichtigt werden [96]. In dieser Arbeit wird jedoch der Sensor nicht explizit simuliert und sondern lediglich der emittierte Strahlungsanteil berechnet. Daher wird die Strahlungsdämpfung der Atmosphäre vernachlässigt.

Eine weitere Besonderheit gegenüber der visuellen Bildgenerierung besteht darin, dass nahezu jede Oberfläche ein Wärmespeicher ist. Das heißt, dass sich die Strahlungswerte mit der Zeit verändern. Diese Eigenschaft wird insbesondere bei Schatten sichtbar. Abhängig von der Oberfläche kann ein Schatten noch über längere Zeit sichtbar sein (thermischer Schatten).

Um diese Besonderheiten in der Infrarotbildgenerierung zu berücksichtigen, wird die Rendering-Gleichung (Gleichung 2.26) an eine vierdimensionale Wärmegleichung (Gleichung 2.7) gekoppelt. Mit Hilfe der Wärmegleichung kann die Oberflächentemperatur berechnet werden. Aus diesen Oberflächentemperaturen wird, anhand des wellenlängenabhängigen Emissionsvermögens, die emittierte Strahlung abgeleitet. Des Weiteren beeinflusst die Oberflächenbeschaffenheit und die Stoffart (Metall, Nichtleiter) das Emissionsvermögen [77]. Diese Eigenschaften sind allerdings bei einer Simulation nicht für alle Oberflächen bekannt. Bei der Infrarotbildgenerierung wird daher oft vom idealisierten Fall eines schwarzen Körpers ausgegangen. Die emittierte Strahlungsintensität kann dann mit Hilfe des Planck'schen Strahlungsgesetzes bestimmt werden [77]:

$$e_{\lambda,S}(T,\lambda) = \frac{c_1}{\lambda^5 \left[exp(\frac{c_2}{\lambda T}) - 1\right]} \tag{2.33}$$

$$c_1 = 2\pi c^2 h \tag{2.34}$$

$$c_2 = \frac{ch}{k} \tag{2.35}$$

Mit der Lichtgeschwindigkeit c, der Boltzmann'schen Konstante k, dem Planck'schen Wirkungsquantum h, sowie der Wellenlänge λ und der Temperatur T. Um verschiedene Oberflächenarten zu unterstützen, wird ein grauer Strahler definiert, dessen spektrale Intensität proportional zum schwarzen Strahler ist. Damit kann die emittierte Strahlung über einen Emissionsgrad ϵ einer Oberfläche angepasst werden [77]. Die emittierte Strahlung ergibt sich somit wie folgt:

$$L_e = \epsilon \cdot e_{\lambda,S}(T,\lambda) \tag{2.36}$$

Als Alternative zur emittierten Strahlung, können die Oberflächentemperaturen durch eine Falschfarbendarstellung visualisiert werden. Dabei werden die Temperaturen zwischen einer gewählten Minimal- und Maximaltemperatur linear interpoliert und eine Farbe mit Hilfe einer Lookup-Tabelle zugewiesen. In dieser Arbeit wird die Falschfarbendarstellung verwendet, um Unterschiede in den Oberflächentemperaturen zu verdeutlichen.

2.5. OpenSceneGraph

Im Rahmen dieser Arbeit wird OpenSceneGraph (OSG) für die Implementierung eingesetzt. OSG ist eine Bibliothek für einen, auf OpenGL basierenden, Szenengraphen.

Ein Szenengraph ist ein gerichteter azyklischer Graph, der sich aus unterschiedlichen Knotenarten zusammensetzt. Beispielsweise verwaltet ein Geode-Knoten Geometrieinformationen. Ein Matrix-Transform Knoten beschreibt dagegen eine Matrix-Transformation die auf die Geometrie der Kindknoten angewandt wird. Jeder Knoten kann ein StateSet besitzen, der OpenGL Zustände verwaltet [70].

2.5.1. Traversierung des Szenengraphen

Für die Darstellung wird der Szenengraph von der Wurzel aus durchlaufen (traversiert) und OpenGL-Zustände sowie Render-Befehle in eine oder mehrere Listen, den sogenannten RenderBins, gesammelt [70]. Die Render-Befehle werden anschließend nach Zuständen oder der Tiefe sortiert und ausgeführt.

Die Traversierung bei OpenSceneGraph erfolgt durch mehrere Node-Visitor. Zuerst durchläuft ein Update-Visitor den Szenengraphen und aktualisiert die Transformationsknoten und sonstige Zustände. Im Anschluss erfolgt die Traversierung durch einen Cull-Visitor, der die OpenGL-Zustände und Render-Befehle in RenderBins aufzeichnet [70].

2.5.2. osgShadow

Neben der Szenengraphen-Bibliothek stellt OSG zusätzlich Node-Kits zur Verfügung. Diese erweitern den Szenengraphen um weitere Funktionen. Eines dieser Node-Kits ist die osgShadow Bibliothek, die eine Schattendarstellung ermöglicht. osgShadow stellt, neben regulären Shadow Mapping, auch Verfahren bereit, die ein Frustum Fitting durchführen, wie z.B. Cascaded Shadow Mapping. In dieser Arbeit wird osgShadow als Basis für die Implementierung eingesetzt und um Algorithmen für weiche Schatten erweitert.

2.6. Numerische Stabilität

Bei der Analyse von numerischen Näherungen wird zwischen einem Vorwärts- und einem Rückwärtsfehler unterschieden (vgl. [37, 41]). Gesucht ist die Lösung y einer Funktion $y = f(x)$, die mittels einer Näherung $\hat{y} = \hat{f}(x)$ bestimmt wird. Der Vorwärtsfehler gibt an, inwiefern sich das numerisch berechnete Ergebnis vom wahren Ergebnis unterscheidet. Ein Algorithmus gilt als vorwärts stabil wenn gilt [37]:

$$\left| \frac{\hat{f}(x) - f(x)}{f(x)} \right| \leq C \cdot eps \tag{2.37}$$

Mit einer Konstanten C. Diese Konstante ist problemabhängig zu wählen.

Bei der Problemstellung in dieser Arbeit ist das wahre Ergebnis durch eine analytische Lösung der Fourier-Reihe nicht lösbar. Dadurch kann der Vorwärtsfehler nicht ermittelt werden. Stattdessen wird der Rückwärtsfehler betrachtet. Der Rückwärtsfehler definiert eine Störung Δx einer Eingabevariable x, die so klein ist, dass sie keine Auswirkung auf das numerische Ergebnis besitzt [37]:

$$\hat{y} = \hat{f}(x + \Delta x) \tag{2.38}$$

Ein Algorithmus wird rückwärts stabil genannt, wenn gilt [37]:

$$\left| \frac{\Delta x}{x} \right| \leq C \cdot eps \tag{2.39}$$

Da die Kriterien für eine Vorwärts- und Rückwärtsstabilität oft schwer nachweisbar sind, können diese verallgemeinert und zu einem kombinierten Fehler zusammengefasst werden [41]. Dieser Fehler gibt an, dass sich die Lösung der Näherung nur durch ein kleines Δy unterscheidet, wenn die Eingabevariable um Δx variiert wird:

$$\hat{y} + \Delta y = \hat{f}(x + \Delta x) \tag{2.40}$$

Eine Näherung gilt als numerisch stabil, wenn Δy und Δx problembezogen ausreichend klein sind [41]. Im Rahmen dieser Untersuchung wird gefordert, dass Δy und Δx sich jeweils nur um einen Faktor 10 von der Maschinengenauigkeit unterscheiden.

3. Entwicklung eines echtzeitfähigen thermischen Modells

Dieses Kapitel widmet sich der Entwicklung eines echtzeitfähigen thermischen Modells. Die Idee besteht darin, das thermische Verhalten einer Oberfläche durch einfache mathematische Funktionen zu realisieren. Um das thermische Verhalten einer Oberfläche zu messen, wurde ein Experiment mit einer Infrarot-Kamera durchgeführt. Dieses Experiment wird anschließend über eine instationäre Wärmeleitung nachgestellt und durch eine numerische Simulation verifiziert. Ausgehend von dieser instationären Wärmeleitung wird ein echtzeitfähiges thermisches Modell hergeleitet und eine Fehleranalyse durchgeführt.

3.1. Versuchsaufbau

Um das thermische Verhalten von Oberflächen bestimmen zu können, wurde eine Messung mit einer Infrarot-Kamera durchgeführt. Im Folgenden wird der Versuchsaufbau beschrieben. Abbildung 3.1 zeigt den Versuchsaufbau.

Abbildung 3.1.: Versuchsaufbau für die Messung von thermischen Schatten. Fotos aus [71].

Das Ziel der Aufnahmen war es, den thermischen Schatten eines schwarzen Styroporblocks aufzunehmen. Der Styroporblock stand auf einer Asphaltoberfläche. Es wurde bewusst ein schwarzer Styroporblock ausgewählt, um den Wärmetransfer sowie die Reflexionen der Sonne zu minimieren. Da die genaue Zusammensetzung der Asphaltoberfläche nicht bekannt war, wurde angenommen, dass die Oberfläche aus einer Asphaltschicht, gefolgt von einer Kiesschicht und einer trockenen Erdschicht bestand.

Die Infrarot-Aufnahmen wurden im Wellenlängenbereich von 8-12 μm mit einer CEDIP JADE UC33 Ex Kamera aufgenommen. Die Kamera besitzt einen Öffnungswinkel von 14° und eine Brennweite von 16 mm [16]. Um eine genaue Messung zu gewährleisten, wurde die Kamera mit einem Schwarzkörperstrahler kalibriert. Für die Aufnahmen wurde ein sonniger Tag im August mit einigen Schleierwolken gewählt. Es wurde eine Messreihe mit jeweils einer Minute Zeitabstand, beginnend um 10 Uhr morgens, durchgeführt.

3.2. Auswertung der Messung

Die Messung wurde in zwei Aspekten ausgewertet. Zum einen, durch eine Beurteilung der visuellen Ergebnisse und zum anderen, durch eine Ermittlung des Temperaturverhaltens der Oberfläche. Letzteres stellt die Basis für die nachfolgende Herleitung des thermischen Modells dar.

3.2.1. Visuelle Ergebnisse

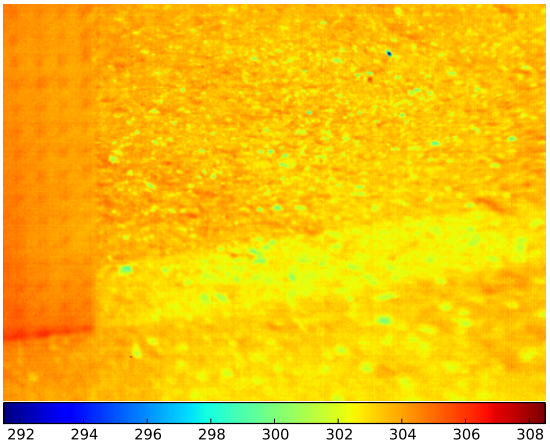
Abbildung 3.2 zeigt Beispielbilder der Messreihe. Im Bild 3.2a wurde der Styroporblock zwei Minuten zuvor aufgestellt. Die Bilder 3.2b und 3.2c zeigen den thermischen Schatten nach 10 bzw. 40 Minuten. In dieser Bildsequenz ist deutlich zu sehen, wie die Temperaturdifferenz zwischen einer Sonnen- und Schattenfläche zunimmt.

Die Eigenschaften von thermischen Schatten werden in Abbildung 3.3 verdeutlicht. Die Oberflächentemperatur an Punkten, die sich vor kurzer Zeit noch in der Sonne befunden haben, nimmt schnell ab. Dies wird durch den schmalen Übergang des roten, gelben und grünen Bereichs im Bild deutlich. Das schnelle Abklingverhalten lässt sich dadurch begründen, dass nur die Asphaltschicht eine Rolle spielt. Diese besitzt eine geringe Wärmeleitfähigkeit. Sobald tiefere Schichten eine Rolle spielen, sinkt die Temperatur langsamer. Dies wird im Bild durch die breiten blauen und cyanfarbigen Bereiche dargestellt. In diesen Bereichen sind die Auswirkungen eines Temperaturaustauschs mit den tieferen Schichten sichtbar.

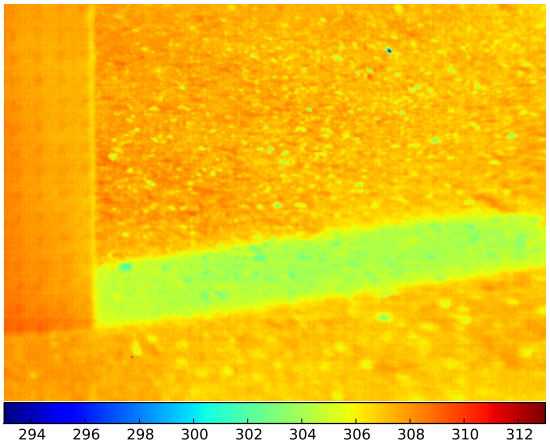
In Bereichen, die vom Schatten in die Sonne gelangen, ist das Verhalten entgegengesetzt. Die Temperaturen steigen schnell an, wodurch die blauen und cyanfarbigen Bereiche schmal sind. Allerdings sind die grünen und gelben Bereiche nun größer, da die tieferliegenden Schichten wieder aufgeheizt werden.

3.2.2. Temperaturverhalten

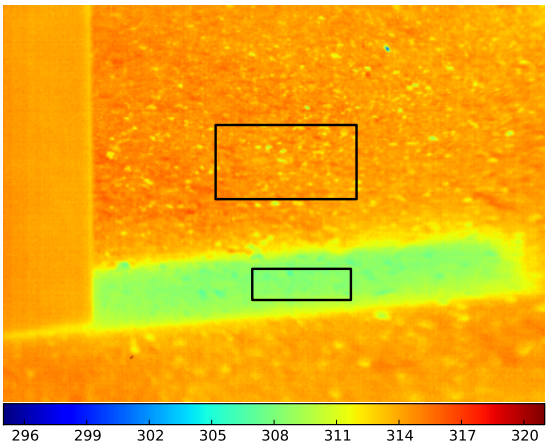
Bevor die Ergebnisse der Messung ausgewertet werden konnten, musste zuerst die Kamera, mit Hilfe eines Schwarzkörperstrahlers, kalibriert werden. Dazu wurde eine Aufnahme bei konstanter Temperatur $T_0 = 293.15$ K sowie bei konstanter Temperatur



(a) Aufnahme nach zwei Minuten.



(b) Aufnahme nach 10 Minuten.



(c) Aufnahme nach 40 Minuten. Die Rechtecke zeigen die Sonnen- und Schattenfläche für die mittlere Temperaturbestimmung an.

Abbildung 3.2.: Falschfarbenbilder der Messung (Temperaturen in Kelvin).

$T_1 = 303.15$ K gemacht. Die Kamera zeichnet die Aufnahmen in 14 Bit Graustufen auf [16]. Für die Umrechnung der Grauwerte in Temperaturen wurden die Aufnahmen der Kalibrierung genutzt. Dazu werden die mittleren Grauwerte G_0 und G_1 für T_0 bzw. T_1 bestimmt und die Differenz berechnet. Für die Messung betrug die Differenz 173.1 Grauwerte, wodurch eine Temperaturauflösung von 0.06 K erreicht wird [71]. Der Grauwert G kann dann nach Schätz [71] wie folgt in eine Temperatur T umgerechnet werden:

$$T = T_0 + (T_0 - T_1)\frac{G - G_0}{G_1 - G_0} \tag{3.1}$$

Für die Auswertung der Messung wurde ein Zeitraum von 40 Minuten, beginnend um 10:12 Uhr, betrachtet. Da die Oberfläche inhomogen ist, wurde die Temperatur über

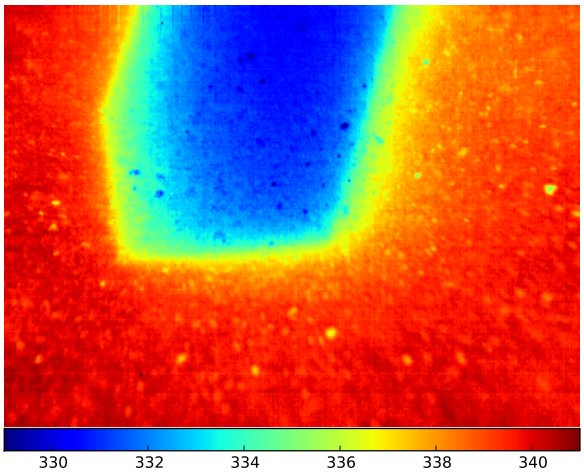


Abbildung 3.3.: Detailaufnahme eines thermischen Schattens um 13 Uhr mittags.

eine Fläche der Aufnahme gemittelt. Die gewählten Flächen werden in Abbildung 3.2c dargestellt.

Die Temperatur der Sonnen- und Schattenfläche wird in Abbildung 3.4 dargestellt. Durch die Sonneneinstrahlung stieg die Temperatur innerhalb von 40 Minuten um ca. 10 K an. Die Schwankungen in den Temperaturen werden auf eine leichte Bewölkung zurückgeführt [71].

Um das Abklingverhalten der Oberfläche zu bestimmen, wird die Differenz der Temperaturen der Schatten- und Sonnenfläche gebildet (Abbildung 3.5). Im Messzeitraum nimmt die Temperatur im Schatten um ca. 5.5 K, gegenüber einer Fläche in der Sonne, ab.

3.3. Numerische Simulation

Ausgehend von der thermischen Messung wird versucht, dass gemessene Temperaturverhalten in einer numerischen Simulation nachzustellen. Dazu wird der Versuchsaufbau mit RadTherm [93] nachgebildet.

Um eine numerische Simulation durchzuführen, muss die Schichtung der Oberfläche bekannt sein. Bei unserer Messung war sie jedoch nicht genau bekannt. Es wurde daher angenommen, dass die Oberfläche aus einer 5 cm breiten Asphaltschicht, gefolgt von einer 20 cm breiten Kiesschicht und einer 75 cm breiten, trockenen Erdschicht bestand. Allerdings ist die Dicke der trockenen Erdschicht zu vernachlässigen, da die Wärmediffusion innerhalb des Messzeitraums zu gering war um die Erdschicht zu erreichen. Eine Schicht wurde jeweils über mehrere Teilschichten repräsentiert, deren Dicke mit ansteigender Tiefe verdoppelt wurde. Die oberste Teilschicht hatte eine Dicke von 1 mm.

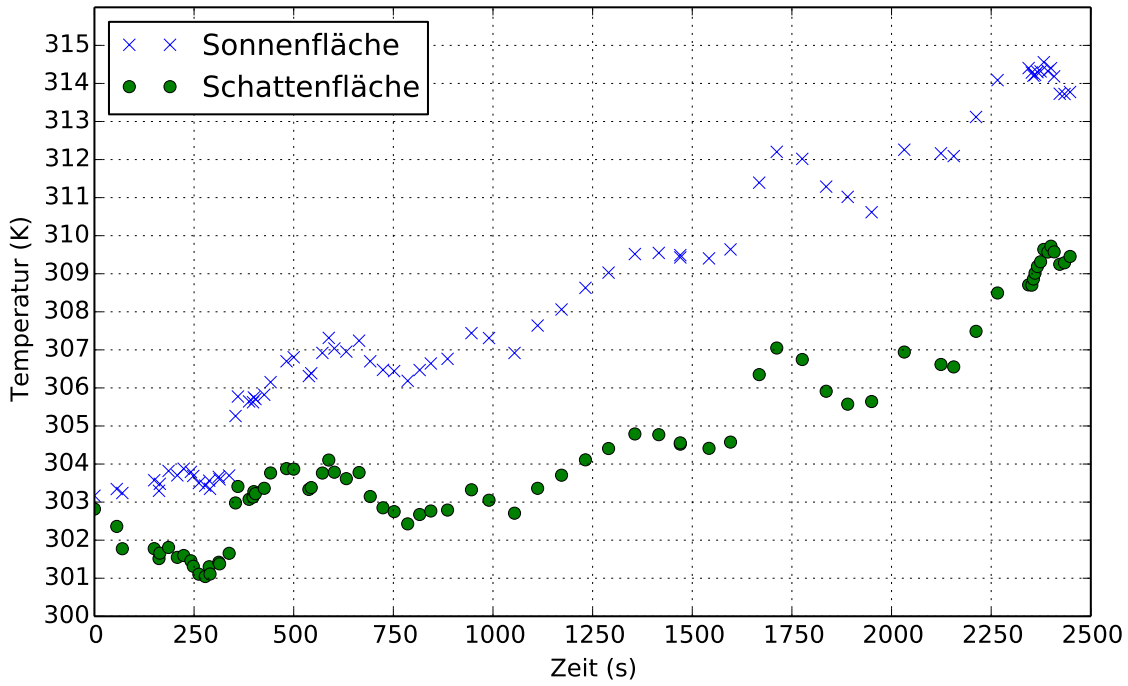


Abbildung 3.4.: Temperatur der Sonnen- und Schattenfläche der Messung.

Da das Material der obersten Schicht entscheidend für die Oberflächentemperatur ist, muss dieses genau nachgebildet werden. Bei der Messung bestand die Asphaltschicht aus einer inhomogenen Mischung von Kieselsteinen und Bitumen. Um dieses Material nachzubilden, wurde die Albedo der Oberfläche gemessen, indem ein Foto der Oberfläche zusammen mit einer weißen und schwarzen Fläche aufgenommen wurde (Abbildung 3.6).

Für die weiße Fläche wird eine Albedo von 0.95 und für die schwarze Fläche eine Albedo von 0.05 angenommen. Mit diesem Referenzflächen wird die mittlere Albedo der Oberfläche auf 0.4 geschätzt [71].

	Wärmeleitfähigkeit λ $[\frac{W}{mK}]$	Dichte ρ $[\frac{kg}{m^3}]$	Wärmekapazität c $[\frac{J}{kgK}]$
Asphalt	2.0	2700	774
Kies	2.2	2800	750
Trockene Erde	1	1500	1840

Tabelle 3.1.: Verwendete Stoffwerte für die numerische Simulation mit RadTherm.

Die numerische Simulation erfolgt mit RadTherm. In Tabelle 3.1 werden die verwendeten Stoffwerte aus der RadTherm Materialdatenbank dargestellt. Die Emissivität der Asphaltschicht wird ebenfalls aus der RadTherm Materialdatenbank entnommen und auf 0.93 festgelegt. Um die Effekte der anfänglichen Oberflächeneigenschaften zu minimieren, wird die Simulation 24 Stunden vor dem eigentlichen Messzeitraum

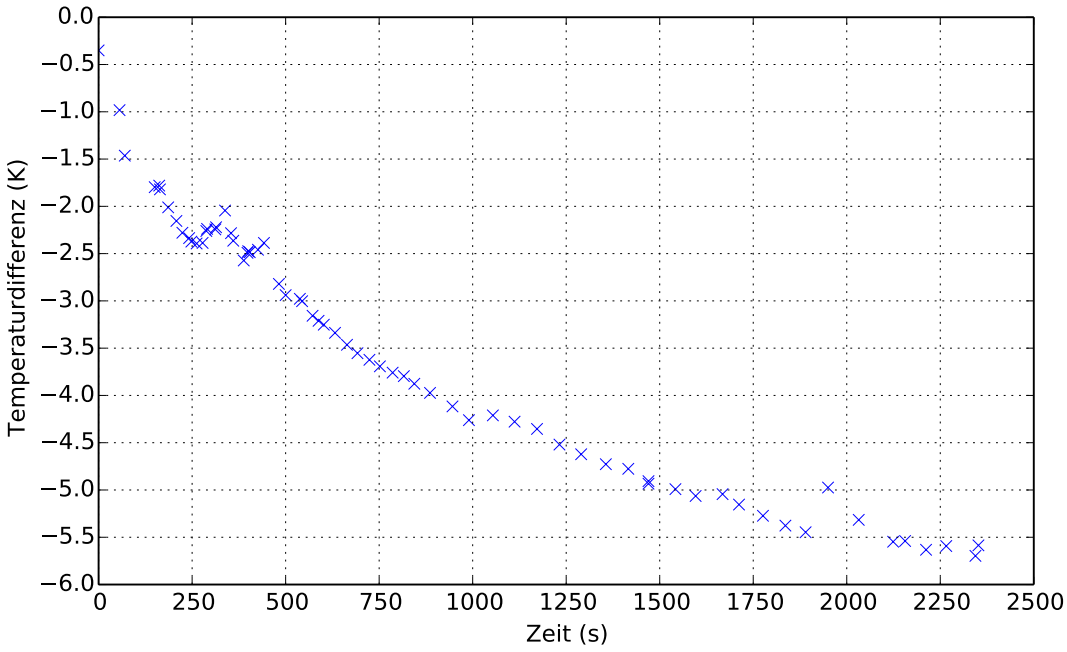


Abbildung 3.5.: Temperaturdifferenz der Sonnen- und Schattenfläche der Messung.

gestartet.

Das thermische Verhalten der Oberfläche kann durch einen Vergleich der Temperaturdifferenz einer Sonnen- und Schattenfläche charakterisiert werden. Da die Asphaltschicht inhomogen ist, wird die mittlere Temperatur in zwei Bereichen gemessen. Beim Vergleich mit den Temperaturen aus der Messung mit der Infrarotkamera muss jedoch beachtet werden, dass zusätzlich die reflektierte Strahlung der Oberfläche erfasst wurde. Dieser Beitrag ist gerade für Sonnenflächen signifikant. Um diesen Anteil zu berücksichtigen, wurde die Irradianz der Sonne im Wellenlängenbereich 8-12 μm mit Hilfe von MODTRAN [6], für die Wetterbedingungen der Messung, berechnet und die Temperaturwerte korrigiert.

Das Simulationsergebnis wird im Abbildung 3.7 mit dem Messergebnis verglichen. Das simulierte Verhalten der Oberfläche stimmt gut mit dem Messergebnis überein. Nach etwa 600 Sekunden unterscheiden sich die Temperaturdifferenzen um ca. 0.5 K. Dies ist auf die Unsicherheiten in der Messung zurückzuführen.

3.4. Echtzeitfähiges thermisches Modell

In diesem Kapitel wird ein thermisches Modell vorgestellt, welches das thermische Verhalten von Oberflächen mit Hilfe einfacher mathematischer Funktionen approximiert. Für die Herleitung wird von einer unendlich ausgedehnten ebenen Platte ausgegangen. Wie in den Grundlagen (Kapitel 2.1.4.2) beschrieben, wird das thermische Verhalten

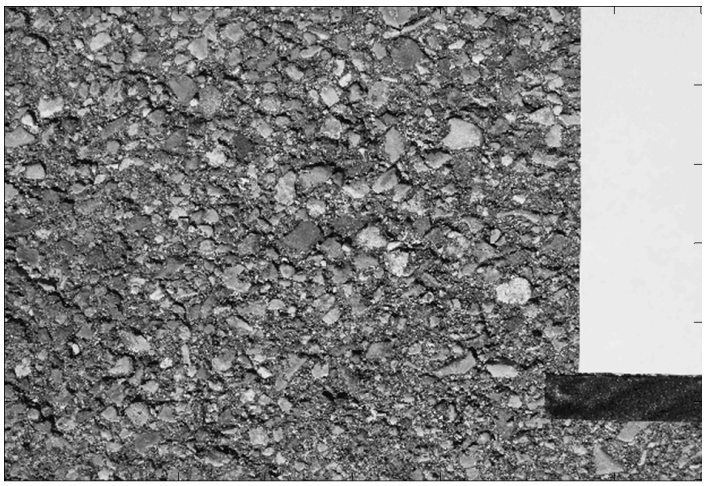


Abbildung 3.6.: Bestimmung des Albedo der Oberfläche mit Hilfe einer weißen und schwarzen Referenzfläche. Foto aus [71].

einer unendlich ausgedehnten ebenen Platte mit einer Fourier-Differenzialgleichung beschrieben, dessen Lösung eine Fourier-Reihe ist. Diese Lösung kann, basierend auf der Zeit, in zwei Teillösungen unterteilt werden: einer Kurzzeit- und einer Langzeitlösung [77, 10]. Nach Polifke und Kopitz [77] kann mit steigender Ordnungsnummer k der Fourier-Reihe beobachtet werden, dass die Eigenwerte große Werte annehmen. Dadurch wird der Exponent $-\delta_k^2 \cdot Fo$ der Exponentialfunktion sehr groß, wodurch ihr Wert gegen Null konvergiert. Das gleiche gilt für große Fourier-Zahlen. Für eine Langzeitlösung sind daher alle Reihenglieder mit $k > 1$ der Fourier-Reihe vernachlässigbar [77]. Das thermische Verhalten einer Oberfläche kann mit dem ersten Reihenglied der Fourier-Reihe angenähert werden:

$$\theta(\xi = 1, Fo) \approx C_1 cos(\delta_1) e^{-\delta_1^2 Fo} \tag{3.2}$$

Wobei das anfängliche Temperaturprofil $\theta(\xi, 0)$ der Oberfläche bekannt sein muss, da es in den Koeffizienten C_1 einfließt.

Für kurze Zeiten hingegen, konvergiert die Fourier-Reihe langsam. Daher muss eine große Anzahl an Reihengliedern berechnet werden. In der Literatur wird für kurze Zeiten das Modell der halbunendlichen Körper vorgeschlagen ([77, 10, 57]). Auf die Herleitung der halbunendlichen Körper wird hier verzichtet und stattdessen auf Polifke und Kopitz [77] verwiesen. Die dimensionslose Darstellung eines halbunendlichen Körpers lautet nach [57]:

$$\theta(\xi = 1, Fo) \approx e^{Fo \cdot Bi^2} \cdot erfc(\sqrt{Fo} \cdot Bi) \tag{3.3}$$

Wobei $erfc(x) = 1 - erf(x)$ die konjugierte Fehlerfunktion ist:

$$erf(x) = \frac{2}{\sqrt{\pi}} \int_x^\infty e^{-k} dk \tag{3.4}$$

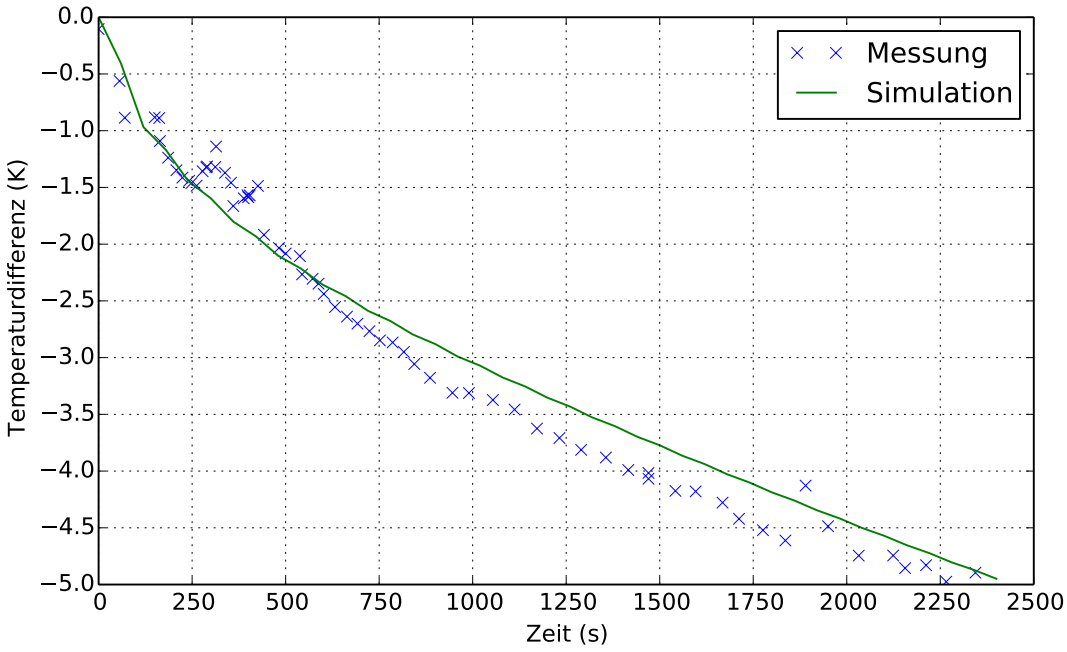


Abbildung 3.7.: Temperaturdifferenz zwischen einer Schatten- und Sonnenfläche bei der Messung und numerischen Simulation.

Die Fehlerfunktion kann nach Abramowitz und Stegun [1] mit einer absoluten Genauigkeit von $2.5 \cdot 10^{-5}$ numerisch angenähert werden, so dass es für einen Einsatz in einer Echtzeitsimulation geeignet wäre:

$$erf(x) \approx 1 - (a_1 t + a_2 t^2 + a_3 t^3) e^{-x^2} \tag{3.5}$$

Mit $a_1 = 0.3480242$, $a_2 = -0.0958795$, $a_3 = 0.748556$ und $t = \frac{1}{1+0.47047x}$.

Für die Lösung der Wärmeleitungsgleichung des halbunendlichen Körpers wird angenommen, dass eine konstante Temperaturverteilung in der Oberfläche herrscht [77]. Diese Tatsache wird auch in Gleichung 3.3 deutlich, da dort das anfängliche Temperaturprofil $\theta(\xi, 0)$ nicht einfließt. Diese Näherung ist daher nur für ein konstantes Temperaturprofil $\theta(\xi, 0) = const$ oder für große Biot-Zahlen sinnvoll (vgl. Abschnitt 3.5.5) . Bei großen Biot-Zahlen ist der Wärmeleitwiderstand im Verhältnis zum Wärmeübergangswiderstand groß. Dadurch werden die Temperaturen in einem Festkörper nur sehr langsam ausgeglichen und das anfängliche Temperaturprofil spielt eine untergeordnete Rolle.

Um allerdings auch nicht-konstante, anfängliche Temperaturprofile zu unterstützen, besteht unsere Idee darin, die Kurzzeitlösung ebenfalls über eine Exponentialfunktion anzunähern. Die Parameter dieser Exponentialfunktion werden durch ein Optimierungsverfahren gegenüber einer Referenzlösung angepasst. Unser thermisches Modell kann wie folgt definiert werden:

$$\theta(\xi = 1, Fo) \approx \beta_0 e^{-\tau_0 Fo} + \beta_1 e^{-\tau_1 Fo} \tag{3.6}$$

Die Zeitkonstanten τ_0 und τ_1, sowie die Gewichte β_0 und β_1 werden über eine Kurvenanpassung an eine Referenzlösung ermittelt.

Vollmer [95] sowie Vollmer und Möllmann [96] haben gezeigt, dass eine Reihe von Exponentialfunktionen das Temperaturverhalten von Oberflächen beschreiben kann und passen die Reihe von Exponentialfunktionen mit Hilfe eines Optimierungsverfahrens an Messwerte an. In Gegensatz dazu, grenzen wir jedoch stärker zwischen einer Kurzzeit- und Langzeitlösung ab und ermitteln die Zeitkonstante für die Langzeitlösung direkt aus den physikalischen Parametern.

3.4.1. Bestimmung der Parameter

Für die Langzeitlösung entspricht die Zeitkonstante τ_1 dem Quadrat des ersten Eigenwerts der Fourier-Reihe:

$$\tau_1 = \delta_1^2 \tag{3.7}$$

Die Eigenwerte δ_k werden über eine transzendente Gleichung bestimmt (siehe Kapitel 2.24). Da eine analytische Lösung einer transzendenten Gleichung nicht möglich ist, werden stattdessen numerisch die Schnittpunkte der Kurven bestimmt. Eine Numpy-Implementierung der Eigenwertbestimmung ist im Anhang A.3 zu finden.

Ebenso kann der Parameter β_1 direkt aus dem Koeffizienten der Fourier-Reihe ermittelt werden:

$$\beta_1 = C_1 cos(\delta_1) \tag{3.8}$$

Um die Zeitkonstante τ_0 sowie den Koeffizienten β_0 der Kurzzeitlösung zu bestimmen, wird eine nicht lineare Kurvenanpassung gegen eine Referenzlösung durchgeführt. Das nichtlineare Optimierungsproblem kann wie folgt definiert werden:

$$\underset{\tau_0,\beta_0}{min} \left| y - (\beta_0 e^{-\tau_0 Fo} + \beta_1 e^{-\delta_1^2 Fo}) \right|^2 \tag{3.9}$$

Wobei y die Referenzlösung ist. Als nichtlineares Optimierungsverfahren wurde der Levenberg-Marquardt Algorithmus eingesetzt (siehe [13]).

Im Gegensatz zur Parameterbestimmung, wird die Simulation mit dimensionsbehafteten Parametern durchgeführt. Allerdings muss darauf geachtet werden, dass die Zeitkonstanten nach der Kurvenanpassung dimensionslos sind. Diese müssen also vor dem Einsatz wieder dimensioniert werden. Dazu wird eine dimensionsbehaftete Zeitkonstante τ_d eingeführt:

$$\tau_d \cdot t = \tau \cdot Fo \iff \tau_d = \tau \frac{\lambda}{\rho \cdot c \cdot L^2} \tag{3.10}$$

Wobei ρ die Dichte, c die Wärmekapazität, λ die Wärmeleitfähigkeit und L die charakteristische Länge ist.

3.4.2. Erhöhung der Robustheit einer nichtlinearen Kurvenanpassung

Das vorgestellte thermische Modell besitzt sowohl lineare als auch logarithmische Parameter, die durch eine Kurvenanpassung ermittelt werden. Durch die logarithmischen Parameter muss jedoch eine nichtlineare Kurvenanpassung durchgeführt werden.

Um die Robustheit der Anpassung gegenüber schlecht gewählten Startparametern zu erhöhen und die Anzahl der Funktionsauswertungen zu verringern, schlagen Golub und Pereyra [32] eine Variablenprojektion vor. Die Idee besteht darin, die linearen Parameter c und die logarithmischen Parameter α getrennt voneinander anzupassen. Das Minimierungsproblem kann, folgend der Notation von O'Leary et al. [65], für eine Funktion $f(\alpha, c)$ wie folgt definiert werden:

$$\min_{\alpha} |y - f(\alpha, c(\alpha))|^2 \tag{3.11}$$

wobei der Parameter $c(\alpha)$ ein Funktional ist. Das Funktional ist definiert als eine lineare Kurvenanpassung der linearen Parameter c für gegebene, feste Parameter α:

$$\min_{c} |y - f(\alpha, c)|^2 \tag{3.12}$$

Eine Numpy-Implementierung der Variablenprojektion für das vorgestellte thermische Modell kann dem Anhang A.2 entnommen werden.

3.5. Fehlerabschätzung

In diesem Abschnitt wird eine Fehlerabschätzung des entwickelten thermischen Modells durchgeführt und mit Standardverfahren aus der Literatur verglichen.

3.5.1. Definition des Fehlers

In der Physik wird der Fehler oft mit Unsicherheiten in einer physikalischen Messung gleichgesetzt [92]. Nach Taylor [92] können diese Unsicherheiten beispielsweise durch Ungenauigkeiten in der Messung oder zufällige Zustände auftreten.

Für die Fehlerabschätzung in diesem Kapitel wird hingegen der Fehler als die absolute Abweichung einer Referenz- und einer Approximations-Temperaturkurve definiert. Die Referenz-Temperaturkurve wird über eine numerische Lösung der Fourier-Reihe berechnet. Hingegen wird die Approximations-Temperaturkurve über eine Approximation bestimmt. Für eine Abschätzung des Fehlers wird im Folgenden das Maximum dieser Abweichung verwendet.

3.5.2. Modellaufbau

Abbildung 3.8 zeigt den schematischen Aufbau des Modells. Es wird angenommen, dass der zu vermessende Festkörper aus einer unendlich ausgedehnten ebenen Platte der Dicke L besteht. In der Platte liegt ein anfängliches Temperaturprofil $\theta(\xi, 0)$ vor. Der Festkörper besitzt die Wärmeleitfähigkeit λ, die Wärmekapazität c und die spezifische Dichte ρ. Zwischen dem Festkörper und der Umgebung findet ein konvektiver Wärmeaustausch mit dem Wärmeübergangskoeffizienten α statt (Randbedingung 3. Art). Die Fluidtemperatur ist T_∞. Am Rand bei $\xi = 0$ herrscht eine Adiabasie (Randbedingung 2. Art). Die anfängliche Temperatur an der Oberfläche beträgt $\theta(\xi = 0, Fo = 0) = 1$ (Randbedingung 1. Art). Gesucht ist die dimensionslose Temperatur an der Oberfläche $\theta(\xi = 1, Fo)$.

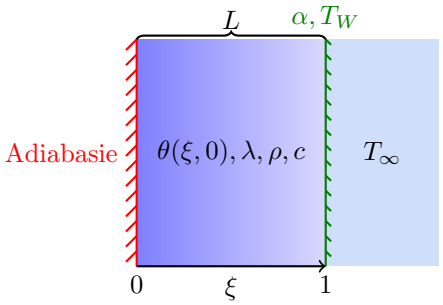


Abbildung 3.8.: Schematischer Aufbau des Festkörpers für die Fehlerabschätzung.

Als Einflussgrößen für die instationäre Wärmeübertragung in diesem Modell gelten:

- Zeit t $[s]$
- Wärmeleitfähigkeit λ $[\frac{W}{mK}]$
- Wärmekapazität c $[\frac{J}{kgK}]$
- Spezifische Dichte ρ $[\frac{kg}{m^3}]$
- Dicke der Platte L $[m]$
- Emissivität an der Oberfläche ϵ
- Fluidtemperatur T_∞ $[K]$
- Anfangstemperatur an der Oberfläche $T(r = L, t = 0) = T_W$ $[K]$

- Temperaturprofil $\theta(\xi, 0)$

Bei Betrachtung der, für die analytische Lösung notwendigen, Einflussgrößen fällt jedoch auf, dass nur die Biot- und Fourier-Zahl, das Temperaturprofil und die dimensionslose Länge einfließen. Diese Tatsache wird ausgenutzt, um die Anzahl der Veränderlichen für die Fehlerabschätzung zu reduzieren. Die dimensionslose Länge kann ebenfalls außen vorgelassen werden, da für die Infrarotbildgenerierung die Temperatur an der Oberfläche ($\xi = 1$) entscheidend ist.

Der Bereich der Variation der Kenngrößen wird in Tabelle 3.2 gezeigt. Dabei wurde darauf geachtet, möglichst praxisrelevante Bereiche zu erfassen. Beispielsweise werden, über den gewählten Bereich der Biot-Zahl, Materialien mit langsamer und schneller Wärmediffusion erfasst, wie z.B. Holz und Metalle. Der Bereich der Fourier-Zahl wurde so gewählt, dass sowohl sehr kurze Abklingphasen als auch eine Langzeit-Lösung gefordert sind. Für die Variation des Temperaturprofils wurde ein konstantes, ein lineares und ein exponentielles Profil in der Schicht gewählt.

Kenngröße	**Bereich**
Biot-Zahl	10^{-1}- 10^{1}
Fourier-Zahl	10^{-5} - 10^{3}
Temperaturprofil	konstant: $\theta(\xi, 0) = 1$ linear: $\theta(\xi, 0) = \frac{\xi}{2} + \frac{1}{2}$ exponentiell: $\theta(\xi, 0) = e^{-1+\xi}$

Tabelle 3.2.: Variation der Kenngrößen für die Fehlerabschätzung.

3.5.3. Referenzlösung

Als Referenzlösung wird eine Fourier-Reihe mit 100 Gliedern numerisch gelöst:

$$\theta(\xi = 1, Fo) \approx \sum_{k=1}^{100} C_k cos(\delta_k \xi) e^{-\delta_k^2 Fo} \tag{3.13}$$

Das dabei auftretende Integral im Koeffizienten C_k wird über eine Gauß-Kronrod Quadratur mit Hilfe der Bibliothek QUADPACK [75] berechnet.

Die bei der Referenzlösung auftretenden numerischen Fehler werden gesondert in Abschnitt 3.5.6 behandelt.

3.5.4. Anpassung zweier Exponentialfunktionen

Unsere Approximation beruht auf der Anpassung zweier Exponentialfunktionen. Für die Kurvenanpassung wird das in Kapitel 3.4.2 beschriebene Verfahren der Variablenprojektion benutzt.

In diesem Abschnitt wird außerdem verglichen, ob eine Festlegung der Parameter τ_1, β_1 über den Eigenwert δ_1 bzw. Koeffizienten C_1 eine geringere Abweichung zur Referenzkurve verursacht, als wenn die Parameter über ein nichtlineares Optimierungsverfahren angepasst werden.

3.5.4.1. Festlegung der Parameter über Eigenwerte

In diesem Vergleich werden die Parameter τ_1, β_1 mit Hilfe des ersten Eigenwerts δ_1 und des Koeffizienten C_1 bestimmt und die verbleibenden Parameter β_0 und τ_0 angepasst. Der erste Eigenwert wird über die transzendente Gleichung 2.24 und der Koeffizient C_1 über die Gleichung 2.25 bestimmt.

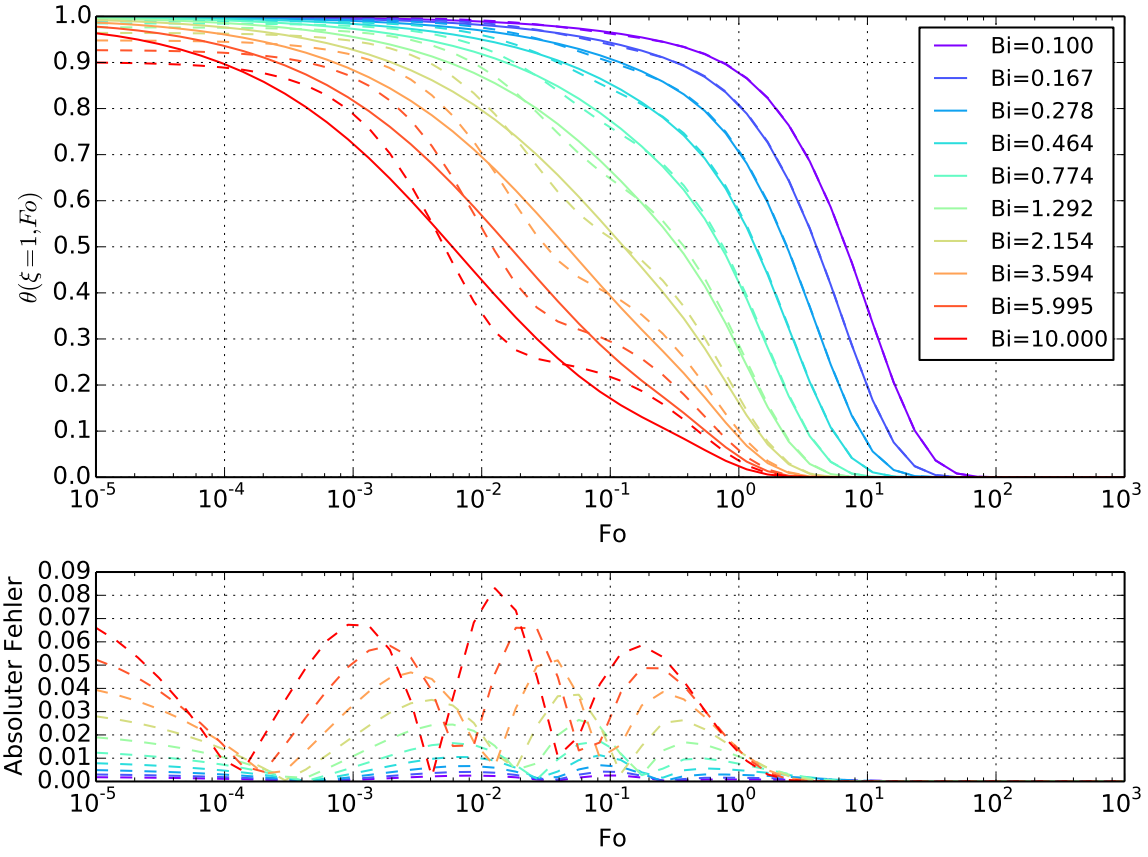


Abbildung 3.9.: Vergleich der Fourier-Reihe (durchgezogene Linie) und der Approximation mit zwei Exponentialfunktionen (gestrichelte Linie) für ein konstantes Temperaturprofil. Die Zeitkonstante τ_1 wurde mit Hilfe des ersten Eigenwerts und der Parameter β_1 über den Koeffizienten C_1 bestimmt.

Abbildung 3.9 zeigt das Ergebnis der Fehlerabschätzung für das konstante Temperaturprofil. Bei der Betrachtung der Fehlerkurve kann beobachtet werden, dass der absolute Fehler für Biot-Zahlen $Bi > 1$ und für Fourier-Zahlen $Fo < 1$ am größten ist.

Das liegt daran, dass für kleine Fourier-Zahlen die Fourier-Reihe langsam konvergiert und daher viele Reihenglieder ausgewertet werden. Diese Reihenglieder werden nun durch eine einzige Exponentialfunktion ersetzt.

Für kleine Biot-Zahlen hingegen, ist der Wärmeleitwiderstand im Festkörper gering. Dadurch wird die Temperatur im Festkörper schnell ausgeglichen. Die Ortskoordinate ξ ist dann vernachlässigbar und das thermische Verhalten kann über eine Blockkapazität $\theta(Fo, Bi) = e^{-Fo \cdot Bi}$ beschrieben werden (siehe [77]). Dieses Verhalten kann daher mit einem maximalen absoluten Fehler von 0.03 über eine Exponentialfunktion angenähert werden. Für Fourier-Zahlen $Fo > 1$ ist der maximale absolute Fehler geringer als 0.01, da die zweite Exponentialfunktion die Langzeitlösung, über das erste Reihenglied der Fourier-Reihe, annähert.

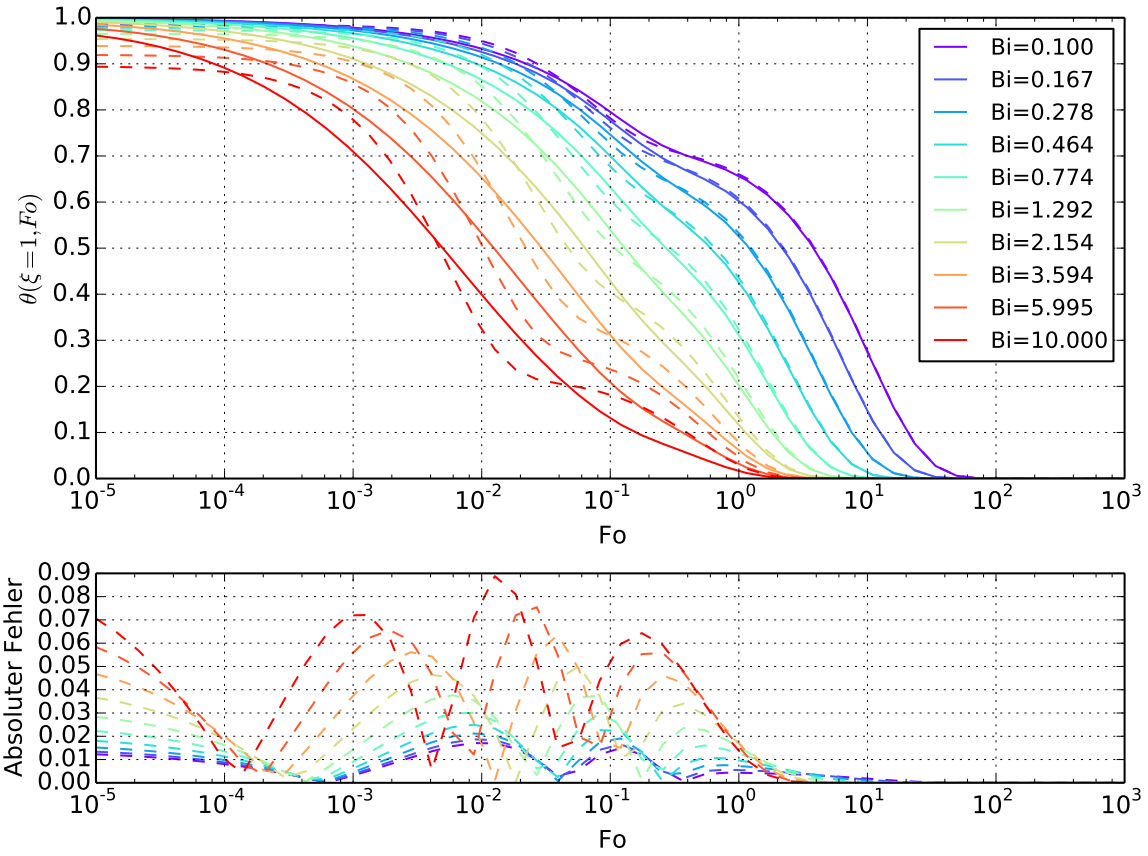


Abbildung 3.10.: Vergleich der Fourier-Reihe (durchgezogene Linie) und der Approximation mit zwei Exponentialfunktionen (gestrichelte Linie) für ein lineares Temperaturprofil. Die Zeitkonstante τ_1 wurde mit Hilfe des ersten Eigenwerts und der Parameter β_1 über den Koeffizienten C_1 bestimmt.

Abbildung 3.10 zeigt die Fehlerabschätzung für ein lineares Temperaturprofil. Die Auswirkungen des linearen Temperaturprofils werden dadurch deutlich, dass sich die Oberflächentemperatur schneller der Fluidtemperatur angleicht. Bereits bei einer Fourier-Zahl von $Fo = 10^{-1}$ ist die Oberflächentemperatur geringer als bei einer konstan-

ten Temperaturverteilung im Festkörper. Dieses Verhalten kann besonders bei kleinen Biot-Zahlen beobachtet werden. Bei kleinen Biot-Zahlen ist der Wärmeübergangswiderstand im Verhältnis zum Wärmeleitwiderstand groß. Das bedeutet, dass die Wärme im Festkörper schnell ausgeglichen wird, aber sich die Oberflächentemperatur nur langsam an die Fluidtemperatur angleicht. Durch den schnellen Temperaturausgleich im Festkörper nimmt die Oberflächentemperatur zunächst schneller ab, weil die Temperaturdifferenz der unteren Schichten des Festkörpers ausgeglichen wird. Nach dem die Temperaturen ausgeglichen sind, überwiegt für eine kurze Zeit der hohe Wärmeübergangswiderstand, wodurch die Oberflächentemperatur langsamer abnimmt. Dieses Verhalten kann in Abbildung 3.11, die das Temperaturprofil des Festkörpers zeigt, verdeutlicht werden. Zwischen $Fo = 10^{-1}$ und $Fo = 10^{0}$ herrscht ein Temperaturgleichgewicht, wodurch der Temperaturausgleich verlangsamt wird.

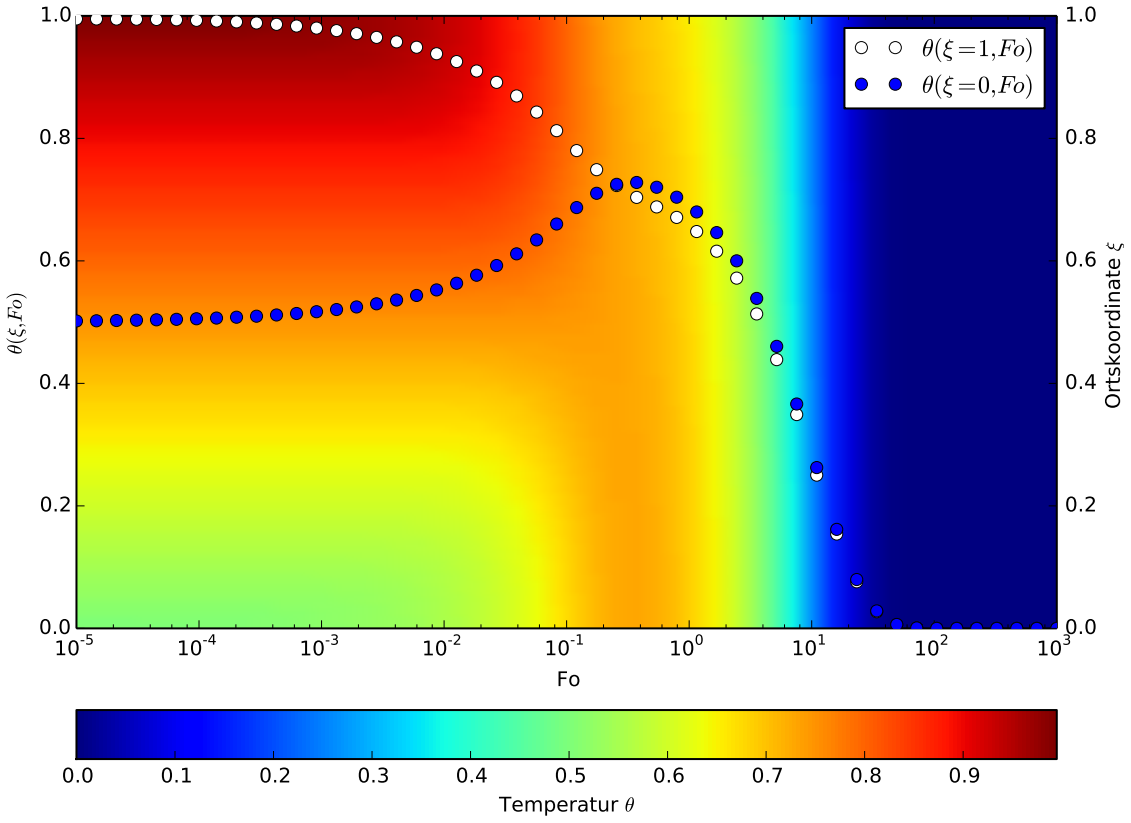


Abbildung 3.11.: Temperaturprofil des Festkörpers für $Bi = 0.1$ bei einer anfänglichen, linearen Temperaturverteilung im Festkörper. Zwischen $Fo = 10^{-1}$ und $Fo = 10^{0}$ herrscht ein Temperaturgleichgewicht, wo durch der Temperaturausgleich mit dem Fluid verlangsamt wird.

Bei großen Biot-Zahlen spielt hingegen das Temperaturprofil $\theta(\xi, 0)$ nur eine untergeordnete Rolle, da durch den hohen Wärmeleitwiderstand die Temperaturen im Festkörper nur langsam ausgeglichen werden. Ein geringer Wärmeübergangswiderstand

führt dazu, dass die Oberflächentemperatur schnell an die Fluidtemperatur angeglichen wird. Bei der Betrachtung des absoluten Fehlers kann beobachtet werden, dass für kleine Fourier-Zahlen das lineare Temperaturprofil zu einem größeren absoluten Fehler führt als ein konstantes Profil. Die Fehlerkurven an sich verhalten sich jedoch analog zum konstanten Profil. Die Fehlerabschätzung zum exponentiellen Temperaturprofil ist im Anhang A.1.1 zu finden.

3.5.4.2. Bestimmung der Parameter über Kurvenanpassung

Bei diesem Vergleich werden die Parameter β_0, β_1 sowie die Zeitkonstanten τ_0, τ_1 über eine nichtlineare Optimierung an eine Referenzlösung angepasst.

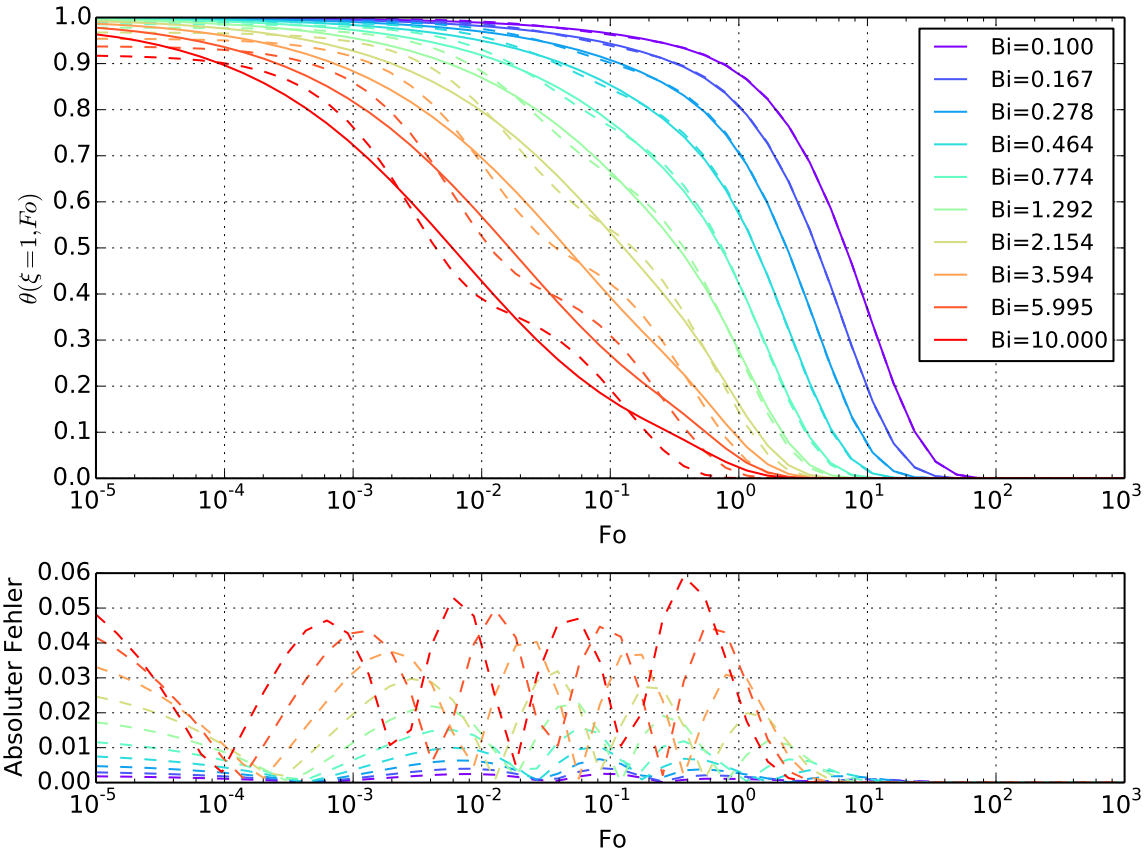


Abbildung 3.12.: Vergleich der Fourier-Reihe (durchgezogene Linie) und der Approximation mit zwei Exponentialfunktionen (gestrichelte Linie) für ein konstantes Temperaturprofil. Alle Parameter wurden durch eine Kurvenanpassung bestimmt.

Abbildung 3.12 zeigt die Fehlerabschätzung für ein konstantes Temperaturprofil. Im Vergleich zur Festlegung der Zeitkonstante τ_1 auf das Quadrat des ersten Eigenwerts und des Parameters β_1 auf $C_1 cos(\delta_1)$ fällt auf, dass ein kurzzeitiges Temperaturverhalten mit einem um 0.02 geringeren maximalen absoluten Fehler approximiert werden

kann. Jedoch ist für Fourier-Zahlen $Fo \geq 1$ der Fehler anfänglich höher. Es kann daher die Schlussfolgerung gezogen werden, dass eine Kurvenanpassung von τ_1 und β_1 einen verzögerten Übergang in die Langzeitlösung verursacht. Ein analoges Verhalten entsteht für das lineare und das exponentielle Temperaturprofil (Abbildung im Anhang A.2 und A.3).

3.5.5. Halbunendliche Körper und 1. Reihenglied der Fourier-Reihe

Im Abschnitt 3.4 wurde beschrieben, dass das thermische Verhalten einer Oberfläche durch eine Kombination von halbunendlichen Körpern und dem ersten Reihenglied der Fourier-Reihe approximiert werden kann. Es muss aber noch geklärt werden, wann von der Kurzzeitlösung auf die Langzeitlösung gewechselt werden soll. Grigull et al. [34] führt dazu eine kritische Fourier-Zahl Fo^* ein, die den Zeitpunkt für einen Wechsel, unter Berücksichtigung eines maximalen relativen Fehlerbetrags, angibt. Für einen Fehlerbetrag von 0.5%, bei einer ebenen Platte, ist die kritische Fourier-Zahl $Fo^* = 0.3$ [34]. Bei der Approximation mit halbunendlichen Körpern wird jedoch von einem konstanten Temperaturprofil ausgegangen [77]. Daher wird es zu Abweichungen kommen, wenn ein anderes Temperaturprofil vorliegt.

Abbildung 3.13 zeigt die Fehlerabschätzung für ein konstantes Temperaturprofil. Der halbunendliche Körper, in Kombination mit dem ersten Reihenglied der Fourier-Reihe, bieten eine sehr genaue Approximation des Temperaturverhaltens. Die, von Grigull et al., angegebene Fehlerschranke von 0.5% wird eingehalten.

Anders sieht das Ergebnis bei einem linearen Temperaturprofil aus (Abbildung 3.14). Für sehr kleine Fourier-Zahlen $Fo < 10^{-3}$ kann die Referenzkurve mit einem maximalen absoluten Fehler von 0.02 approximiert werden. Für Fourier-Zahlen zwischen $0.001 < Fo < 0.4$ ist der Einfluss des Temperaturprofils für kleine Biot-Zahlen am größten. Da die Temperatur im Festkörper bei kleinen Biot-Zahlen schnell ausgeglichen wird, beeinflusst das anfängliche Temperaturprofil die Oberflächentemperatur deutlich. Dadurch weicht die Kurzzeitlösung, mit einem maximalen absoluten Fehler von 0.23, deutlich von der Referenzkurve ab. Dies wird auch dadurch bestätigt, dass für große Biot-Zahlen $Bi > 3.5$ die Temperatur im Festkörper nur langsam ausgeglichen wird und somit das Temperaturprofil nur einen geringen Einfluss ausübt. Für diese Biot-Zahlen ist der halbunendliche Körper eine genaue Approximation. Es entsteht ein maximaler absoluter Fehler von 0.09. Ein analoges Verhalten kann bei einem exponentiellen Temperaturprofil beobachtet werden, jedoch ist der maximale absolute Fehler mit 0.33 noch größer (Abbildung im Anhang A.4).

3.5.6. Numerische Stabilität

Im Folgenden wird die numerische Stabilität für die Referenzlösung sowie für die Approximation mit zwei Exponentialfunktionen mittels des kombinierten Fehlers un-

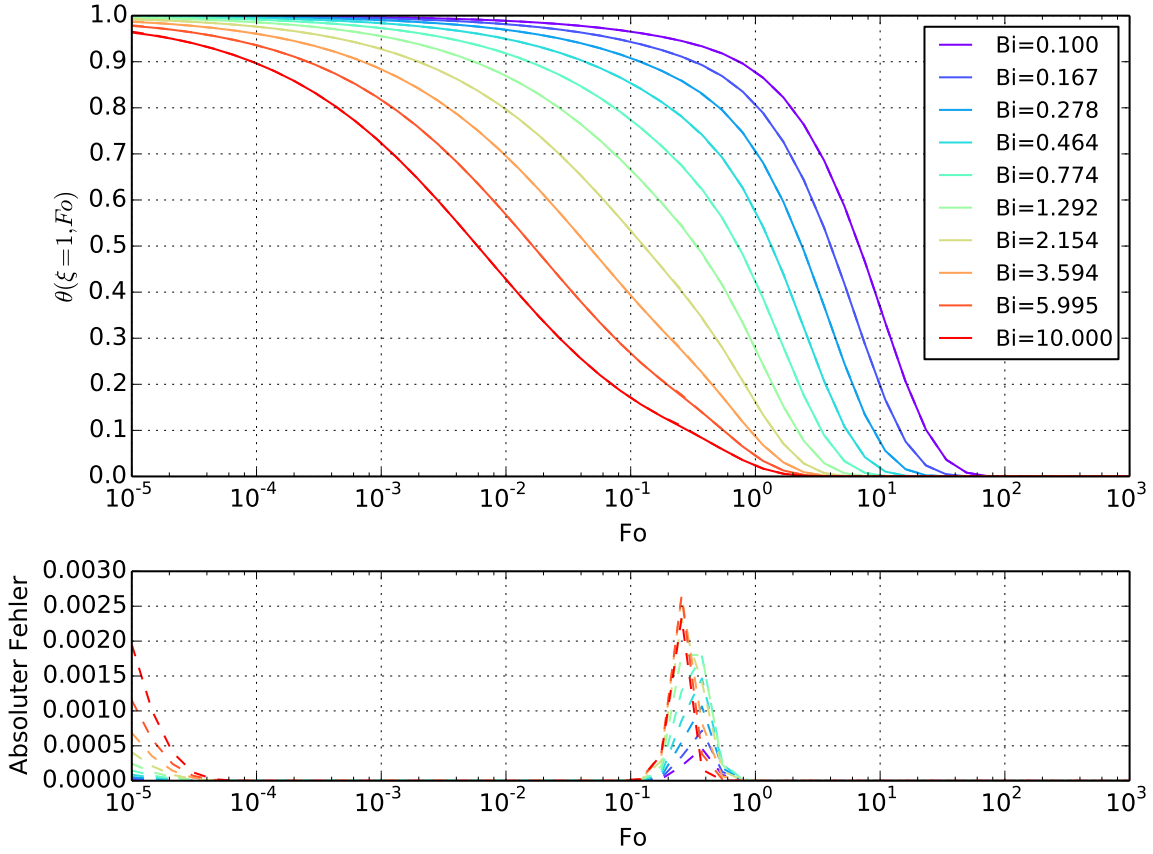


Abbildung 3.13.: Vergleich der Fourier-Reihe (durchgezogene Linie) und der Approximation mit halbunendlichem Körper sowie dem ersten Reihenglied der Fourier-Reihe (gestrichelte Linie) für ein konstantes Temperaturprofil. Der Wechsel auf die Langzeitlösung erfolgt bei $Fo = 0.3$.

tersucht. Für die Untersuchung wird Numpy [62] und Scipy [88] verwendet. Numpy benutzt IEEE-754 Floating Point Zahlen mit doppelter Genauigkeit (64 Bit). Die Maschinengenauigkeit *eps* betrug bei der Fehlerrechnung $2.2204460492503131e-16$.

3.5.6.1. Referenzlösung

Da die Referenzlösung sowohl von der Fourier- als auch von der Biot-Zahl abhängt, wird die numerische Stabilitätsprüfung für beide Variablen separat durchgeführt. Dabei werden diese Eingabeparameter jeweils um eine Maschinengenauigkeit gestört und das Resultat mit dem Ergebnis ohne Störung verglichen. Für die Auswertung wird das Maximum der absoluten Abweichung Δy bestimmt.

Abbildung 3.15 zeigt die Ergebnisse für die Stabilitätsprüfung bei einer Störung der Fourier-Zahl um eine Maschinengenauigkeit. Wenn die Fourier-Zahl um eine Maschinengenauigkeit gestört wird, führt dies zu einer Abweichung von bis zu einer 160

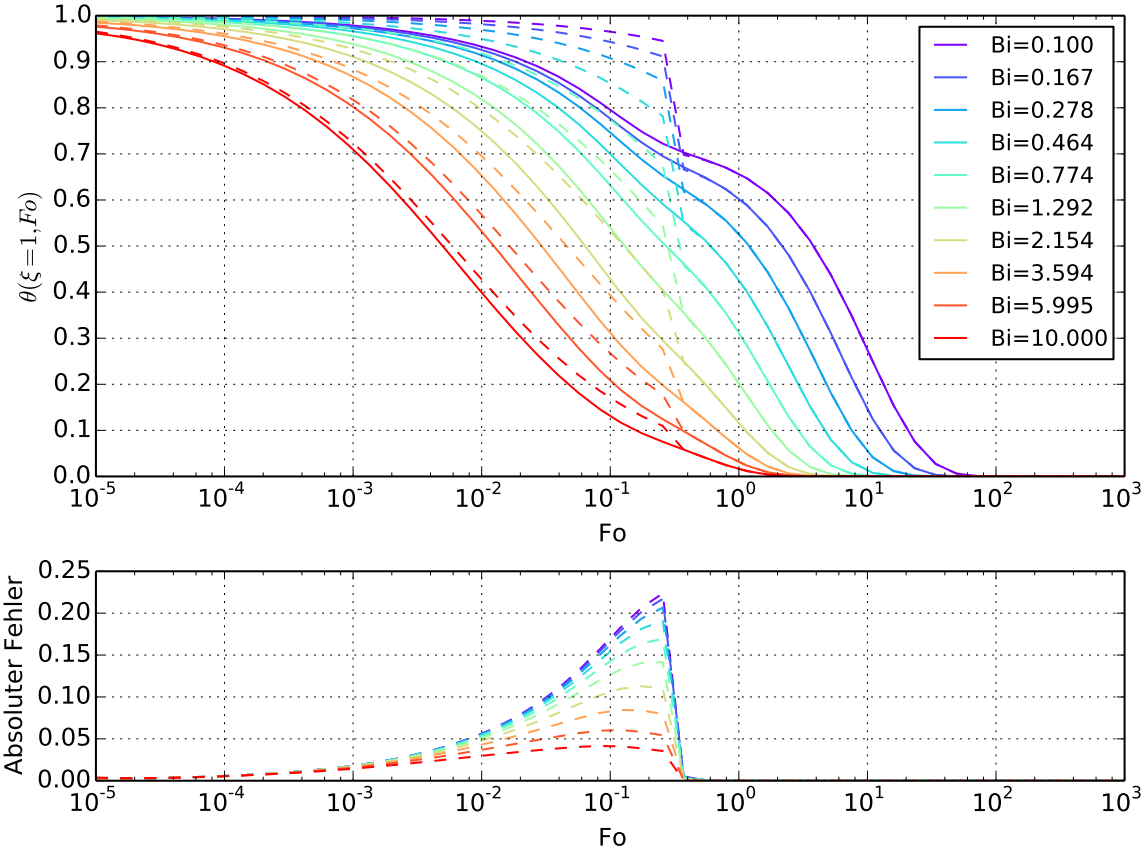


Abbildung 3.14.: Vergleich der Fourier-Reihe (durchgezogene Linie) und der Approximation mit halbunendlichem Körper sowie das erste Reihenglied der Fourier-Reihe (gestrichelte Linie) für ein lineares Temperaturprofil. Der Wechsel auf die Langzeitlösung erfolgt bei $Fo = 0.3$.

fachen Maschinengenauigkeit. Daher ist die Referenzlösung für kleine Fourier-Zahlen numerisch instabil.

Abbildung 3.16 zeigt die Ergebnisse der Stabilitätsprüfung bei einer Störung der Biot-Zahl um eine Maschinengenauigkeit. Die Ergebnisse zeigen, dass die Referenzlösung gegenüber Störungen der Biot-Zahl numerisch stabil ist. Die Lösung weicht maximal um vier Maschinengenauigkeiten ab.

3.5.6.2. Approximation mit zwei Exponentialfunktionen

Bei der Approximation mit zwei Exponentialfunktionen fließt nur direkt die Fourier-Zahl ein. Die Biot-Zahl fließt indirekt über die Kurvenanpassung an die Referenzlösung ein. Im Folgenden wird daher untersucht, wie stabil die Approximation gegenüber einer Störung der Fourier-Zahl sowie der Parameter $\tau_{0,1}$ und $\beta_{0,1}$ ist. Diese Parameter werden zuerst für verschiedene Biot-Zahlen über eine Kurvenanpassung ermittelt und

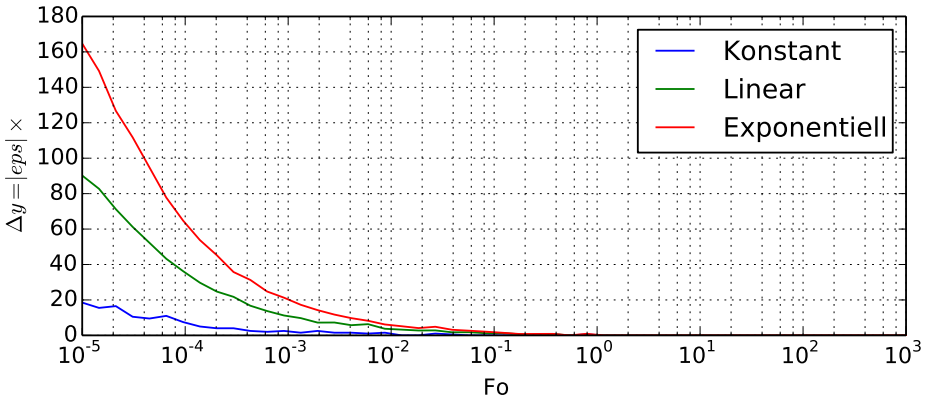


Abbildung 3.15.: Δy bei einer Störung von Fo um eine Maschinengenauigkeit bei der Referenzlösung für $Bi = 0.1$. Die Y-Achse gibt an, um wie viele Maschinengenauigkeiten das Ergebnis abweicht. X-Achse ist logarithmisch skaliert.

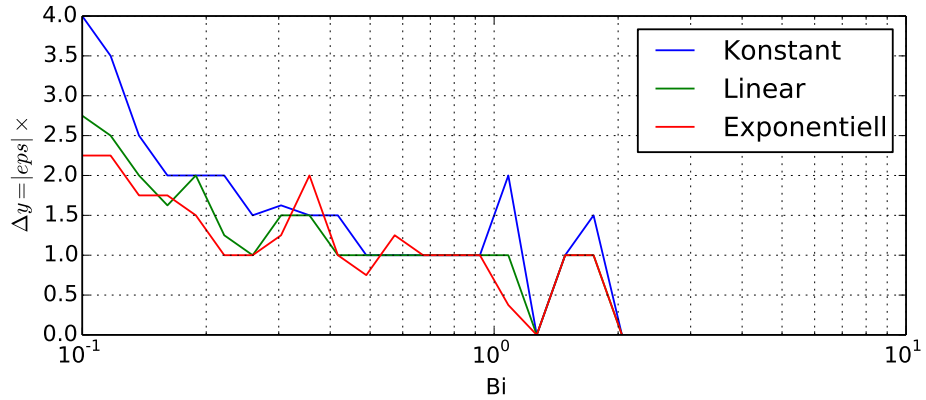


Abbildung 3.16.: Δy bei einer Störung von Bi um eine Maschinengenauigkeit bei der Referenzlösung. Die Y-Achse gibt an, um wie viele Maschinengenauigkeiten das Ergebnis abweicht. X-Achse ist logarithmisch skaliert.

anschließend um eine Maschinengenauigkeit verändert. Für die Auswertung wird das Maximum der absoluten Abweichung Δy bestimmt.

Abbildung 3.17 zeigt Δy bei einer Störung von Fo um eine Maschinengenauigkeit. Das Ergebnis zeigt, dass die Abweichung maximal $3.5 \cdot eps$ entspricht. Daher ist der Algorithmus numerisch stabil gegenüber einer Störung der Fourier-Zahl.

Das Ergebnis der Stabilitätsprüfung, bei einer Störung von $\tau_{0,1}$ und $\beta_{0,1}$ um eine Maschinengenauigkeit, wird in Abbildung 3.18 dargestellt. Das Ergebnis weicht maximal um das 4,5 fache der Maschinengenauigkeit ab. Die Näherung ist daher gegenüber einer Störung von $\tau_{0,1}$ sowie $\beta_{0,1}$ numerisch stabil.

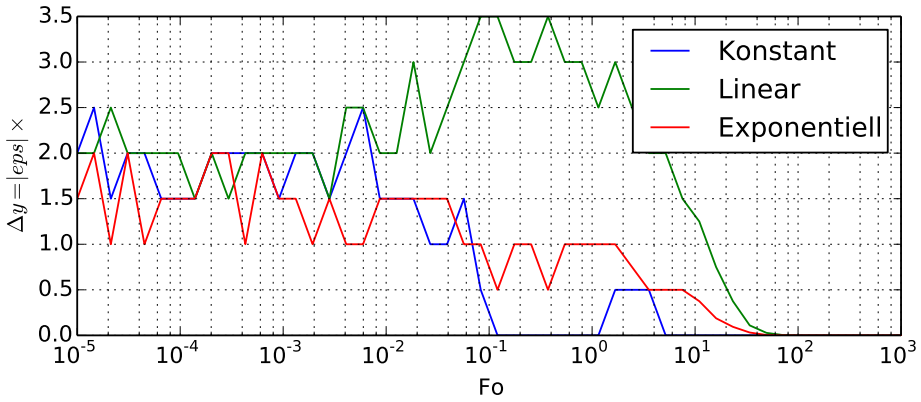


Abbildung 3.17.: Δy bei einer Störung von Fo um eine Maschinengenauigkeit bei der Approximation mit zwei Exponentialfunktionen für $Bi = 0.1$. Die Y-Achse gibt an, um wie viele Maschinengenauigkeiten das Ergebnis abweicht. X-Achse ist logarithmisch skaliert.

3.5.7. Diskussion

In diesem Kapitel wurde ein echtzeitfähiges thermisches Modell vorgestellt, dass einen Temperaturausgleich über zwei Exponentialfunktionen durchführt. Dieses Modell zeichnet sich dadurch aus, dass die rechenintensive Fourier-Reihe durch zwei einfach auszuwertende Funktionen approximiert werden kann. Das vorgestellte thermische Modell wurde mit Verfahren aus der Literatur verglichen und eine Fehlerabschätzung durchgeführt.

Durch die Fehlerabschätzung kann die Auswahl des thermischen Modells auf den jeweiligen Anwendungsfall konkretisiert werden. Für ein konstantes Temperaturprofil liefern die halbunendlichen Körper in Kombination mit dem ersten Reihenglied der Fourier-Reihe die genauste Näherung. Allerdings ist dabei zu berücksichtigen, dass hierfür das Temperaturprofil der Schicht bekannt sein muss, da es in den Koeffizienten des ersten Reihenglieds der Fourier-Reihe einfließt.

Sobald ein anderes Temperaturprofil in der Schicht vorliegt, ist eine Näherung über zwei Exponentialfunktionen vorzuziehen. Der zu wählende Ansatz kann zusätzlich über den Simulationszeitraum eingegrenzt werden. Falls die Simulation für kurze Fourier-Zahlen stattfindet, liefert die Kurvenanpassung aller Parameter die geringere Abweichung zur Referenzlösung. Wenn insbesondere Fourier-Zahlen zwischen 10^{-1} und 10^{1} relevant sind, liefert die Bestimmung der Zeitkonstante τ_1 und des Parameters β_1 über den ersten Eigenwert bzw. den Koeffizienten C_1 eine geringere Abweichung. Für Fourier-Zahlen größer als 10^{1} liefern beide Verfahren zur Parameterbestimmung nur minimale Abweichungen. Die Fehleranalyse hat gezeigt, dass dieses thermische Modell für alle getesteten Biot-Zahlen akzeptable Fehlerschranken liefert und somit universell für viele Materialien eingesetzt werden kann.

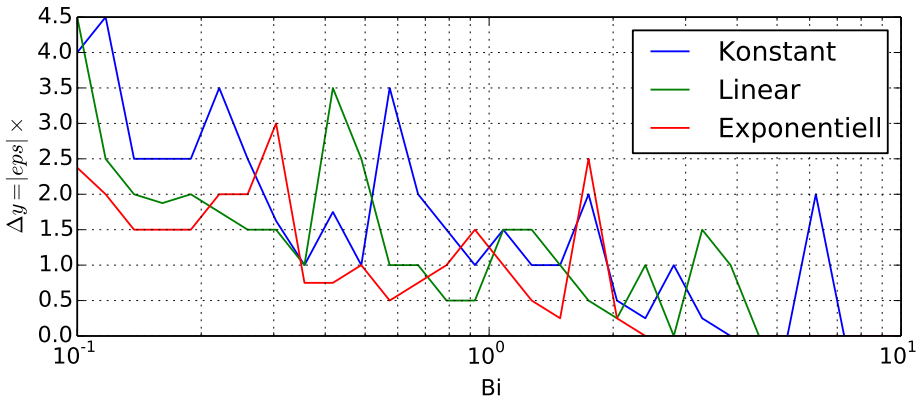


Abbildung 3.18.: Δy bei einer Variation von $\tau_{0,1}$ und $\beta_{0,1}$ um eine Maschinengenauigkeit bei der Approximation mit zwei Exponentialfunktionen. Die Y-Achse gibt an, um wie viele Maschinengenauigkeiten das Ergebnis abweicht. X-Achse ist logarithmisch skaliert.

4. Echtzeitfähige Darstellung von thermischen Schatten

In diesem Kapitel wird eine echtzeitfähige Darstellung von thermischen Schatten vorgestellt. Dabei wird zuerst auf die Zielsetzung und auf unsere Idee eingegangen. Anschließend wird der Stand der Technik erörtert und Details zur Implementierung beschrieben. Am Ende des Kapitels wird eine Fehler- und Geschwindigkeitsanalyse durchgeführt.

4.1. Zielsetzung und Idee

Da die Berechnung einer Oberflächentemperatur die Lösung einer Wärmebilanzgleichung erfordert, werden wichtige thermische Effekte, wie z.B. Abschattungen, oft bei Echtzeitsimulationen vernachlässigt. In anderen Ansätzen wird, des Weiteren, oft die Oberflächentemperatur für 3D-Szenen vorberechnet. Dadurch erlauben diese Ansätze keine dynamischen Abschattungen durch eine Geometrie. Unser Ziel besteht darin, thermische Schatten unter Echtzeitbedingungen zu realisieren. Dabei wird vor allem auch die Unterstützung von dynamischer Geometrie berücksichtigt.

Eine Lösung der 4D-Wärmebilanzgleichung für 3D-Szenen ist sehr rechenaufwändig. Wir versuchen diesen Aufwand zu reduzieren, indem die im vorherigen Kapitel vorgestellte Approximation für das thermische Verhalten von Oberflächen mit einem echtzeitfähigen Bildsyntheseverfahren kombiniert wird. Anstatt einer Lösung einer Wärmebilanzgleichung wird eine Ausgleichstemperatur bestimmt, an die sich die Oberflächentemperatur exponentiell annähert. Dazu wird folgende Annahme getroffen:

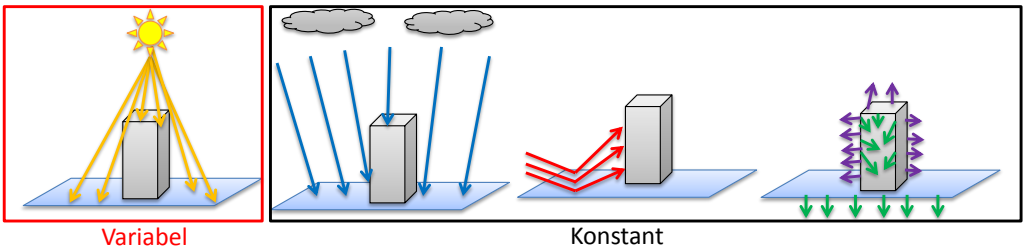


Abbildung 4.1.: Der direkte Wärmestrom wird dynamisch über einen Schattenfaktor berücksichtigt. Die restlichen Wärmeströme werden als Konstante angesehen und fließen in die vorberechneten Temperaturen ein.

Es wird angenommen, dass sich eine Oberflächentemperatur einer Strahlungstemperatur annähert, wenn sie einer direkten Einstrahlung einer Wärmequelle ausgesetzt ist.

Die Oberflächentemperatur, die keiner direkten Einstrahlung ausgesetzt ist, konvergiert hingegen zu einer Schattentemperatur. Die Strahlungs- und Schattentemperaturen entsprechen also der Lösung einer Wärmebilanz mit einem direkten Schattenfaktor von $f_d = 1$ bzw. $f_d = 0$. Anders ausgedrückt, ist in der Energiebilanz nur der direkte Wärmestrom variabel, die anderen Wärmeströme werden durch eine Vorberechnung als Konstant angesehen (Abbildung 4.1).

Um die Ausgleichstemperatur zu ermitteln, wird der Schattenfaktor für einen Oberflächenpunkt einer 3D-Szene bestimmt und für eine Interpolation zwischen der Strahlungs- und der Schattentemperatur genutzt. Die Temperatur wird anschließend mit Hilfe des thermischen Modells aus Kapitel 3.4 an die Ausgleichstemperatur exponentiell angenähert. Abbildung 4.2 zeigt beispielhaft einen Verlauf der Strahlungs- und Schattentemperatur.

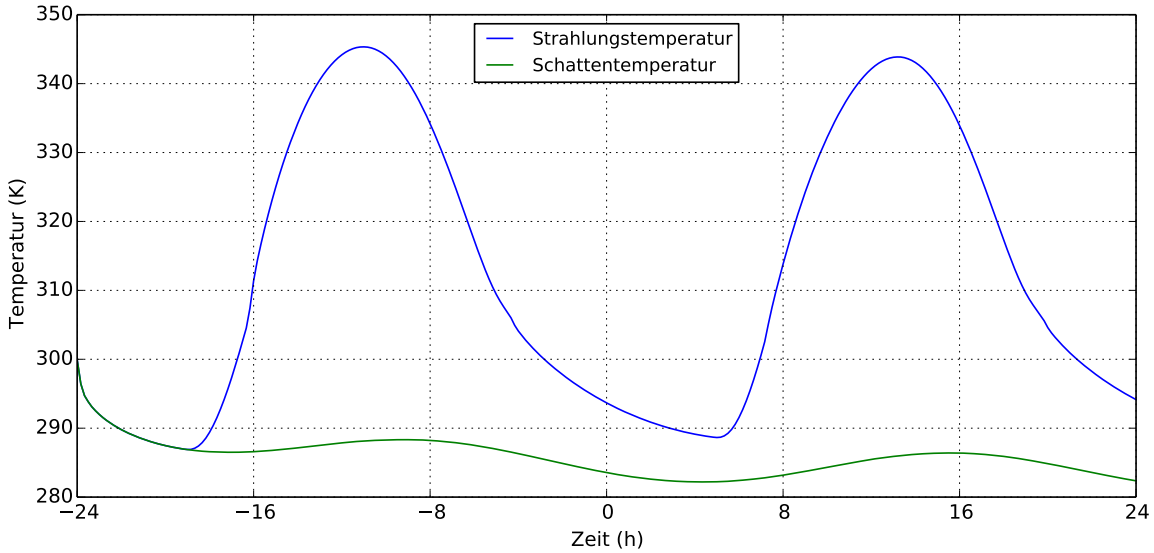


Abbildung 4.2.: Zeitlicher Verlauf der Strahlungs- und Schattentemperatur.

Der Vorteil dieser Vorgehensweise besteht darin, dass die rechenintensive Lösung der Wärmebilanzgleichung für eine Szene vorberechnet werden kann und dynamische Abschattungen dennoch möglich sind. Allerdings werden dadurch Fehler eingeführt, die das Ergebnis gegenüber einer Referenzlösung verändern. Auf die Fehler wird im Kapitel 4.4.1 genauer eingegangen.

4.2. Stand der Technik

Es eine Vielzahl von kommerziellen Produkten für eine 3D-Infrarotbildgenerierung, die auch thermische Schatten berechnen können. Allerdings sind diese Produkte oft unter Verschluss und es werden keine Details zur Umsetzung veröffentlicht. Dieser Stand

der Technik untersucht daher nur wissenschaftliche Veröffentlichungen, die für diese Arbeit relevant sind.

Cathcart [15] realisiert thermische Schatten in einer Simulation, indem eine zusätzliche Geometrie in Form einer Schürze um Schattenspender modelliert wird. Dieser Geometrie wird anschließend optische Eigenschaften zugewiesen.

Poglio et al. [76] triangulieren adaptiv eine Szene, die aus geschichteten, homogenen Patches besteht. Diese Patches werden dazu genutzt um eine Wärmebilanz zu berechnen. Die Wärmebilanz wird mittels eines Radiosity-Verfahrens realisiert und der Schatten mit Shadow Mapping erzeugt.

Biesel und Rohlfing [12] stellen eine echtzeitfähige Infrarotbildgenerierung vor. Die Idee besteht darin, nur Oberflächen mit einer dünnen Schicht zu modellieren. Dadurch ist eine Lösung einer stationären Wärmeleitung möglich. Durch diese Vorgehensweise werden allerdings zeitliche Effekte, wie z.B. thermische Schatten, vernachlässigt.

Schott et al. [87] kombinieren ein nicht-echtzeitfähiges, instationäres thermisches Modell mit einem Ray-Tracer. Dabei werden Materialeigenschaften und zeitabhängige Umgebungsparameter, wie z.B. die Sonneneinstrahlung, genutzt. Um einen Schatten zu erzeugen, wird eine Schattenhistorie für jeden Pixel mit einem Ray Tracer erstellt. Wenn ein Pixel zu einem Zeitpunkt im Schatten ist, wird der Wärmestrom der Sonne auf null gesetzt und die Oberflächentemperatur mit dem thermischen Modell aktualisiert.

Joly [45] benutzt einen Ray Tracer um thermische Schatten zu realisieren. Dabei wird der Sonnenstand in der Vergangenheit abgetastet und die Oberflächentemperatur über ein thermisches Modell angepasst. Joly benennt allerdings keine Details zum thermischen Modell.

Maréchal et al. [56] berechnen eine Wärmeleitung für eine voxelisierte Szene mit Hilfe der Finiten-Volumen Methode um realistische Winterszenarien zu erzeugen. Dabei werden Phasenübergange zwischen Wasser und Eis, sowie Strahlungsbeiträge zwischen den Voxeln berücksichtigt. Für die Berechnung von Schatten wird für jeden Voxel ein Strahl zur Sonne verfolgt. Wenn der Strahl eine Geometrie schneidet, wird der Betrag der Sonneneinstrahlung deaktiviert. Durch diese Vorgehensweise ist die Auflösung des Schattens abhängig von der Feinheit der Voxel.

Weitere Arbeiten beschäftigen sich mit der Berechnung von Infrarotsignatur von Schiffen. Bei diesen Signaturen werden auch thermische Schatten berücksichtigt. Dumont et al. [27] berechnen die Infrarotsignatur von Schiffen mit Hilfe der Finiten-Elemente Methode. Lapierre und Acheroy [51] unterteilen Facetten hierarchisch an Schattenkanten um die Berechnungszeit zu reduzieren. Der Sichtbarkeitsfaktor für die Vertices einer Facette wird dabei mit einem Shadow Volume bestimmt und eine Facette so lange unterteilt, bis sie, mit einer gegebenen Toleranz, in einen Schattenbereich und einem beleuchteten Bereich getrennt werden kann.

4.3. Implementierung

Im folgenden Abschnitt wird genauer auf die Implementierung eingegangen.

4.3.1. Iterative Formulierung des thermischen Modells

Um das vorgestellte thermische Modell auf Grafikkarten umzusetzen, wird es iterativ formuliert. Die Temperatur zum Zeitschritt i kann wie folgt definiert werden:

$$T(i) = Eq(i) + \Delta T_0(i) + \Delta T_1(i) \tag{4.1}$$

Wobei $Eq(i)$ die Ausgleichstemperatur, $\Delta T_0(i)$ und $\Delta T_1(i)$ die Beiträge der Kurz- bzw. Langzeitlösung sind.

Die Ausgleichstemperatur $Eq(i)$ wird über eine Interpolationsfunktion p mit Hilfe eines Schattenfaktors F zwischen der Strahlungstemperatur $S(i)$ und der Schattentemperatur $K(i)$ interpoliert:

$$Eq(i) = p(K(i), S(i), F(i)) \tag{4.2}$$

Die Beiträge der Kurz- bzw. Langzeitlösung werden wie folgt definiert:

$$\Delta T_0(i) = (\Delta T_0(i-1) + \Delta T(i) \cdot \beta_0)(1 - \tau_0 \cdot dt) \tag{4.3}$$
$$\Delta T_1(i) = (\Delta T_1(i-1) + \Delta T(i) \cdot \beta_1)(1 - \tau_1 \cdot dt) \tag{4.4}$$

Mit den Gewichtungsfaktoren β_0 und β_1, den Zeitkonstanten τ_0 und τ_1 sowie der Temperaturdifferenz ΔT, die ausgeglichen werden soll. Der Zeitschritt ist mit dt definiert.

Die Temperaturdifferenz ΔT ergibt sich aus der Zustandsänderung der Ausgleichstemperaturen:

$$\Delta T(i) = p(K(i), S(i), F(i-1)) - Eq(i)$$

4.3.2. Parameterbestimmung des thermischen Modells

Für die Implementierung müssen für jedes Material die Zeitkonstanten des thermischen Modells bestimmt werden. Diese können mit einer Kurvenanpassung gegen das Ergebnis einer numerischen Simulation, wie z.B. mit RadTherm, erzeugt werden. Für die Schattentemperatur wird dabei der Wärmestrom der direkten Einstrahlung auf null gesetzt.

Alternativ können die Parameter mit Hilfe einer Fourier-Reihe in erster Näherung bestimmt werden. Dazu wird allerdings die Biot-Zahl benötigt. Wenn die Oberflächentemperatur nahe an der Ausgleichstemperatur liegt, kann ein Wärmeübergangskoeffizient α anhand des Strahlungsgleichgewichts definiert werden [72]:

$$\alpha = 4\varepsilon\sigma T_R^3 \tag{4.5}$$

Mit der Strahlungstemperatur, bzw. Ausgleichstemperatur, T_R, der Emissivität der Oberfläche ε und der Stefan-Bolzmann Konstante σ. Mit Hilfe des Wärmeübergangskoeffizienten, der spezifischen Länge L und der Wärmeleitfähigkeit λ kann anschließend die Biot-Zahl bestimmt werden:

$$Bi = \frac{\alpha L}{\lambda} \tag{4.6}$$

Mit der Biot-Zahl kann eine Fourier-Reihe numerisch gelöst werden und die Kurvenanpassung analog zu Kapitel 3.4 durchgeführt werden. Allerdings werden bei diesem Ansatz die Temperaturgradienten der Oberfläche vernachlässigt, was vor allem bei Isolatoren zu einem Fehler führt (Siehe Kapitel 3.5).

4.3.3. Berechnung des Sonnenstands

Für die Implementierung wird als Wärmequelle die Sonne genutzt. Um einen realistischen Schattenwurf der Sonne zu ermöglichen, muss der Sonnenstand berechnet werden. Dazu wird der Solar Position Algorithm (SPA) [78] eingesetzt. Auf den Algorithmus wird nicht weiter eingegangen und stattdessen auf Reda und Andreas [78] verwiesen.

Der Algorithmus benötigt als Parameter den Längen- und Breitengrad, die Höhe über den Meeresspiegel, sowie die Zeitzone des Ortes. Für jeden Zeitschritt der Simulation liefert der SPA den Azimut Winkel az, den Zenit Winkel zen sowie den Radius r der Sonne. Um einen Richtungsvektor zur Sonne zu erhalten, werden die Winkel vom geozentrischen Koordinatensystem in kartesische Koordinaten umgewandelt. Dazu ist eine Transformation der Winkel in das Kugelkoordinatensystem notwendig. Die Winkel ϕ bzw. θ ergeben sich wie folgt:

$$\begin{aligned} \theta &= 90^\circ - zen \\ \phi &= -az \end{aligned}$$

Anschließend können die Kugelkoordinaten zu kartesische Koordinaten konvertiert und somit der Richtungsvektor zur Sonne bestimmt werden:

$$
\begin{aligned}
x &= r \cdot cos(\theta) \cdot cos(\phi) \\
y &= r \cdot cos(\theta) \cdot sin(\phi) \\
z &= r \cdot sin(\theta)
\end{aligned}
$$

Abbildung 4.3 zeigt ein Beispiel des Sonnenverlaufs zwischen 8 und 12 Uhr.

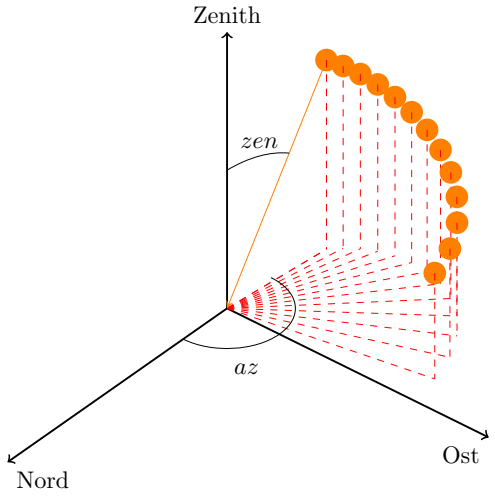


Abbildung 4.3.: Beispiel eines Sonnenverlaufs von 8-12 Uhr im geozentrischen Koordinatensystem.

4.3.4. Materialsystem

Um die vorberechneten Temperaturen und die Parameter des thermischen Modells einer Oberfläche zuzuordnen, wurde ein Materialsystem implementiert. Jeder Oberfläche eines 3D-Modells wird dabei genau ein Material zugewiesen. Ein Material besitzt folgende Daten:

- Die Parameter $\beta_0, \beta_1, \tau_0, \tau_1$ für das thermische Modell.
- Einen zeitlichen Verlauf der Strahlungs- und der Schattentemperatur im Simulationszeitraum (in der Regel von -24 bis +24 Stunden in 10 Minuten Schritten).
- Eine minimale Halbschattenbreite, um einen materialspezifischen lateralen Wärmeaustausch zu approximieren.
- Einen Schalter, der eine thermische Simulation für die Oberfläche unterdrücken kann.

Die Zuordnung zwischen einem Material und einer Oberfläche wird über einen eindeutigen Namen realisiert. Ein Hilfsprogramm liest das 3D-Modell und die Materialdateien ein und konvertiert es in einen OpenSceneGraph Szenengraphen. Dabei wird der Name

der Geometrie mit dem Namen der Materialdatei verglichen. Wenn der Name übereinstimmt, werden die Materialdaten direkt in einem User-Data Feld im Szenengraphen abgelegt. Der Szenengraph wird anschließend als Datei ausgegeben.

Beim Einlesen des Szenengraphen für die thermische Simulation, wertet ein Node-Visitor die User-Data Felder aus. Die Parameter $\beta_0, \beta_1, \tau_0, \tau_1$ werden direkt als Uniform-Variable im StateSet der Geometrie angelegt. Der zeitliche Verlauf der Strahlungs- und Schattentemperatur wird hingegen in eine zweikanalige 32 Bit Float 1D-Textur abgelegt und dem StateSet als Textur-Attribute zugewiesen. Als Texturfilter wird dabei ein linearer Verkleinerungs- und Vergrößerungsfilter gesetzt. Dadurch werden die Zugriffe auf die Strahlungs- und Schattentemperatur zwischen zwei Zeitschritten linear interpoliert.

Die minimale Halbschattenbreite wird bei der Berechnung der weichen Schatten berücksichtigt und dient dazu, einen lateralen Wärmeaustausch zu simulieren. Damit kann allerdings nur ein Verschwimmen der Schattenkanten simuliert werden.

4.3.5. Wahl des Shadow Mapping Verfahrens

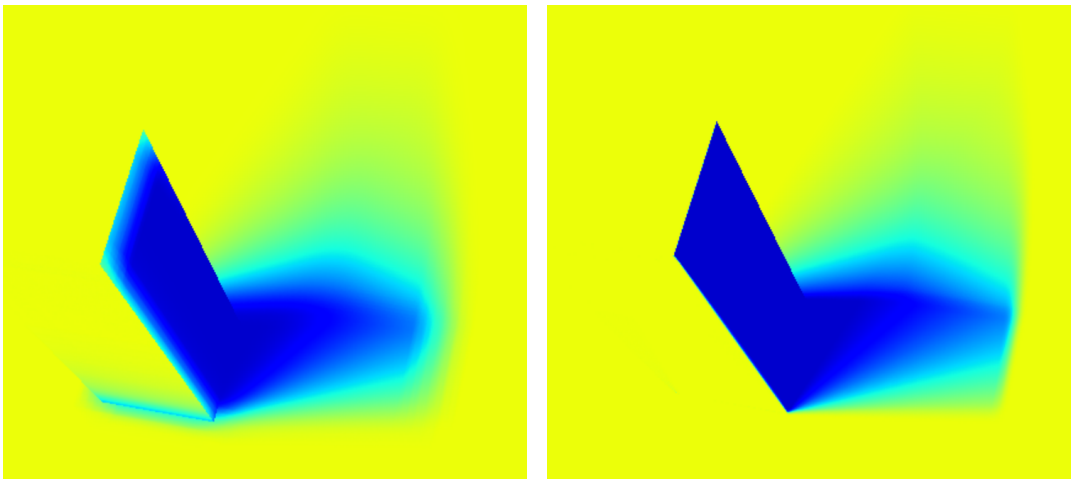
Das Hauptproblem bei der Implementierung besteht darin, für jeden Iterationsschritt der Simulation einen Schatten zu erzeugen. Da das Verfahren unter Echtzeitbedingungen eingesetzt werden soll, muss auf einen schnellen Algorithmus für weiche Schatten zurückgegriffen werden. In die nähere Auswahl kommen dabei:

- Percentage Closer Filtering (PCF)
- Percentage Closer Soft Shadows (PCSS)
- Variance Shadow Mapping (VSM)
- Exponential Shadow Mapping (ESM)

4.3.5.1. Percentage Closer Filtering (PCF)

Der Vorteil von PCF besteht darin, dass ein Halbschatten durch eine Mehrfachabtastung der Shadow Map in einem Filterfenster realisiert wird. Dafür sind weder Anpassungen in der Shadow Map noch zusätzliche Renderpässe notwendig. Die Abtastung der Shadow Map kann jedoch, wenn das Filterfenster groß wird, zu einem Flaschenhals bei den Texturzugriffen führen. Dieser Flaschenhals lässt sich, mit Hilfe einer Poisson-Disk Abtastung mit konstanter Anzahl von Abtastpunkten, vermeiden.

Ein gravierender Nachteil ist jedoch, dass jeder Schatten eine konstante Halbschattenbreite besitzt. Dies ist nicht physikalisch plausibel, da die Halbschattenbreite mit der Entfernung zwischen Schattenspender und Schattenempfänger wächst. Bei einer thermischen Simulation würde eine konstante Halbschattenbreite dazu führen, dass eine Temperaturabkühlung an einer bestrahlten Fläche auftritt (Abbildung 4.4).



(a) Schatten mit konstanter Halbschattenbreite

(b) Schatten mit variabler Halbschattenbreite

Abbildung 4.4.: Vergleich einer konstanten und einer variablen Halbschattenbreite bei der thermischen Simulation. Eine bestrahlte Oberfläche wird durch den Halbschatten an der Kontaktfläche abgekühlt.

4.3.5.2. Percentage Closer Soft Shadows (PCSS)

Die Idee von PCSS besteht darin, das Filterfenster von PCF, abhängig von der Entfernung zwischen Schattenspender und Schattenempfänger, anzupassen. Dadurch ist eine variable Halbschattenbreite möglich. Die Realisierung erfolgt durch eine zusätzliche Abtastung der Shadow Map, um den mittleren Tiefenwert der Schattenspender zu ermitteln. Das PCF-Fenster wird anschließend abhängig von der mittleren Tiefe skaliert.

Das Verfahren besitzt dieselben Vorteile wie PCF und gleicht den gravierendsten Nachteil aus. Allerdings wird die Halbschattenbreite teilweise über- und unterschätzt, was zu Artefakten im Ergebnis führen kann (siehe [4, 28]). Bei der thermischen Simulation ist dies, durch laterale Wärmetransfers, jedoch akzeptabel.

4.3.5.3. Variance Shadow Mapping und Exponential Shadow Mapping

Bei Variance Shadow Mapping (VSM) und Exponential Shadow Mapping (ESM) muss neben dem Tiefenwert weitere Zusatzinformationen für jeden Sonnenstand abgespeichert werden. Dadurch erhöht sich der Aufwand für die Erzeugung der Shadow Maps, was bei vielen Lichtquellen zu einem Flaschenhals führen kann.

Der Hauptnachteil bei VSM bzw. ESM besteht jedoch darin, dass, für eine variable Halbschattenbreite, das Filterfenster für die Vorfilterung je Pixel angepasst werden muss. Dadurch kann keine globale Vorfilterung erfolgen. Abhilfe schafft eine Summed-Area Table, welche allerdings die Größe der Shadow Map einschränkt. Des Weiteren

kann ein Light-Leaking auftreten, was zu fehlerhaften Temperaturanstiegen im Schatten führen kann.

4.3.5.4. Auswahl des Verfahrens

Nach der Abwägung der Vor- und Nachteile ist die Wahl des Verfahrens auf PCSS gefallen. Da für jeden Iterationsschritt eine Shadow Map gerendert werden muss, sollte der Aufwand zur Erzeugung der Shadow Map so gering wie möglich gehalten werden. PCSS erfordert keine Anpassungen beim Rendern der Shadow Map. Dadurch kann das Schreiben der Farbkomponenten des Framebuffers deaktiviert werden. Allerdings erfordert PCSS eine hohe, jedoch konstante Anzahl an Texturzugriffen. Diese profitieren allerdings, aufgrund der Lokalität, von dem Texturcaching.

4.3.6. Forward Rendering Ansatz

Die Idee des Forward Rendering Ansatzes besteht darin, die thermische Simulation in jedem Zeitschritt iterativ durchzuführen. Dadurch wird die Menge des erforderlichen Grafikkartenspeichers reduziert, da nur ein Shadow Map im Speicher gehalten wird. Abbildung 4.5 zeigt das Ablaufdiagramm der Forward Rendering Ansatzes.

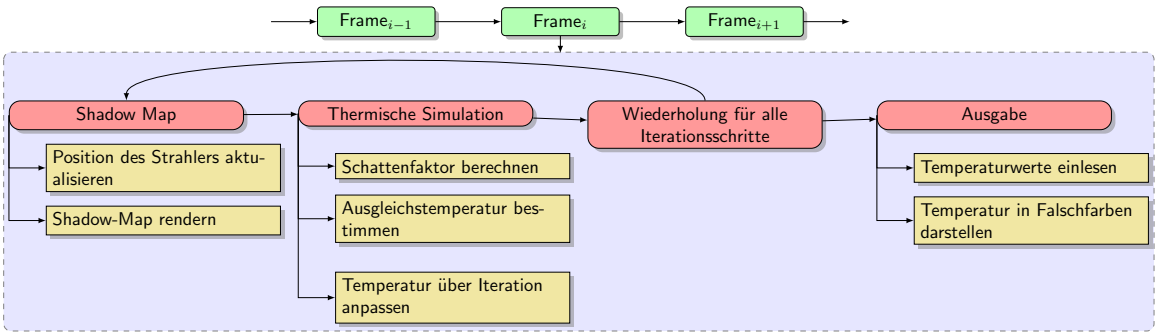


Abbildung 4.5.: Ablaufdiagramm des Forward Rendering Ansatzes.

Der Forward Rendering Ansatz kann grundlegend in die drei Renderpässe eingeteilt werden: Shadow Map, thermische Simulation und Ausgabe.

Im Shadow Map Renderpass wird zuerst die zeitabhängige Position des Strahlers aktualisiert. Dies kann beispielsweise über eine Transformationsfunktion erfolgen. Ausgehend von dieser Position wird dann eine Shadow Map erzeugt. Im Anschluss erfolgt die thermische Simulation. Diese beiden Pässe werden für jeden Zeitschritt im Simulationszeitraum wiederholt, wobei das vorherige Ergebnis als Eingabe für den nächsten Iterationsschritt benutzt wird. Im letzten Pass, der Ausgabe, wird die Temperatur eingelesen und für die Darstellung in Falschfarben umgewandelt.

4.3.6.1. Shadow Map Pass

Für den Shadow Map Pass wurden zwei Varianten implementiert. Die erste Variante rendert eine Shadow Map manuell über eine Schattenkamera. Die zweite Variante benutzt das osgShadow Node-Kit.

Manuell Shadow Map

Bei der manuellen Shadow Map wird eine Kamera erstellt und eine 32-Bit Float 2D-Textur als Shadow Map mittels Framebuffer-Objekt angebunden. Um Aliasing zu reduzieren, wird ein Frustum-Fitting eingesetzt. Dazu wird die Schnittmenge der Szenen Bounding-Box mit dem View- und Licht-Frustum gebildet. Der daraus entstehende konvexe Polyeder wird anschließend entlang der Lichtrichtung ausgedehnt um Schattenspender außerhalb des View-Frustums zu erfassen. Dieser Schritt kann für unendlich entfernte Richtungslichtquellen mit Hilfe der OpenGL Extension `GL_ARB_DEPTH_CLAMP` vereinfacht werden (siehe [28]). Der konvexen Polyeder wird anschließend in eine Bounding-Box eingepasst und die Projektionsmatrix der Schattenkamera auf die Bounding-Box begrenzt.

osgShadow Node-Kit

Bei der Implementierung mittels den osgShadow Node-Kit wird eine Instanz einer `ShadowedScene` erzeugt werden. Bei dieser wird anschließend eine Shadow Map Technik sowie eine Lichtquelle festgelegt. Bei der Traversierung des Szenengraphen sorgt das Node-Kit dann dafür, dass eine Shadow Map, sowie eine Schattenkamera erzeugt werden.

Als Shadow Map Technik wird eine `MinimalShadowMap` eingesetzt, die Aliasing mittels einem Frustum-Fitting reduziert. Diese wird erweitert, so dass zusätzlich die Light-View Matrix, für die Transformation des Tiefenwertes in das Koordinatensystem der Lichtquelle, als Uniform-Parameter übergeben wird.

4.3.6.2. Thermische Simulation

Für den thermischen Simulation Pass sind zwei zusätzliche 32-Bit RGBA Float 2D-Texturen notwendig. Diese speichern die modifizierten Temperaturdifferenzen $\Delta T_{0,1}$, sowie den Schattenfaktor F. In jedem Iterationsschritt wird eine dieser Texturen als Rendertarget gebunden und die andere als Textureinheit für den Shader aktiviert. Für den nächsten Iterationsschritt werden die Texturen getauscht.

Die thermische Simulation wurde mit einem Fragment-Shader umgesetzt. Der Fragment-Shader führt folgende Schritte durch. Im ersten Schritt wird ein Schattenfaktor mit PCSS berechnet. Mit Hilfe dieses Schattenfaktors kann dann die Ausgleichstemperatur nach Gleichung 4.2 bestimmt werden. Die modifizierten Temperaturdifferenzen $\Delta T_{0,1}$

werden anschließend nach Gleichung 4.3 und 4.4 aktualisiert und, zusammen mit dem Schattenfaktor, in eine Textur gespeichert. Im letzten Iterationsschritt wird zusätzlich ein zweites Rendertarget gebunden, indem die Oberflächentemperatur (Gleichung 4.1) gespeichert wird.

4.3.7. Deferred Rendering Ansatz

Eine Möglichkeit um die Geschwindigkeit zu erhöhen, besteht darin, die thermische Simulation von der Geometriedarstellung zu trennen. In der Computergrafik werden solche Verfahren Deferred Rendering oder Deferred Shading genannt (siehe [38]). Ein Geschwindigkeitsvorteil wird dadurch erreicht, dass nur sichtbare Pixel beleuchtet werden. Dies kann jedoch auch bei einem klassischen Forward Renderer durch einen Early-Z Pass erreicht werden. Zusätzlich wird allerdings die Vertex- und die Geometrie-Stufe der Grafikpipeline entlastet, da statt der Geometrie ein Hüllkörper für jede Lichtquelle gerendert wird. Lichtquellen können durch ein Culling des Hüllkörpers mit dem View-Frustum ausgeschlossen werden. Um den Rechenaufwand zu reduzieren kann das Culling nur für jede Gruppe von beispielsweise 32x32 Pixel durchgeführt werden (siehe [17, 91, 7]).

Beim Deferred Rendering Ansatz ist die Vorgehensweise analog zum Forward Rendering Ansatz. Allerdings werden die 3D-Koordinaten aus der Betrachtersicht sowie die Parameter des thermischen Modells vorab in einem Geometry-Buffer (G-Buffer) gerendert. Das besitzt den Vorteil, dass im besten Fall die Geometrie nur einmal aus der Betrachtersicht gerendert werden muss. Abbildung 4.6 zeigt das Ablaufdiagramm des Deferred Rendering Ansatzes.

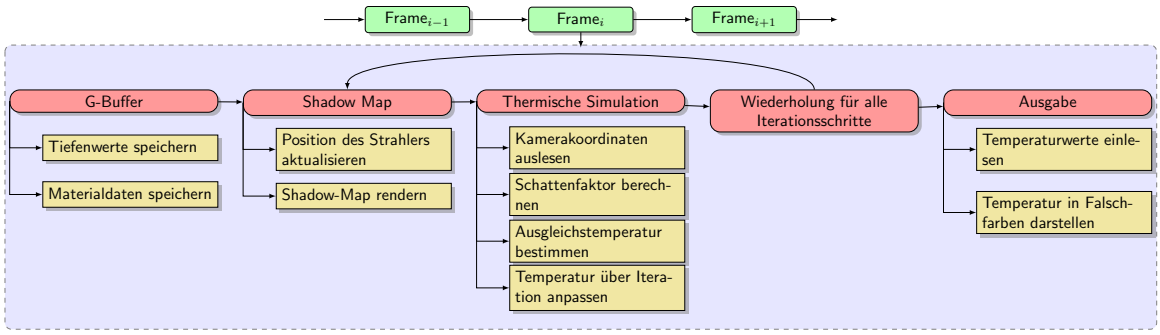


Abbildung 4.6.: Ablaufdiagramm des Deferred Rendering Ansatzes.

Für diesen Ansatz ist zusätzlich ein G-Buffer Renderpass notwendig. Dieser wird immer dann ausgeführt, wenn sich die Geometrie ändert. Im Worst-Case muss dieser also zu jedem Simulationszeitschritt ausgeführt werden. Zusätzlich muss der Node-Visitor des Material-Systems angepasst werden.

4.3.7.1. Anpassung des Node-Visitors des Materialsystems

Der Deferred Rendering Ansatz erfordert eine Anpassung des Materialsystems, da die 1D-Texturen zur Speicherung der Strahlungs- und Schattentemperatur durch den Node-Visitor an die Geometrie gebunden werden. Allerdings wird für den Deferred Rendering Ansatz ein bildschirmfüllendes Rechteck gezeichnet und die Geometriedaten vorab in den G-Buffer gerendert.

Anstatt eine 1D-Textur pro Geometrie zu binden, wird die Strahlungs- und Schattentemperaturen in ein 1D-Texture Array gespeichert. Für jede Geometrie wird eine Uniform-Variable angelegt, die den Index des 1D-Texture Arrays angibt. Beim Rendern des G-Buffers wird dieser Index in den G-Buffer gespeichert.

Alternativ können statt des 1D-Texture Arrays auch Bindless Textures eingesetzt werden. Dazu wird das Handle der Textur in den G-Buffer gespeichert und die Textur, im Fragment-Shader der thermischen Simulation, dynamisch gebunden. Gegenüber 1D-Textures hat dieser Ansatz allerdings den Nachteil, dass ein Texture-Handle 64 Bit groß ist und somit zwei Felder einer G-Buffer Textur belegen würde. Des Weiteren unterstützt OpenSceneGraph in der Version 3.30 keine Bindless-Textures. Daher wurde dieser Ansatz nicht weiter verfolgt.

4.3.7.2. Berechnung der Schattenfaktoren

Für jeden Iterationsschritt der Simulation wird, analog zum Forward Rendering Ansatz, eine Shadow Map gerendert. Allerdings kann bei der Berechnung des Schattenfaktors die Geometrie, durch ein bildschirmfüllendes Rechteck, ersetzt werden. Dazu werden vorab Geometrieinformationen in einen G-Buffer gespeichert. Der G-Buffer wird durch eine 32-Bit RGBA Float 2D-Textur und eine 32-Bit Float 2D-Textur realisiert. Die erste Textur beinhaltet die Zeitkonstanten, die Gewichtsfaktoren und den Index der Lookup-Tabelle für die Strahlungs- und Schattentemperaturen. In der zweiten Textur wird ein linearisierter Tiefenwert aus der Kameraperspektive gespeichert. Dies benötigt weniger Speicherbandbreite als die Speicherung der vollen 3D-Koordinaten (siehe [38, 60, 94]). Die 3D-Kamerakoordinaten können mit Hilfe des Tiefenwertes über einen Sichtstrahl rekonstruiert werden. Anschließend werden die 3D-Kamerakoordinaten mit der View- und Projektionsmatrix der Schattenkamera in das Koordinatensystem der Lichtquelle transformiert und ein Schattenfaktor mit Hilfe von PCSS berechnet. Im Gegensatz zum Forward Rendering Ansatz erfolgt die Transformation der Koordinaten jedoch pro Pixel und nicht pro Vertex.

Gegenüber dem Forward Renderer treten durch die Rekonstruktion nur geringe Artefakte beim Selbstschatten auf (Abbildung 4.7). Hargreaves und Harris [38] schlagen vor, nach einer Rekonstruktion des Tiefenwerts für das Shadow Mapping den Bias anhand des Tiefenwertes zu erhöhen. Allerdings konnten dadurch die Aliasing-Artefakte nicht weiter reduziert werden.

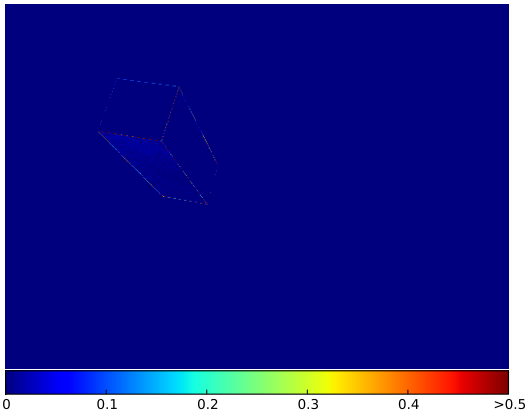


Abbildung 4.7.: Differenzbild zwischen Deferred- und Forward-Renderer. Durch die Rekonstruktion der 3D-Koordinaten treten Unterschiede im Selbstschatten auf.

4.3.8. Kombination mit vorberechneten Temperaturen

Für den praktischen Einsatz ist es oft notwendig, dass die vorgestellte thermische Simulation mit vorberechneten Temperaturwerten einer 3D-Geometrie kombiniert wird. Beispielsweise wird für die 3D-Geometrie von Tracking-Zielen eine aufwändige thermische Simulation, wie z.B. mit RadTherm, durchgeführt. Für die Oberflächen dieser Geometrie soll dann keine thermische Simulation erfolgen, jedoch kann die Geometrie andere Oberflächen abschatten.

Um dies zu realisieren, wird das Materialsystem um einen Schalter erweitert, der die thermische Simulation unterbindet. Wenn der Schalter gesetzt ist, wird beim Einlesen des Szenengraphen der Geometrie-Knoten mit einer Bit-Maske versehen. Bei der Traversierung wird die Bit-Maske getestet und die Knoten ausgeschlossen, die dieses Bit gesetzt haben. Zusätzlich wird ein separater Renderpass ausgeführt, indem alle Geometrie-Knoten mit dieser Bit-Maske gerendert werden. Dabei werden die vorberechneten Temperaturen als Vertex-Attribute eingelesen und durch die Rasterisierung über das Dreieck interpoliert. Im Fragment-Shader wird dann die interpolierte Temperatur direkt ausgegeben und die korrespondierenden Pixel in der Temperaturtextur aktualisiert. Abbildung 4.8 zeigt eine Szene, bei der ein Quader mit vorberechneten Temperaturwerten ausgestattet ist. Der thermische Schatten an der Bodenfläche hingegen wird dynamisch berechnet.

4.3.9. Zeitliche Abtastung

Da thermischen Schatten nur für diskrete Zeitpunkte im Simulationszeitraum berechnet werden, wird der Simulationszeitraum abgetastet. Dazu wurden zwei Abtastungen implementiert, eine gleichmäßige und eine exponentielle Abtastung.

Bei der gleichmäßigen Abtastung wird der Simulationszeitraum gleichmäßig in eine, vom Benutzer vorgegebene, Anzahl von Schritten unterteilt. Da sich die Oberflächen-

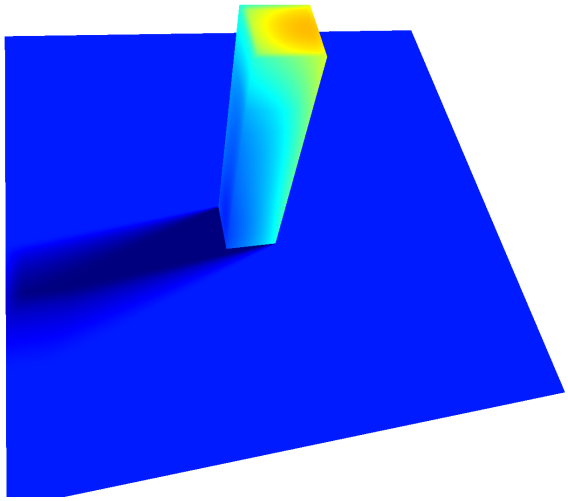


Abbildung 4.8.: Der Quader wurde vorberechnet und der thermische Schatten dynamisch erzeugt.

temperatur allerdings der Ausgleichstemperatur exponentiell annähert, kann es vorkommen, dass Zeitpunkte berücksichtigt werden, die auf das Endergebnis kaum noch Einfluss ausüben. Im Folgenden werden zwei Strategien zur Optimierung der zeitlichen Abtastung vorgestellt.

4.3.9.1. Dauer des Temperaturausgleichs bestimmen

Die erste Strategie besteht darin, die Dauer zu bestimmen, wie lange eine Oberfläche für den Temperaturausgleich, mittels des vorgestellten thermischen Modells aus Kapitel 3.4, benötigt. Simulationszeiten, die diese Dauer überschreiten, üben keinen Einfluss auf das Ergebnis aus und können vernachlässigt werden.

Da sich der Temperaturausgleich für die Langzeitlösung exponentiell verhält (vgl. Kapitel 3.4), konvergiert die Temperatur nur sehr langsam gegen die Ausgleichstemperatur. Um die Genauigkeit zu steuern, wird eine Temperaturtoleranz ε_θ eingeführt. Die Fourier-Zahl für den Temperaturausgleich ergibt sich wie folgt:

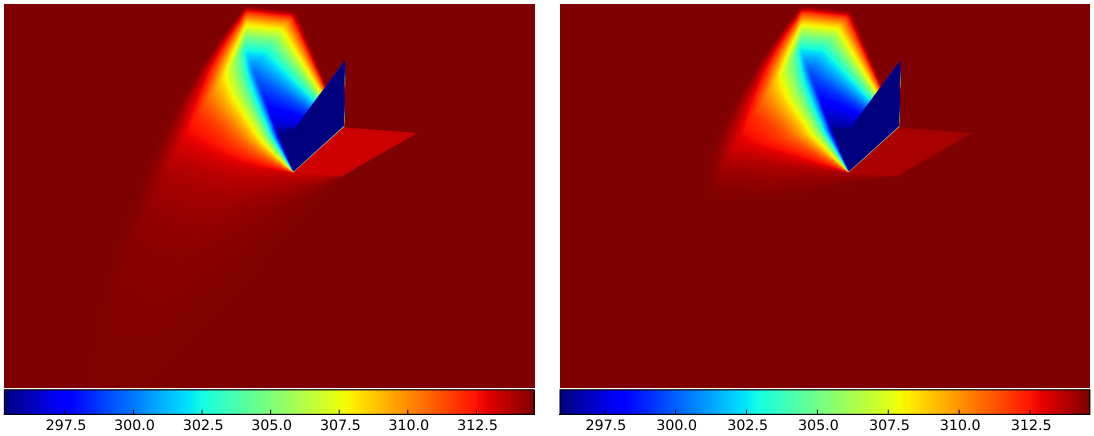
$$Fo = \frac{1}{\delta_1^2} log\left(\frac{C_1}{\varepsilon_\theta} cos(\delta_1)\right) \tag{4.7}$$

Mit $\varepsilon_\theta \neq 0$. Die Temperaturtoleranz gibt an, um wie viel sich die Temperatur nach einer Dauer von Fo von der Ausgleichstemperatur unterscheidet. Durch den Eigenwert δ_1 fließen die Stoffwerte in die Formel ein, wodurch sie abhängig vom Material ist.

Die dimensionsbehaftete Darstellung der Formel 4.7, für die im Kapitel 3.4 vorgestellte Approximation, lautet wie folgt:

$$t = \frac{1}{\tau_1} log\left(\frac{\Delta T}{\varepsilon_T} \beta_1\right) \tag{4.8}$$

Mit der Temperaturtoleranz ε_T [K], sowie den Gewichtungsfaktor β_1 und der Zeitkonstanten τ_1 der Langzeitlösung.



(a) Temperaturtoleranz 0.1 K (b) Temperaturtoleranz 1.0 K

Abbildung 4.9.: Ergebnis mit Simulationsdauer abhängig von der gewählten Temperaturtoleranz.

Tabelle 4.1 zeigt Beispiele für die ermittelte Dauer des Temperaturausgleichs für die Asphaltoberfläche aus Kapitel 3.3. Die Auswirkungen auf das Bild wird in Abbildung 4.9 gezeigt. Bei einer gewählten Temperaturtoleranz von 0.1 K, im Vergleich zu einer Temperaturtoleranz von 1 K, ist die Historie der Schatten deutlich kürzer.

Toleranz ε_T [K]	Simulationsdauer t [s]
0.01	31477 (8,73 h)
0.1	21851 (6,1 h)
0.5	15124 (4,2 h)
1.0	12226 (3,39 h)

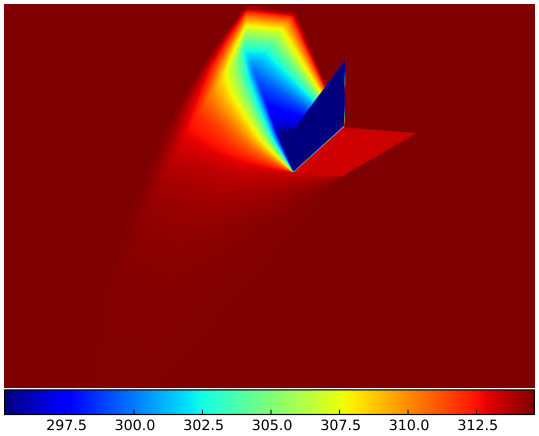
Tabelle 4.1.: Dauer eines Temperaturausgleichs von 20 K für die Asphaltoberfläche aus Kapitel 3.3.

4.3.9.2. Exponentielle Abtastung

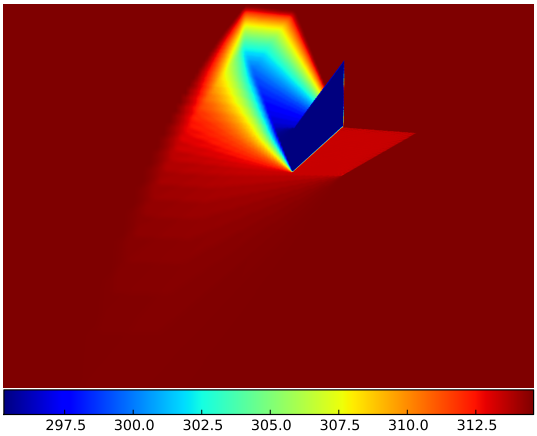
Die zweite Strategie besteht darin, statt der gleichmäßigen Abtastung des Simulationszeitraums eine exponentielle Abtastung durchzuführen. Die Idee besteht darin, dass der Zeitschritt *dt* exponentiell mit der Vergangenheit wächst. Dadurch werden in der nahen Vergangenheit mehr Iterationsschritte durchgeführt als in der fernen Vergangenheit. Es werden also die Bereiche genauer erfasst, bei denen die schnellste Temperaturveränderung stattfindet. Die Realisierung erfolgt durch zwei Benutzerparameter, die den exponentiellen Faktor sowie den minimalen Zeitschritt *dt* angeben.

Durch die exponentielle Abtastung können allerdings deutlich sichtbare Sprünge im Schatten auftreten, da die Schatten mit größeren Abständen berechnet werden.

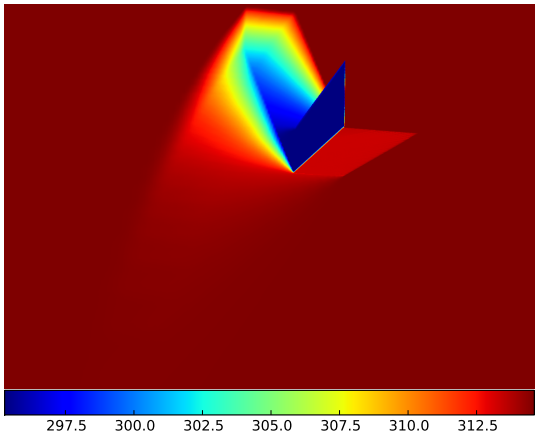
Abbildung 4.10 zeigt ein Beispiel einer exponentiellen Abtastung einer Simulationszeit von 31477 Sekunden. Bei einer gleichmäßigen Abtastung mit einer Abtastrate von $dt = 60$ Sekunden werden 526 Iterationsschritte berechnet. Die exponentielle Abtastung erzeugt hingegen nur 49 Iterationsschritte, wobei in der kurzen Vergangenheit die meisten Iteration durchgeführt werden. Allerdings ist die Diskretisierung der Schattenhistorie deutlich zu sehen (Abbildung 4.10b).



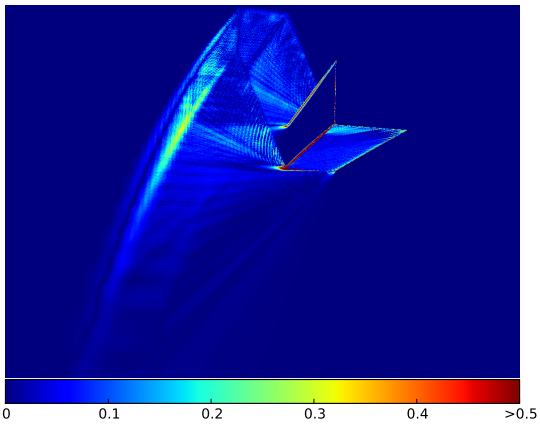
(a) Gleichmäßige Abtastung.



(b) Exponentielle Abtastung. Die Diskretisierung der Schattenhistorie ist deutlich zu erkennen.



(c) Wie (b) jedoch zusätzlich mit exponentieller Anpassung der Halbschattenbreite.



(d) Absolute Differenz der Temperaturen zwischen (a) und (c).

Abbildung 4.10.: Vergleich zwischen einer exponentiellen und gleichmäßigen Abtastung der Simulationsdauer.

Diese Artefakte können reduziert werden, indem die Halbschattenbreite ebenso exponentiell angepasst wird. Das heißt, dass für die ferne Vergangenheit ein größerer Halbschatten berechnet wird als für die nahe Vergangenheit. Physikalisch kann dies

durch einen lateralen Wärmeaustausch auf der Oberfläche begründet werden, bei dem die Schattenkanten verschwimmen. Abbildung 4.10c zeigt das Ergebnis mit einer Anpassung der Halbschattenbreite. Um die Unterschiede zur gleichmäßigen Abtastung sichtbar zu machen, wird ein Differenzbild erzeugt, dass die absolute Differenz der Temperaturen anzeigt (Abbildung 4.10d). Im Schatten entstehen durch die exponentielle Abtastung geringe Temperaturunterschiede die sich vor allem auf die unterschiedliche Halbschattenbreite zurückführen lässt. Des Weiteren findet durch die gleichmäßige Abtastung eine Interpolation der Strahlungs- und Schattentemperatur statt, die bei den großen Zeitsprüngen der exponentiellen Abtastung nicht stattfindet (vgl. Abschnitt 4.3.4).

4.3.10. Mehrfachverwendung von Shadow Maps

Ein Flaschenhals des Algorithmus besteht darin, dass eine Shadow Map für jeden Simulationsschritt gerendert werden muss. Allerdings ist der zeitliche Abstand zwischen zwei Zeitschritten oft klein und somit unterscheiden sich die Einstrahlungsrichtungen der Sonne nur gering. Diese Eigenschaft kann genutzt werden, um die Anzahl der Shadow Maps zu reduzieren.

Die Idee besteht darin, die Shadow Map eines Sonnenstands für mehrere Simulationsschritte zu benutzen. Das entspricht einer Faltung des Filterfensters von PCF oder der Blockersuche mit der Laufbahn der Sonne, welches in einem ellipsoiden Filterkern resultiert (Abbildung 4.11). Für den ellipsoiden Filterkern müssen allerdings die Abtastpunkte dynamisch erzeugt werden, da das Ellipsoid erst nach der Faltung bekannt ist. Des Weiteren ist ein Schattenfaktor zu diskreten Zeitschritten notwendig, um die Temperatur iterativ anzupassen. Wir nähern daher den ellipsoiden Filterkern durch eine Abtastung mit mehreren, diskreten Filterkernen an.

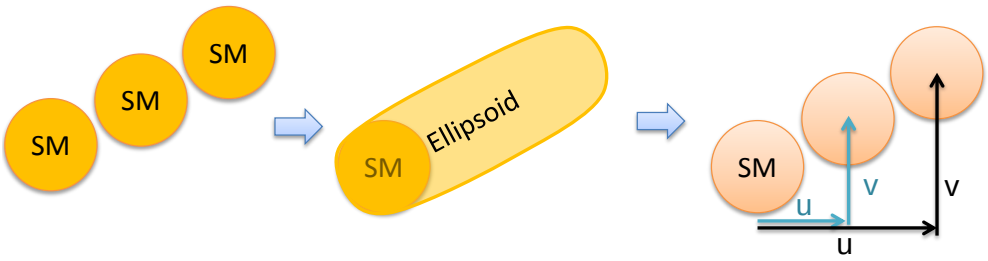


Abbildung 4.11.: Faltung des Filters mit dem Sonnenstand resultiert in einem ellipsoiden Filterkern. Dieser wird durch eine Mehrfachabtastung angenähert. Die Filterkerne werden dabei jeweils um einen u- und v-Abstand verschoben.

Dazu wird der Vektor zwischen zwei Sonnenständen berechnet und dieser in den Texturraum, analog zu Agrawala et al. [2], projiziert:

$$\Delta uv = \frac{\Delta v}{z} z_{near}$$

Wobei Δv der Vektor zwischen den Sonnenständen in Kamerakoordinaten, z der Tiefenwert des Pixels und z_{near} der Tiefenwert der Near-Plane der Kamera ist. Dieser Abstand wird anschließend benutzt um die UV-Koordinaten des Texturzugriffs bei der Blockersuche und PCF zu verschieben.

Wenn jedoch der Abstand zwischen zwei Sonnenständen zu groß ist, entstehen Fehler in der Schattendarstellung. Dies ist insbesondere der Fall, wenn sich die verschobenen Filter-Fenster nicht überlappen. Dadurch entsteht ein Stufeneffekt im Schatten (Abbildung 4.12). Für die Test-Szenen hat sich ein zeitlicher Abstand von 10 Minuten als maximale Schwelle bewährt, andernfalls sind die Stufeneffekte deutlich zu erkennen.

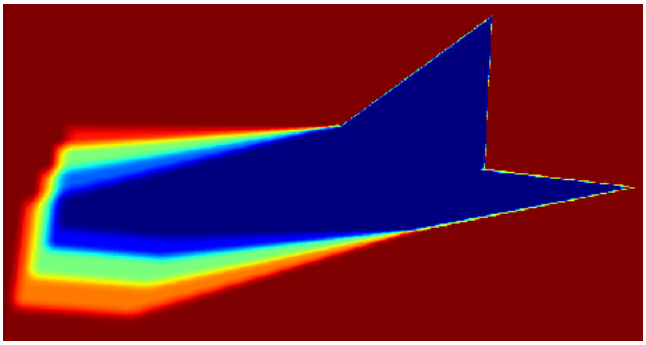


Abbildung 4.12.: Stufeneffekt durch zu großen zeitlichen Abstand.

Ein weiterer Fehler tritt auf, wenn sich die Einstrahlungsrichtungen stark unterscheiden. Dadurch entstehen andere Schatten, da die Schattenspender aus einer anderen Richtung beleuchtet werden (Abbildung 4.13). Bei thermischen Schatten fällt dieser Effekt deutlich auf, da die Schatten zur Veränderung der Temperatur genutzt werden. Gleiche Schatten für mehrere Simulationsschritte fallen daher auch besonders bei Selbstschatten auf.

4.3.11. Dynamische Geometrie

Durch den Einsatz von Shadow Mapping erlaubt sowohl der Forward als auch der Deferred Rendering Ansatz dynamische Geometrie in einer Szene. Dabei muss jedoch zwischen schnellen und langsamen Animationen unterschieden werden.

Schnelle Animationen, wie z.B. ein schnell bewegtes Fahrzeug, sind in der Regel vernachlässigbar da die Oberfläche nur sehr kurz abgeschattet und die Oberflächentemperatur nur minimal verändert wird. Die Temperatur wird schnell wieder ausgeglichen.

Bei langsamen Animationen, oder wenn ein dynamisches Objekt längere Zeit an einem Ort steht, entstehen jedoch thermische Schatten, die berücksichtigt werden müssen. Analog zur Schattenhistorie, muss eine Transformationshistorie für jede Animation vorliegen. Beispielsweise muss zu jedem Simulationszeitpunkt eine Transformationsmatrix vorliegen, die auf die Geometrie angewandt wird. Beim Deferred Rendering Ansatz muss der G-Buffer neu erzeugt werden.

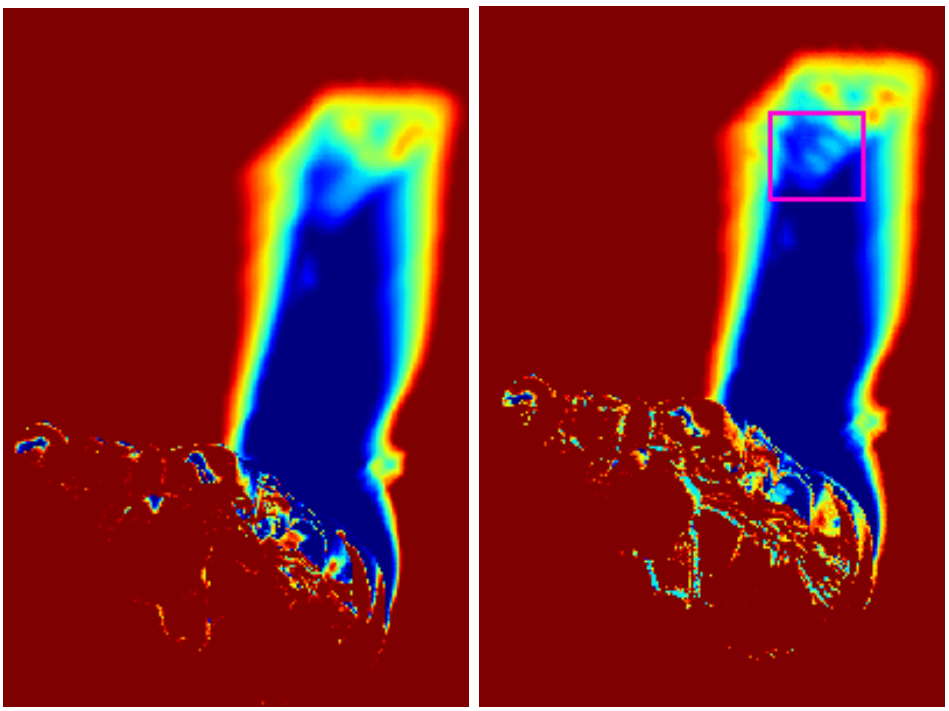


Abbildung 4.13.: Links: keine Mehrfachbenutzung. Rechts: fehlerhafte Schattendarstellung durch Mehrfachbenutzung (gekennzeichneter Bereich).

Eine Geometrie mit zustandserhaltender Animation, z.B. ein deformbarer Körper, wird hingegen nur bei einer fortlaufenden Simulation unterstützt, da sich die Animationszustände nicht in der Vergangenheit rekonstruieren lassen. Dabei darf sich jedoch die Kameraposition des Betrachters nicht ändern. Eine Möglichkeit um diese Beschränkung zu umgehen besteht darin, die Kameraabhängigkeit durch ein Splatting der Temperaturwerte im Objektraum zu ersetzen. Dieser Ansatz wird im Kapitel 7 genauer beschrieben.

4.4. Ergebnisse

Im Folgenden wird auf die Ergebnisse des Algorithmus eingegangen. Dazu wird zuerst eine Fehlerabschätzung durchgeführt und anschließend die Geschwindigkeit und der Speicherverbrauch analysiert.

4.4.1. Fehleranalyse

Bei der Darstellung von thermischen Schatten mit dem beschriebenen Ansatz können Ungenauigkeiten entstehen. Eine Fehlerquelle ist das thermische Modell, das abhängig von der Biot-Zahl eine Temperaturabweichung von bis zu 9% verursacht (vgl. Kapitel 3.5).

Eine weitere Fehlerquelle ist die Vorgehensweise bei der Bestimmung der Ausgleichstemperatur.

4.4.1.1. Fehler bei der Ausgleichstemperatur

Eine Fehlerquelle ist die Approximation der Energiebilanz durch eine Ausgleichstemperatur. Dabei wird angenommen, dass sich das Temperaturprofil eines Körpers durch eine Abschattung nicht ändert. Allerdings kann ein verändertes Temperaturprofil die Oberflächentemperatur maßgeblich beeinflussen (vgl. Kapitel 3.5). Der Algorithmus ist vor allem auf eine kurze Simulationsdauer ausgelegt, wodurch ein Temperaturprofil in der Regel nicht stark beeinflusst wird. Da der Fehler von der Energiebilanz abhängt, lässt er sich nur sehr schwer quantifizieren.

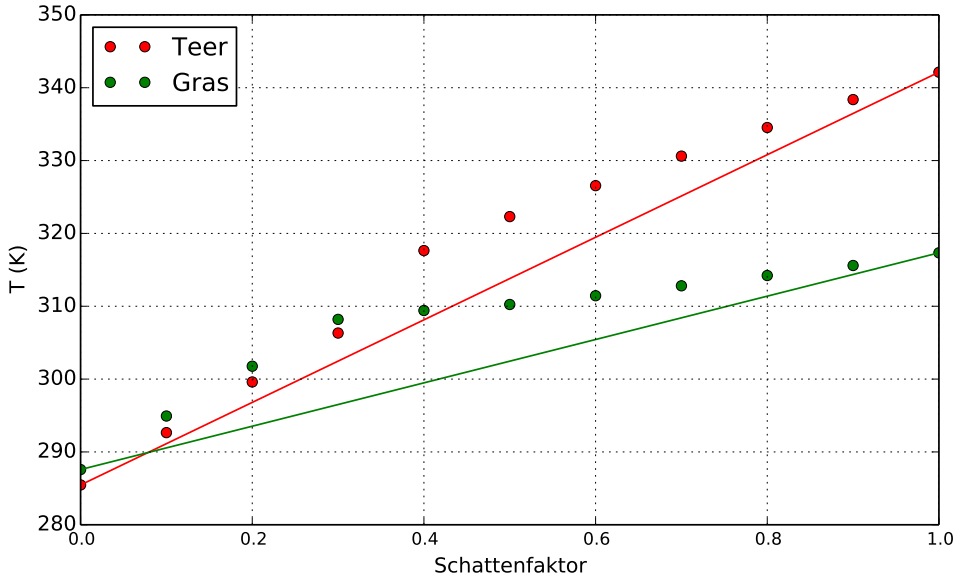


Abbildung 4.14.: Fehler bei einer linearen Interpolation der Ausgleichstemperatur mittels Schattenfaktor für eine Teer- und Grasoberfläche.

Auch bei einer linearen Interpolation zwischen Strahlungs- und Schattentemperatur wird eine Ungenauigkeit verursacht. Die Temperaturunterschiede verhalten sich nicht linear. Abbildung 4.14 zeigt die Auswirkung der Interpolation bei verschiedenen Materialien. Es wird ein Temperaturunterschied von bis zu 10 Kelvin verursacht, allerdings nur im Halbschattenbereich. Für Oberflächen mit Schattenfaktor Null oder Eins entsteht hingegen keine Abweichung. Um die Genauigkeit zu erhöhen kann die lineare Interpolation durch eine exponentielle Anpassung ersetzt werden. Allerdings muss dann, für jeden Zeitschritt, zusätzlich die Faktoren für die Interpolationsfunktion gespeichert werden.

Eine weitere Ungenauigkeit gegenüber einer Referenzlösung besteht darin, dass die indirekte Umgebungsstrahlung, sowie Konvektion, Wärmeleitung und Eigenstrahlung

für jedes Material vorberechnet und somit konstant ist. Besonders die indirekte Umgebungsstrahlung hängt von der Umgebung und vom sichtbaren Raumwinkel der Hemisphäre ab. Im Kapitel 6 wird der Algorithmus um einen indirekten Strahlungsanteil ergänzt.

4.4.1.2. Vergleich mit Messung und numerischer Simulation

Abbildung 4.15 zeigt den Vergleich der Temperaturänderung einer Sonnenfläche im Verhältnis zu einer Schattenfläche gegenüber der Messung und einer numerischen Simulation mit RadTherm. Unsere Approximation kann das Temperaturverhalten der Oberfläche in diesem Zeitraum mit einer maximalen Abweichung 0.5 Kelvin darstellen.

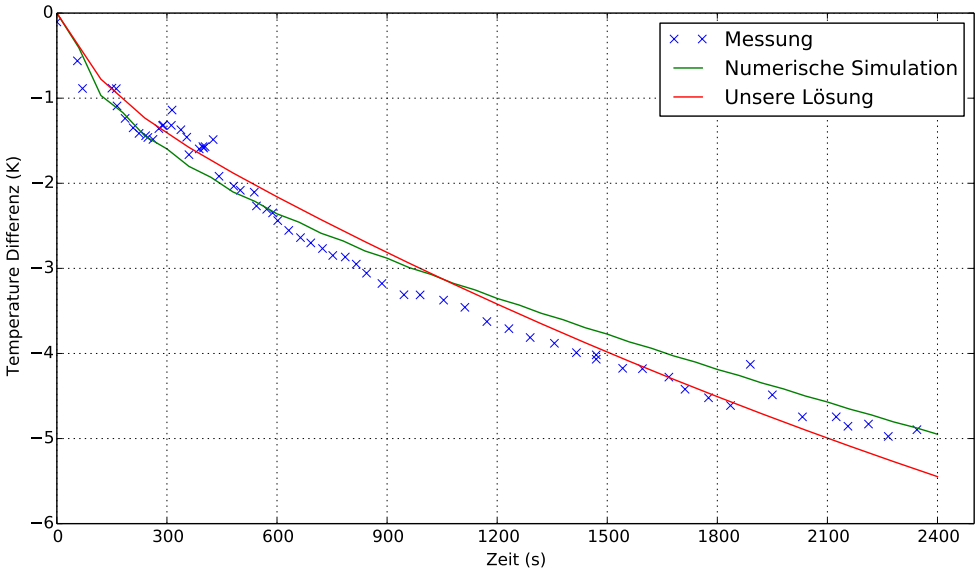
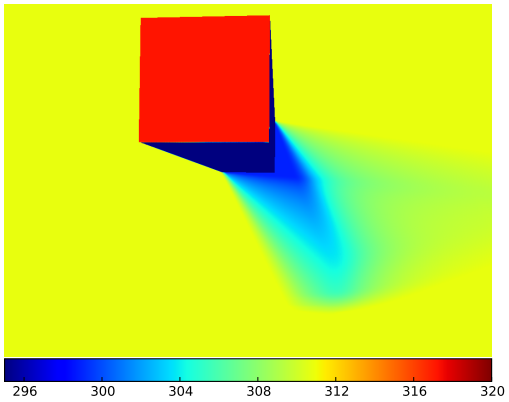


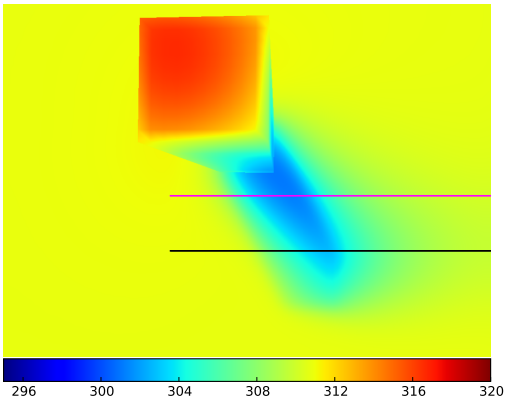
Abbildung 4.15.: Vergleich zwischen Messung, numerischer Simulation mit RadTherm und unserer Approximation.

4.4.1.3. Vergleich der absoluten Temperaturen mit einer Referenzlösung

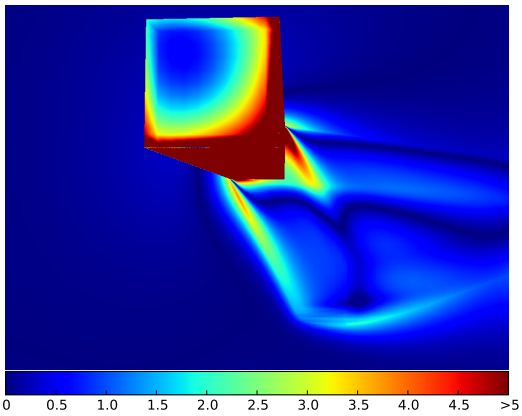
In diesen Abschnitt werden die absoluten Temperaturen unserer Approximation gegen eine Referenzlösung verglichen. Als Referenzlösung wird RadTherm eingesetzt. Für den Vergleich wird zunächst die Strahlungs- und Schattentemperatur mit RadTherm vorberechnet und die Parameter des thermischen Modells über eine Kurvenanpassung gegen die Referenzlösung ermittelt. Die Szene besitzt zwei Materialien. Der Boden besteht aus Asphalt und der Quader aus Zement. Als Simulationszeit wird eine Zeit von 8:00 - 11:30 Uhr mit einen Zeitschritt von 60 Sekunden gewählt. Um den Einfluss der unteren Oberflächenschichten zu reduzieren, wurde die RadTherm Lösung mit einer Simulationszeit von 0:00 bis 11:30 Uhr berechnet.



(a) Ergebnis unserer Approximation



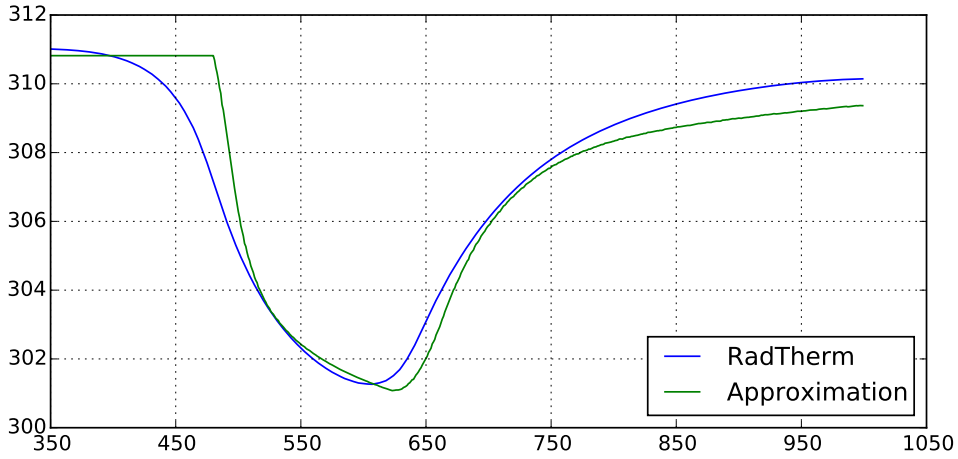
(b) Ergebnis von RadTherm. Die Linien zeigen die Stellen für die Temperaturkurven an.



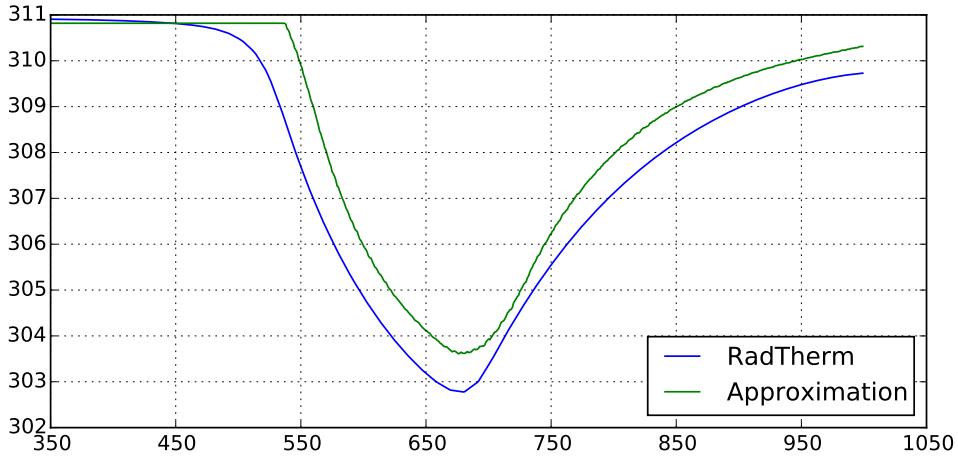
(c) Differenzbild zwischen (a) und (b).

Abbildung 4.16.: Ergebnis unserer Approximation im Vergleich zur RadTherm Lösung.

Abbildung 4.16 zeigt die visuellen Ergebnisse. Bei dem visuellen Vergleich fällt sofort auf, dass in unserer Lösung die Orientierung der Oberfläche vernachlässigt wird. Der Deckel des Quaders, sowie die Seitenwände sind in einer einheitlichen Temperatur. An den Seitenwänden des Quaders überwiegt zudem die Wärmeleitung innerhalb des Körpers, dass durch unseren Algorithmus nicht berücksichtigt wird. Die Temperaturunterschiede werden im Differenzbild in Abbildung 4.16c dargestellt. Des Weiteren fällt auf, dass die Halbschattenbreite, besonders auch im Bereich des Kernschattens, bei der Referenzlösung deutlich größer als bei unserer Lösung ist. Dadurch ist die Temperatur bei unserer Lösung an diesen Stellen zu warm bzw. zu kalt. Allerdings kann, durch die Poisson-Disk Abtastung, die Halbschattenbreite nicht weiter angehoben werden, da sonst Blockartefakte entstehen.



(a) Temperaturkurve in der magentafarbenen Bildzeile.



(b) Temperaturkurve in der schwarzen Bildzeile.

Abbildung 4.17.: Temperaturvergleich in den markierten Bildzeilen aus Abbildung 4.16b

Diese Eigenschaften können auch in den Temperaturkurven in Abbildung 4.17 beobachtet werden. Im Kernschatten ist unsere Lösung um bis zu 0.8 K kühler als die Referenzlösung. Im Halbschatten beträgt der maximale Temperaturunterschied 3.3 K.

Eine Auswertung für weitere Materialien ist im Anhang A.4 zu finden.

4.4.2. Geschwindigkeitsanalyse

In diesem Abschnitt wird die Geschwindigkeit der beiden vorgestellten Ansätze verglichen. Die Schatten werden mit PCSS mit 8 Abtastpunkten für die Blockersuche und 32 Abtastpunkte für PCF gerendert. Die Auflösung beträgt 1024x768 Pixel und

die Shadow Map Auflösung ist 1024x1024. Für die Geschwindigkeitsmessungen wird die manuelle Shadow Map Implementierung eingesetzt. Die Geschwindigkeit wird auf einen Intel i7-3820 Prozessor mit 3,6 GHz, 16 GB RAM und einer NVIDIA GeForce GTX Titan mit 6144 MB Speicher ermittelt. Die Angaben zur Rechendauer sind jeweils die Mittelwerte aus 1000 Bildern.

Tabelle 4.2 zeigt die Rechendauer der beiden Ansätze bei einer gleichmäßigen Abtastung. Die Simulationszeit ist von 9:00 - 12:00 Uhr mit einemn Zeitschritt von 2 Minuten. Des Weiteren wird die Rechendauer für den Worst-Case beim Deferred Rendering Ansatz gemessen, bei dem der G-Buffer in jedem Zeitschritt neu berechnet wird. Bei der Quader-Szene, die nur 45 Polygone besitzt, ist der Forward Rendering Ansatz geringfügig schneller, da die Dauer des Renderns der Geometrie vernachlässigbar ist. Beim Deferred Rendering Ansatz hingegen müssen die Vertex-Positionen zuerst in eine Textur gerendert und pro Pixel die Kamerakoordinaten in das Koordinatensystem der Lichtquelle und der Shadow Map transformiert werden. Wenn die Geometrie in der Szene deutlich mehr Polygone besitzt, wie beispielsweise bei der Buddha-Szene mit 1.08 Millionen Dreiecken, ist der Deferred Rendering Ansatz schneller. Gegenüber dem Forward Renderer wird ein Faktor 1.6 an Rechenzeit eingespart. Im Worst-Case ist der Deferred Renderer jedoch langsamer.

Szene	**Forward**	**Deferred**	**Deferred (Worst-Case)**
Quader	20.3 ms	20.7 ms	31.2 ms
Buddha	78.6 ms	49.0 ms	87.5 ms

Tabelle 4.2.: Rechendauer bei den Testszenen bei einer gleichmäßigen Abtastung mit 90 Schritten.

Tabelle 4.3 zeigt die Rechendauer der einzelnen Renderpässe in der Buddha-Szene mit 90 Simulationsschritten an. Durch das Rendern der Geometrie in jedem Simulationsschritt dauert die thermische Simulation um Faktor 3.6 länger als beim Deferred Rendering Ansatz, bei dem nur ein bildschirmfüllendes Rechteck gezeichnet wird. Der Flaschenhals des Verfahrens ist beim Forward Renderer das doppelte Rendern der Geometrie für die Shadow Maps und für die thermische Simulation. Dadurch ist der Deferred Renderer schneller als der Forward Renderer, da bei der thermischen Simulation die Geometrieinformationen aus dem G-Buffer verwendet werden.

Schritt	**Forward**	**Deferred**
G-Buffer	-	0.8 ms
Shadow Maps	36.1 ms	36.1 ms
Thermische Simulation	41.8 ms	11.4 ms
Darstellung	0.7 ms	0.7 ms
Gesamt	78.6 ms	49.0 ms

Tabelle 4.3.: Dauer der einzelnen Renderpässe beim Forward und Deferred Rendering Ansatz in der Buddha-Szene mit 90 Simulationsschritten.

In Tabelle 4.4 wird die Rechendauer der Buddha-Szene mit 90 Simulationsschritten bei Mehrfachverwendung von Shadow Maps angezeigt. Die Rechendauer kann um Faktor 2.1 beim Forward und um Faktor 1.8 beim Deferred Rendering Ansatz reduziert werden, wenn eine Shadow Map für vier Iterationsschritte benutzt wird. Durch eine Mehrfachverwendung steigt jedoch die Dauer des thermische Simulation Pass an, wodurch keine lineare Geschwindigkeitssteigerung erreicht wird.

Iterationen je Shadow Map	**Iterativ**	**Deferred**
1	78.6 ms	49.0 ms
2	48.8 ms	34.8 ms
3	40.9 ms	29.4 ms
4	37.6 ms	26.7 ms

Tabelle 4.4.: Rechendauer der Buddha-Szene mit 90 Simulationsschritten bei Mehrfachverwendung von Shadow Maps.

In Abbildung 4.18 wird dargestellt, wie die Rechendauer des Deferred Rendering Ansatzes von der Anzahl der Simulationsschritte in der Buddha-Szene abhängt. Die gesetzte Grenze von 16,67 ms, bei 60 Bildern pro Sekunde, wird nach 30 Schritten erreicht. Werden hingegen vier Iterationen je Shadow Map eingesetzt, können 55 Schritten berechnet werden. Die Grafik zeigt weiter, dass die Rechendauer linear mit der Anzahl der Simulationsschritte steigt.

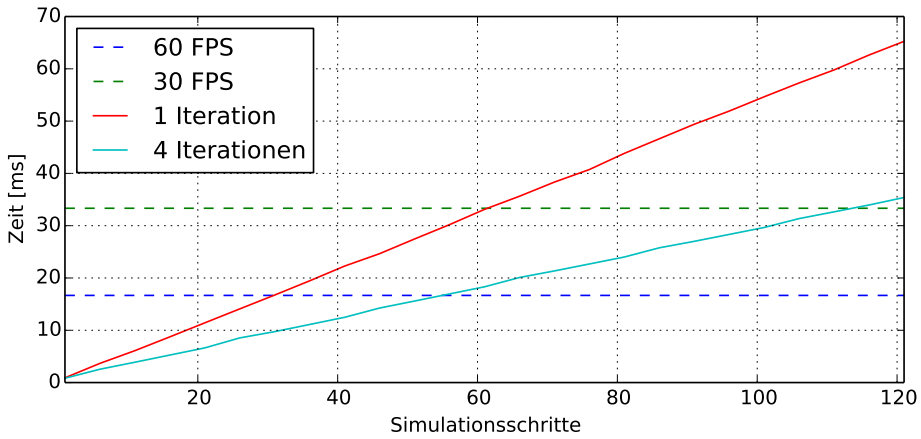


Abbildung 4.18.: Rechendauer in Abhängigkeit der Anzahl der Simulationsschritte.

In Tabelle 4.5 wird die Rechendauer pro Bild in Abhängigkeit von der Shadow Map Auflösung bei der Buddha-Szene mit 90 Simulationsschritten dargestellt. Die resultierenden Bilder werden in Abbildung 4.19 dargestellt. Bei einer Shadow Map Auflösung von 512x512 Pixeln sind Treppenstufeneffekte am Sockel der Statue zu erkennen. Diese sind ab einer Auflösung von 1024x1024 nicht mehr vorhanden. Gegenüber einer Auflösung von 2048x2048 oder 4096x4096 sind im Schatten keine Unterschiede auszumachen. Erst auf den Differenzbildern werden die Unterschiede sichtbar. Diese treten

besonders beim Selbstschatten auf der Statue auf.

Auflösung	Forward	Deferred
512x512	81.9 ms	53.5 ms
1024x1024	78.6 ms	49.0 ms
2048x2048	81.2 ms	50.9 ms
4096x4096	98.1 ms	63.8 ms

Tabelle 4.5.: Rechendauer in Abhängigkeit von der Shadow Map Auflösung bei der Buddha-Szene mit 90 Simulationsschritten.

In Tabelle 4.6 wird die Rechendauer pro Bild in Abhängigkeit von der Bildschirmauflösung angezeigt. Da die thermischen Schatten pro Pixel berechnet werden, steigt die Rechendauer mit der Bildschirmauflösung. Bei einer 1080p Auflösung benötigt der Deferred Renderer 84.2 ms und der Forward Renderer 121.9 ms.

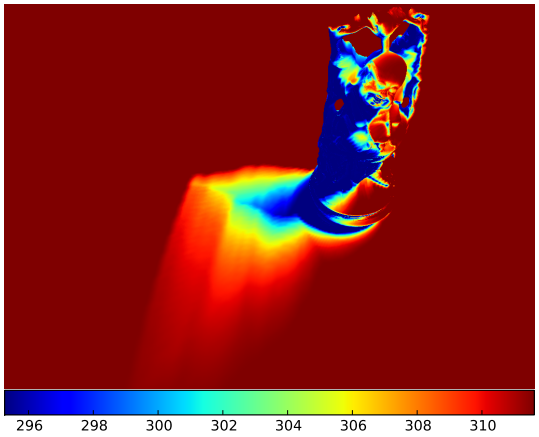
Auflösung	Forward	Deferred
512x512	70.5 ms	45.8 ms
1024x768	78.6 ms	49.0 ms
1280x720	87.4 ms	55.6 ms
1920x1080	121.9 ms	84.2 ms

Tabelle 4.6.: Rechendauer in Abhängigkeit von der Bildschirmauflösung bei der Buddha-Szene mit 90 Simulationsschritten.

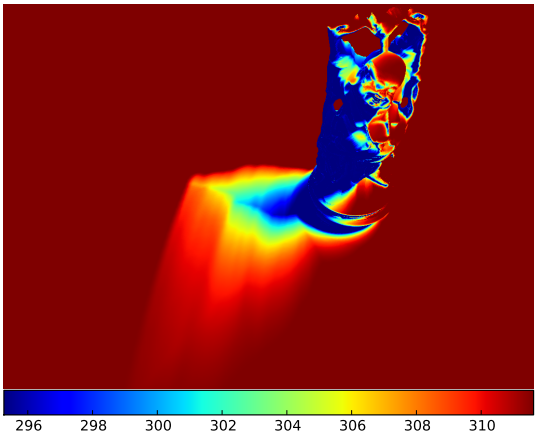
In Abbildung 4.20 wird das Ergebnis der eigenen Shadow Map Implementierung mit dem Ergebnis bei der Benutzung des osgShadow Node-Kit verglichen. Die zugehörige Rechendauer wird in Tabelle 4.7 angezeigt. Beim Frustum-Fitting der Schattenkamera wird beim osgShadow Node-Kit genauer zwischen Schattenspender und Schattenempfänger unterschieden. Dazu wird der Hüllkörper der Geometrie separat für die Schattenspender und Schattenempfänger bestimmt, was zwei Traversierungen des Szenengraphen erfordert. Bei der eigenen Shadow Map Implementierung wird der gleiche Hüllkörper für die Schattenempfänger und Schattenspender benutzt. Dadurch ist die Implementierung schneller, allerdings wird das Aliasing an den Schattenkanten dadurch weniger reduziert.

Shadow Map Implementierung	Forward	Deferred
Eigene Implementierung	76.8 ms	49.0 ms
osgShadow Node-Kit	81.1 ms	50.9 ms

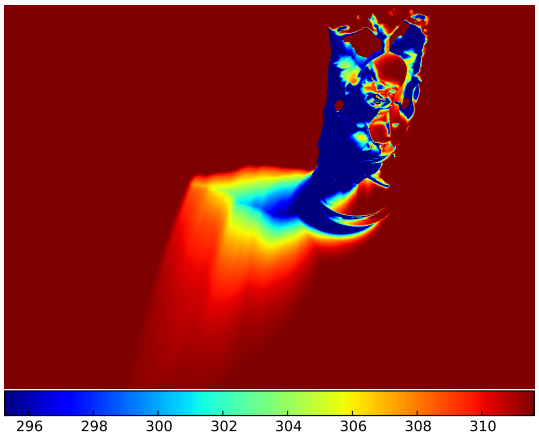
Tabelle 4.7.: Vergleich der Rechendauer zwischen der eigenen Shadow Map Implementierung und dem osgShadow Node-Kit bei der Buddha-Szene mit 90 Simulationsschritten. Shadow Map Auflösung 1024x1024.



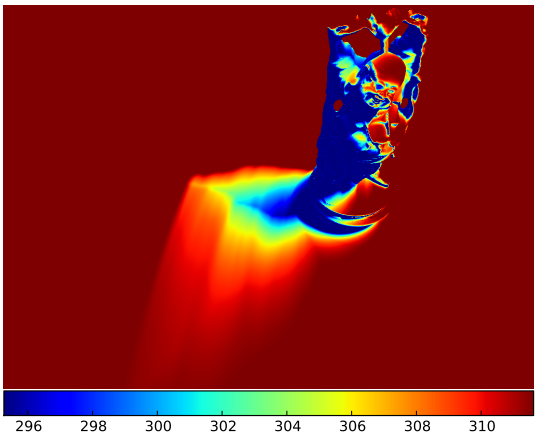
(a) Ergebnis mit 512x512 Shadow Maps



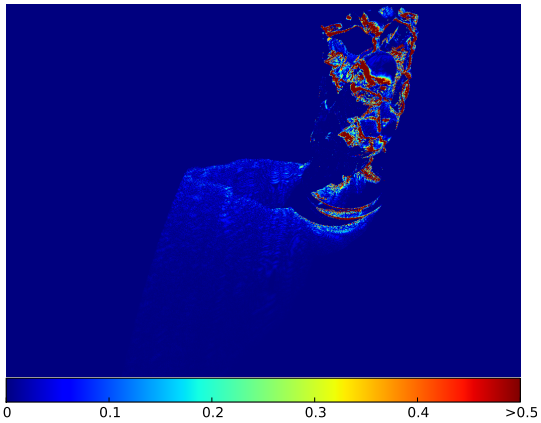
(b) Ergebnis mit 1024x1024 Shadow Maps



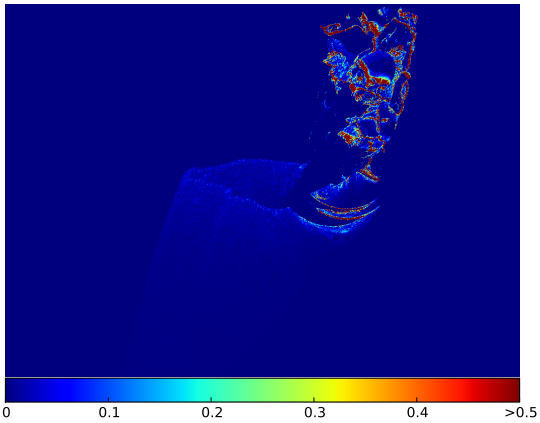
(c) Ergebnis mit 2048x2048 Shadow Maps



(d) Ergebnis mit 4096x4096 Shadow Maps

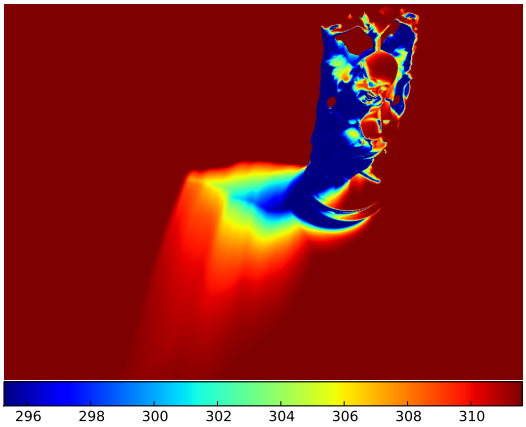


(e) Differenzbild zwischen (b) und (c).

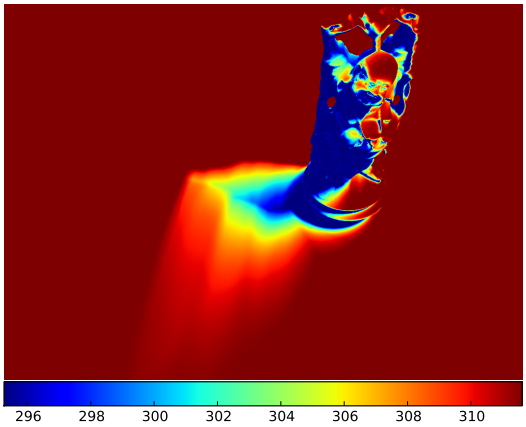


(f) Differenzbild zwischen (c) und (d).

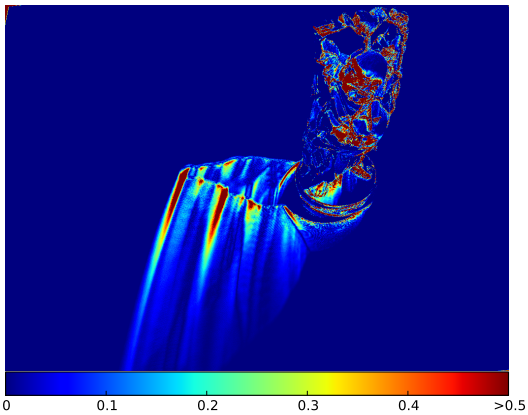
Abbildung 4.19.: Ergebnis in der Buddha-Szene mit verschiedenen Shadow Map Auflösungen.



(a) Ergebnis mit manueller Shadow Map



(b) Ergebnis mit osgShadow Node-Kit



(c) Differenzbild zwischen (a) und (b)

Abbildung 4.20.: Unterschied zwischen manueller Shadow Map Implementierung und osgShadow Node-Kit.

4.4.3. Speicherverbrauch der Grafikkarte

In diesem Abschnitt wird der Speicherverbrauch der Grafikkarte analysiert. Dabei wird nur der Speicherverbrauch der Texturen berücksichtigt. Das Ergebnis wird theoretisch bestimmt und mit AMD CodeXL [5] validiert.

Bei einer Implementierung mit Hilfe des osgShadow Node-Kits wird für jeden thermische Simulation Pass eine Shadow Map erzeugt. Dies kann jedoch nur durch eine Modifizierung des OpenSceneGraph-Quelltextes geändert werden. Tabelle 4.8 zeigt den Speicherverbrauch des Forward und des Deferred Rendering Ansatzes jeweils unter Verwendung von osgShadow und einer eigenen Shadow Map Implementierung. Für den Forward Rendering Ansatz werden drei 1024x768 32-Bit RGBA Float Texturen benötigt. Beim Deferred Rendering Ansatz kommen zusätzlich zwei 1024x768 32-Bit RGBA Float Texturen für den G-Buffer dazu. Für ein Material beträgt die Größe einer Lookup-Tabelle 8 Kilobyte.

	Speicherverbrauch [MB]			
Implementierung	**Shadow Maps**	**Texturen**	**Lookup-Tabelle**	**Gesamt**
Forward(osgShadow)	87	36	0.02	123.02
Forward (Shadow Map)	3	36	0.02	39.02
Deferred (osgShadow)	87	60	0.02	147.02
Deferred (Shadow Map)	3	60	0.02	63.02

Tabelle 4.8.: Speicherverbrauch bei 29 Simulationsschritten für zwei Materialien bei einer Bildschirmauflösung von 1024x768 und einer Shadow Map Auflösung von 1024x1024.

4.5. Diskussion

In diesem Kapitel wurde ein Algorithmus für die echtzeitfähige Simulation von thermischen Schatten aus direkten Strahlungsbeiträgen der Sonne umgesetzt. Der Algorithmus wurde in zwei Ansätzen implementiert, einem Forward und einem Deferred Rendering Ansatz.

Der Forward Rendering Ansatz ist einfach zu implementieren und rendert für jeden Simulationsschritt die Geometrie der 3D-Szene zweimal, einmal aus Sicht der Lichtquelle und einmal aus Sicht des Betrachters. Daher verursacht der Forward Rendering Ansatz eine hohe Last in der Vertex- und Geometrie-Stufe in der Grafikpipeline. Der Deferred Rendering Ansatz dient dazu, diese Last zu reduzieren. Statt dem Rendern der Szene aus Sicht des Betrachters für jeden Simulationsschritt, wird der Schattenfaktor mit Hilfe gespeicherter 3D-Informationen aus einer Textur ermittelt. Allerdings müssen im schlechtesten Fall die 3D-Informationen im jedem Zeitschritt neu ermittelt werden.

Durch den Einsatz einer 3D-Rasterisierung und durch die Implementierung in einem Fragment-Shader werden die Temperaturen der Oberfläche pro Pixel angepasst. Gegenüber polygonbasierten Ansätzen, wie z.B. Radiosity-Verfahren, müssen daher die Polygone nicht an hochfrequenten Bildanteilen fein tesseliert werden.

Die Ergebnisse haben gezeigt, dass der Algorithmus für eine Darstellung von thermischen Schatten mit bis zu 30 Simulationsschritten, bei einer Geometriekomplexität von einer Millionen Dreiecken, echtzeitfähig ist. Simulationen mit mehr Schritten sind durch eine exponentielle Abtastung des Simulationszeitraums oder durch eine Mehrfachbenutzung von Shadow Maps in Echtzeit durchführbar. Der Vergleich der relativen und absoluten Oberflächentemperaturen gegenüber einer RadTherm-Lösung hat gezeigt, dass unser Ansatz akzeptable Ergebnisse liefert. Sobald allerdings Wärmetransportmechanismen überwiegen, die durch unseren Ansatz nicht erfasst werden, entstehen große Abweichungen. Dies ist beispielsweise bei einer Wärmeleitung innerhalb von Körpern der Fall. Zusätzlich scheitert in diesem Falle oft der PCF Algorithmus mit den gewählten, konstanten 32 Abtastpunkten, da der Halbschatten sehr groß skaliert werden müsste. Dabei kommt es allerdings zu starken Block-Artefakten. Daher wurde die Halbschattenbreite, bei lateraler Wärmeleitung, oft bewusst zu klein gewählt, um diese Artefakte zu verhindern.

Der Algorithmus besitzt einige weitere Einschränkungen. Die Haupteinschränkung besteht in der Annahme, dass nur die Sonne einen thermischen Schatten verursacht. Das heißt, dass nur eine Wärmequelle in der Szene unterstützt wird. Die Einschränkung resultiert aus den vorberechneten Strahlungs- und Schattentemperaturen einer Oberfläche. Im Falle von mehreren Lichtquellen müsste die Ausgleichstemperatur dynamisch in jedem Zeitschritt angepasst werden. Das gleiche gilt für Oberflächen mit einer anderen Orientierung. Dadurch muss jede Orientierung als neues Material aufgefasst werden. Bei komplexer Geometrie, wie z.B. der Buddha-Statue, ist dies jedoch kaum möglich. Diese Einschränkungen können gelöst werden, indem direkt die Strahlungsbeiträge betrachtet und aus diesen die Ausgleichstemperatur in jedem Zeitschritt erneut bestimmt wird. Im Kapitel 6 wird dies umgesetzt.

Durch den Einsatz von Shadow Mapping Verfahren ist die Darstellung der thermischen Schatten blickwinkelabhängig, da die Temperaturen für jeden sichtbaren Pixel angepasst werden. Diese Eigenschaft führt jedoch nur zu Einschränkungen bei zustandserhaltenden Animationen und bei direkter Manipulation von Objekten und Oberflächen durch einen Benutzer. Zustandserhaltenden Animationen, wie z.B. deformbare Körper, erfordern die Speicherung der Vertex-Position zu jedem Simulationszeitschritt, was zu einem hohen Rechen- und Speicheraufwand führt. Des Weiteren sind die Positionen nicht für beliebige Zeitschritte ermittelbar. Bei zustandslosen Animationen hingegen, wie z.B. Keyframe-Animationen, kann die Position über eine Funktion ermittelt und für beliebige Simulationszeitpunkte angepasst werden. Bei einer direkten Manipulation von Objekten und Oberflächen, wie z.B. einen manuellen Temperatureintrag auf einer Oberfläche, führt dies zu einem analogen Problem. Diese Zustände müssten zeitlich abgespeichert werden. Einen Lösungsvorschlag für diese Einschränkung wird im Kapitel 7 gegeben.

5. Verfahren zur Beschleunigung der Schattenberechnung

In diesem Kapitel werden Ansätze zur Beschleunigung der Schattenberechnung vorgestellt. Der erste Ansatz behandelt ein Clustering von Lichtquellen, um die Anzahl der Shadow Maps für viele Lichtquellen zu reduzieren. Im zweiten Ansatz wird ein Algorithmus vorgestellt, der Schatten mit variabler Halbschattenbreite über einen Erosionsoperator erzeugt.

5.1. Clustering von Lichtquellen

Shadow Mapping [100] ist ein populäres Verfahren in der Echtzeit-Computergrafik um Schatten zu rendern. Bei Anwendungen mit vielen Lichtquellen, wie z.B. thermische Schatten, muss allerdings für jede Lichtquelle der Szene eine Shadow Map erzeugt werden. Dies kann sowohl beim Speicherverbrauch als auch bei der Berechnungszeit zu einem Flaschenhals führen.

Um die Berechnung von Schatten für viele Lichtquellen zu beschleunigen, besteht unsere Idee darin, ähnliche Lichtquellen in Cluster zusammenzufassen und nur für den Cluster einen Schatten zu erzeugen. Um Fehler in der Schattendarstellung zu reduzieren, wird jeder Cluster als eine ausgedehnte Lichtquelle interpretiert.

Die Vorgehensweise kann in drei Schritten zusammengefasst werden. Im ersten Schritt werden Lichtquellen zu Cluster zusammengefasst und eine Lichtquelle als Repräsentant für jeden Cluster ausgewählt. Im zweiten Schritt wird ein weicher Schatten für die Repräsentanten gerendert und das Ergebnis in eine Sichtbarkeitstextur abgespeichert. Abschließend wird bei der Beleuchtungsrechnung der Schatten mit Hilfe der Sichtbarkeitstextur berücksichtigt.

Unsere Beiträge sind:

- Eine Clustering-Strategie für Punktlichtquellen, die eine variable Anzahl an existierenden Lichtquellen als Repräsentanten auswählt.
- Minimum-Distanz Metriken um die Anzahl der Shadow Maps für verschiedene Lichtquellenarten zu reduzieren und um zwischen der Qualität und der Geschwindigkeit abzuwägen.
- Eine Approximation eines Schattens mehrerer Punktlichtquellen durch eine ausgedehnte Lichtquelle wobei die Fläche der ausgedehnten Lichtquelle von der Distanz zwischen den Clustern abhängt.

5.1.1. Stand der Technik

In der Literatur gibt es eine Vielzahl von Verfahren für die Beleuchtungsrechnung mit vielen Lichtquellen. Im Folgenden werden nur die, für diese Arbeit, relevanten Verfahren vorgestellt. Für eine vollständige Diskussion anderer Verfahren wird auf Dachsbacher et al. [21] verwiesen.

Instant Radiosity [47] approximiert eine globale Beleuchtung durch die lokale Beleuchtung einer Menge von virtuellen Punktlichtquellen (VPL). Für die Erzeugung von indirekte Schatten wird eine Shadow Map für jede VPL gerendert. Um die VPLs mit Hilfe von Texturen zu verteilen, werden Reflective Shadow Maps (RSMs) [22] von der Position der Lichtquelle aus gerendert.

Weitere Ansätze beschleunigen die Berechnung von Instant Radiosity. Laine et al. [50] führen ein inkrementelles Update einer festen Menge an VPLs und Shadow Maps durch. Dazu werden die VPLs auf ein Voronoi-Diagramm aufgetragen und die VPLs entfernt, die den kleinsten Abstand zueinander haben. Die entfernten VPLs werden anschließend neu platziert und gerendert. Ritschel et al. [80, 82] beschleunigen die Berechnung der indirekten Beleuchtung mit VPLs durch das Rendern von niedrig aufgelösten Shadow Maps. Dazu wird eine reduzierte, punktbasierte Darstellung der Szene verwendet.

Walter et al. [97] erstellen einen Binärbaum für alle Lichtquellen in einer Szene. Ein innerer Knoten ist dabei ein Repräsentant für die jeweiligen Blattknoten. Für jedes Bild wird der Binärbaum traversiert und ein Schnitt bestimmt. Dieser Schnitt legt die relevanten Lichtquellen und Repräsentanten über visuelle Kriterien fest, wie z.B. Materialbeschaffenheiten oder geometrische Verhältnisse.

Hašan et al. [39] interpretieren den Zusammenhang zwischen m Oberflächen und n Lichtquellen als eine $m \cdot n$ Matrix und benutzen nur eine kleine Anzahl an Zeilen und Spalten für die Beleuchtungsrechnung. Um ein Flackern bei einer Animation von Lichtquellen zu reduzieren, kann die Matrix zu einem Tensor erweitert werden [40].

Olsson et al. [67] benutzen Clustered Shading um das Rendern von Schatten für viele Lichtquellen zu beschleunigen. Nach dem Ablauf des Clustered Shading Algorithmus liegen für jeden Cluster eine Liste von Lichtquellen vor. Olsson et al. berechnen nun für jede Lichtquelle eine geeignete Shadow Map Auflösung und allokieren dynamisch, über Virtual-Texture Mapping, eine Cube-Shadow Map. Für jede Lichtquelle wird anschließend ein harter Schatten gerendert.

Clustered Visibility [25] beschleunigt die Berechnung von indirekten Schatten durch ein k-means Clustering von VPLs einer RSM. Jeder Cluster wird dabei als eine ausgedehnte Lichtquelle interpretiert, um den Sichtbarkeitstest pro Cluster durchzuführen. Diese Vorgehensweise ähnelt stark unserer Methode. Allerdings ist unsere Methode auf die hochfrequenten Schatten einer direkten Beleuchtung von Lichtquellen ausgelegt. Die Lichtquellen werden dabei frei im Raum platziert und sind nicht durch die Platzierung an einer Geometrieoberfläche durch die RSM beschränkt. Die Auswahl eines

Repräsentanten eines Clusters über die Schwerpunktssuche, wie z.B. k-means Clustering, führt dadurch zu Fehlern, da sich der Schwerpunkt innerhalb einer Geometrie befinden kann. Unser Ansatz benutzt stattdessen eine Minimum-Distanz Metrik für das Clustering und benutzt eine existierende Lichtquelle als Repräsentanten.

5.1.2. Clustering

Für das Clustering benutzen wir ein Minimum-Distanz Clustering [102]. Dabei werden alle Objekte innerhalb einer minimalen Distanz zu Clustern zusammengefasst. Das Clustering kann dabei im Top-Down Prinzip durchgeführt werden. Zuerst wird angenommen, dass sich alle Objekte innerhalb eines einzigen Clusters befinden. Der Algorithmus testet anschließend alle Objekte auf ihre Distanz und erzeugt einen neuen Cluster, wenn die Distanz größer als die Minimum-Distanz ist. Dieser Schritt wird solange wiederholt bis keine Objekte in neue Cluster unterteilt werden können.

Um die Lichtquellen zu einem Cluster zusammenzufassen, müssen zunächst Ähnlichkeitsmetriken bzw. Distanzen ermittelt werden. Dabei wird zwischen den Lichtquellenarten unterschieden (Abbildung 5.1).

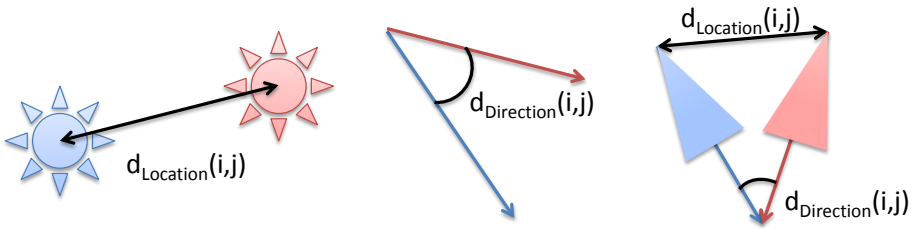


Abbildung 5.1.: Cluster Metriken für lokale Punktlichtquellen, unendlich entfernte Richtungslichtquellen und lokale Richtungslichtquellen.

Für lokale Punktlichtquellen ist das Hauptmerkmal für eine Ähnlichkeit eines Schattens die Position der Lichtquelle. Nur Lichtquellen können zusammengefasst werden, die ähnliche Anteile einer Szene ausleuchten und somit eine Überschneidung von Schattenspendern und Schattenempfängern vorhanden ist. Für zwei lokale Punktlichtquellen i und j wird folgende Distanz definiert:

$$d_{Location}(i,j) = \sqrt{(x_i - x_j)^2}$$

Mit der Position x der Lichtquelle.

Bei unendlich entfernten Richtungslichtquellen hingegen ist die Strahlungsrichtung das entscheidende Merkmal. Ähnliche Strahlungsrichtungen resultieren in ähnlichen Schattenspendern und Schattenempfängern. Die Distanz ist gegeben durch:

$$d_{Direction}(i,j) = (1 - \vec{x}_i \cdot \vec{x}_j)$$

Mit den normierten Richtungsvektoren $\overrightarrow{x}$ der Lichtquelle.

Für eine lokale Richtungslichtquelle sind sowohl die Position als auch die Richtung ein Merkmal für einen Schatten. Die Distanz ist daher das Produkt der örtlichen Informationen und der Strahlungsrichtung:

$$d_{Spot}(i,j) = d_{Point}(i,j) \cdot d_{Directional}(i,j)$$

Beim Clustering werden die Lichtquellenarten aufgrund ihrer Eigenschaften separat geclustert und Schatten durch ein jeweilig angepasstes Shadow Mapping Verfahren realisiert (siehe Abschnitt 5.1.3.2). Im Folgenden wird das Clustering nur für lokale Punktlichtquellen gezeigt und an den Stellen eine Anmerkung hinzugefügt, bei denen eine Anpassung für eine Richtungslichtquelle erforderlich ist.

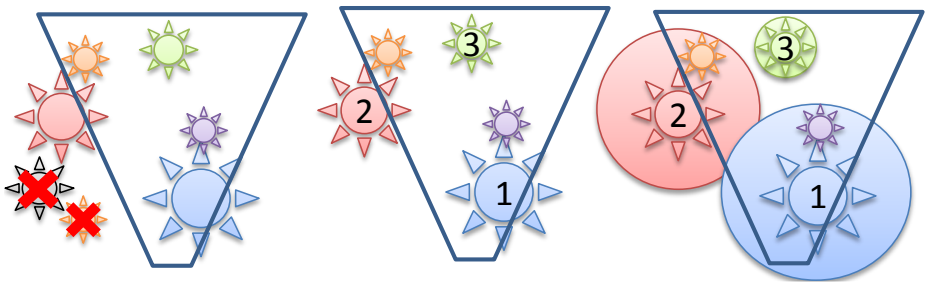


Abbildung 5.2.: Ablauf des Clusterings. Zuerst werden Lichtquellen entfernt, die keinen Einfluss auf das View-Frustum haben. Anschließend werden Repräsentanten ermittelt und die Lichtquellen den Repräsentanten zugeordnet.

In Abbildung 5.2 wird der Ablauf des Clusterings dargestellt. Zuerst werden Lichtquellen ausgeschlossen, die keinen Einfluss auf das View-Frustum haben. Anschließend werden Repräsentanten ermittelt und die Lichtquellen zugeordnet.

5.1.2.1. Auswahl der Repräsentanten

Im ersten Schritt des Clusterings werden die Repräsentanten festgelegt. Dabei werden zunächst alle Lichtquellen auf einen Einfluss auf das View-Frustum getestet und diese ausgeschlossen, die keinen Einfluss ausüben. Lokale Punktlichtquellen werden dabei als Bounding-Sphere, mit dem Radius der Beleuchtungsreichweite, interpretiert und gegen das View-Frustum getestet. Bei lokalen Richtungslichtquellen wird ein Kegel gegen das View-Frustum getestet. Nur unendlich entfernte Richtungslichtquellen können nicht ausgeschlossen werden, da potentiell alle Pixel im View-Frustum beleuchtet werden.

Die Lichtquellen werden anschließend anhand ihrer Beleuchtungsstärke sortiert. Dadurch testet der Algorithmus zuerst die Lichtquellen als Repräsentanten, die die höchste Beleuchtungsstärke besitzen. Nach dem die Lichtquellen sortiert sind, werden die Repräsentanten gesucht. Dabei wird angenommen, dass sich alle Lichtquellen in einen Cluster befinden und das erste Licht der erste Repräsentant ist. Für jedes Licht wird

anschließend die Distanz, mit der oben vorgestellten Metrik, berechnet und mit der minimalen Distanz verglichen. Wenn die Distanz zum Repräsentanten größer als die minimale Distanz ist, wird das Licht als weiterer Repräsentant ausgewählt. Der Vorgang wird solange wiederholt bis keine neuen Repräsentanten gefunden werden.

Bei lokalen Punkt- und Richtungslichtquellen ist es sinnvoll, mehr Repräsentanten in der Nähe der Kamera zu generieren, da dort, durch eine perspektivische Projektion, der Einfluss auf das Bild im Allgemeinen am größten ist. Um mehr Repräsentanten in der Nähe der Kameraposition zu erzeugen wird die minimale Distanz anhand der Kameradistanz skaliert. Die minimale Distanz wird wie folgt berechnet:

$$d_{min} = d_{init} \left(1 - \frac{\|R\|}{(z_{far} - z_{near})}\right)$$

Wobei d_{init} die anfängliche, vom Benutzer gewählte, Distanz ist. $\|R\|$ ist die Position des Repräsentanten im Kameraraum, z_{near} die Entfernung der Kamera Near-Plane und z_{far} die Entfernung der Kamera Far-Plane.

5.1.2.2. Lichtquellen den Repräsentanten zuordnen

Im nächsten Schritt werden die Lichtquellen den Repräsentanten zugeordnet. Dazu wird über alle verbleibenden Lichtquellen iteriert und die minimale Distanz zu den Repräsentanten berechnet. Die Lichtquelle wird dem Repräsentanten zugeordnet, bei dem die Distanz am geringsten ist.

Für lokale Punktlichtquellen wird bei diesem Schritt zusätzlich die Beleuchtungsreichweite aller Lichtquellen maximiert und dem Repräsentanten zugeordnet. Analog wird bei lokalen Richtungslichtquellen die Beleuchtungskegel zu einem maximalen Beleuchtungskegel zusammengefasst.

5.1.3. Rendern der Schatten

Anstatt für jede Lichtquelle einen Schatten zu erzeugen, wird der Schatten nur für die kleinere Anzahl an Repräsentanten erzeugt. Um den Fehler im Schatten zu reduzieren, wird jeder Repräsentant als Flächenlichtquelle interpretiert und ein Schattenfaktor mit Hilfe eines Algorithmus für weiche Schatten erzeugt.

5.1.3.1. Direkte Beleuchtung mit Repräsentanten

Nach der Notation von Eisemann et al. [28] kann die ausgehende Strahlung L_o, für eine direkte Beleuchtung von n Lichtquellen, vereinfacht wie folgt definiert werden:

$$L_o(p, \omega) = \sum_{i=0}^{n} L_d(p, \omega, l_i) \cdot L_{c_i} \cdot V(p, l_i)$$

Mit

$$L_d(p, \omega, l_i) = f_r(p, \omega, p \rightarrow l_i) G(p, l_i)$$

Wobei p ein Punkt auf der Oberfläche, ω eine Richtung, f_r eine bidirektionale Reflektanzverteilungsfunktion (BRDF), V eine Sichtbarkeitsfunktion, G der geometrische Term, L_{c_i} die Farbe und l_i die Position der i-ten Lichtquelle ist.

Durch das Clustering wird der binäre Sichtbarkeitstest V durch einen Sichtbarkeitstest einer Flächenlichtquelle A_c für einen Repräsentanten c ersetzt:

$$L_o(p, \omega) \approx \sum_{i=0}^{n} L_d(p, \omega, l_i) \cdot L_{c_i} \cdot \frac{1}{A_c} \int_{A_c} V(p, l) dl$$

5.1.3.2. Rendern der Schatten

Um die Schatten zu rendern wird Shadow Mapping eingesetzt. Dabei wird anhand der Lichtquellenart unterschieden. Für eine unendlich entfernte und eine lokale Richtungslichtquellen wird eine Shadow Map erzeugt. Der Schatten für eine lokale Punktlichtquelle wird hingegen mit einer Shadow CubeMap gerendert.

Nach dem die Shadow Map gerendert ist, wird für jeden Cluster ein Schattenfaktor bestimmt. Ein Cluster wird dabei als scheibenförmige Flächenlichtquelle interpretiert, mit der Minimum-Distanz als Radius und dem Repräsentanten als Zentrum. Dieser Radius wird benutzt um das Filterfenster von PCSS [30] und PCF [79] zu skalieren. Für lokale Punkt- und Richtungslichtquellen wird der Abschwächungsfaktor bzw. der Beleuchtungskegel der Lichtquelle, dazu benutzt um die Schattenberechnung eines Pixels vorzeitig zu terminieren.

Der Schattenfaktor wird anschließend in einen 2D-Texture Array gespeichert. Dies erlaubt es, die Schatten für jeden Cluster getrennt von der Beleuchtung zu rendern. Zusätzlich reduziert es den Speicherbedarf bei der Beleuchtung von einer Shadow Map je Cluster zu einer Textur mit Bildschirmauflösung. Eine Reduzierung des Speicherbedarfs ist jedoch abhängig von der Lichtquellenart, der Shadow Map Größe und der Bildschirmauflösung. Für lokale Punktlichtquellen wird dies allerdings deutlich, wenn eine Shadow CubeMap durch eine 2D-Textur ersetzt wird.

5.1.3.3. Anwendung auf die Beleuchtung

Nachdem die Schatten für die Cluster bestimmt wurden, werden die Schatten bei der Beleuchtungsrechnung angewendet. Dazu wird die Szene für jede Lichtquelle beleuchtet und die resultierende Farbe mit dem Schattenfaktor des zugehörigen Clusters moduliert.

5.1.4. Implementierung

Der Algorithmus wird mit Tiled Deferred Shading [53] umgesetzt. Allerdings kann das Verfahren genauso mit anderen Techniken für die Beleuchtung von vielen Lichtquellen, wie z.B. Tiled Forward Shading [66], eingesetzt werden. In jedem Frame wird das Clustering durchgeführt, der G-Buffer gerendert, die Schatten je Cluster berechnet und die Beleuchtungsrechnung durchgeführt.

Der Minimum-Distanz Cluster-Algorithmus wird vollständig auf der CPU implementiert, da es nur einen geringen Einfluss auf die Gesamtgeschwindigkeit hat. Die Position des Repräsentanten, der Abschwächungsfaktor des Clusters und ein Cluster-Index pro Lichtquelle werden in einem Unordered-Access-View auf der GPU abgelegt.

Die Shadow CubeMap für lokale Punktlichtquellen wird in einem einzelnen Renderpass erzeugt. Dazu wird ein Geometry-Shader eingesetzt, um die Geometrie für jede Raumrichtung zu replizieren. In der Shadow CubeMap wird die Distanz in Weltkoordinaten gespeichert. Während des Schattentests wird sie mit den rekonstruierten Weltkoordinanten aus den Tiefeninformationen des G-Buffers verglichen.

5.1.5. Ergebnisse

Für die Auswertung der Ergebnisse werden eine Bildschirmauflösung von 1920x1080 und eine Shadow-CubeMap mit der Auflösung von 1024x1024 je Raumrichtung verwendet. Für die Schattenberechnung wird eine Poisson-Disk mit 8 Abtastpunkten, bei der Blockersuche und PCF, benutzt. Die Geschwindigkeit wird auf einen Intel Xeon E5620 Prozessor mit 2,4 GHz, 8 GB RAM und einer NVIDIA GeForce GTX 680 mit 2048 MB Speicher ermittelt.

5.1.5.1. Szenen

Für die Auswertung werden drei Szenen ausgewählt. In der Sponza Szene werden 80 Lichtquellen in Form von Kreisen hinter jeder Säule platziert (Abbildung 5.5a). In der Cornell-Box Szene werden 32 Lichtquellen zufällig verteilt (Abbildung 5.6a). Die letzte Szene zeigt einen Ausschnitt aus einem Restaurant mit zwei Tischen und einem Kerzenleuchter. Insgesamt werden 14 Lichtquellen platziert, eine an jeder Kerze und Lampe, sowie eine an jeder Lampe des Kerzenleuchters (Abbildung 5.7a).

5.1.5.2. Referenzlösung

Als Referenzlösung wird für jede Lichtquelle ein Schatten mit Hilfe von Shadow Mapping berechnet. Da die Schatten für lokale Punktlichtquellen berechnet werden, wird ein harter Schatten erzeugt. Das bedeutet, dass für die Sponza Szene 80 Shadow CubeMaps (Abbildung 5.5b), für die Cornell-Box 32 Shadow CubeMaps (Abbildung

5.6b) und für die Restaurantszene 14 Shadow CubeMaps (Abbildung 5.7b) gerendert werden.

5.1.5.3. Speicherverbrauch

Unser Algorithmus benötigt für jedem Cluster eine zusätzliche 32 Bit Float 2D-Textur in Bildschirmauflösung. Daher beträgt der Speicherverbrauch die Anzahl der Cluster multipliziert mit der Bildschirmauflösung. Für die Sponza Szene mit 26 Clustern wird beispielsweise 205 MB Speicher zusätzlich für die Beleuchtungsrechnung verbraucht. Wenn allerdings die Schatten während der Beleuchtungsrechnung berechnet werden, müssen 26 Shadow CubeMaps mit 624 MB im Speicher gehalten werden.

5.1.5.4. Geschwindigkeit

Die Geschwindigkeit wird gegen die Referenzlösung verglichen. In allen Testszenen wird eine maximale View-Distanz von $z_{far} = 30$ benutzt und nur die anfängliche Distanz d_{init} variiert.

Sponza		**Zeit [ms]**		**Cornell-Box**		**Zeit [ms]**	
d_{init}	Clusters	PCF	PCSS	d_{init}	Clusters	PCF	PCSS
0.1	80	100.1	115.3	0.3	13	12.9	18.0
0.3	44	57.0	67.3	0.4	6	9.8	12.9
0.5	26	36.4	43.1	0.5	5	9.1	12.1
1.6	10	18.7	20.7	0.6	4	8.6	11.1
Referenzlösung		97.8		Referenzlösung		20.0	

Restaurant		**Zeit [ms]**	
d_{init}	Clusters	PCF	PCSS
0.2	12	36.0	40.1
0.3	9	28.9	32.9
0.4	8	26.8	30.5
0.5	7	24.9	27.8
Referenzlösung		39.3	

Tabelle 5.1.: Geschwindigkeit im Vergleich zur Referenzlösung. Die anfängliche Distanz d_{init} wird jeweils variiert.

Tabelle 5.1 stellt die Geschwindigkeit im Vergleich zur Referenzlösung dar. Unser Algorithmus ist schneller als die Referenzlösung wenn Lichtquellen zu Cluster zusammengefasst werden können. Je größer die anfängliche Distanz d_{init} gewählt wird, desto mehr Lichtquellen werden zu Cluster zusammengefasst und desto weniger Schatten müssen berechnet werden. Daher bestimmt der Faktor d_{init} maßgeblich die Qualität und Geschwindigkeit unseres Algorithmus. Der Worst-Case für unseren Algorithmus besteht darin, wenn keine Lichtquellen zu einem Cluster zusammengefasst werden. Das

kann genau dann auftreten, wenn d_{init} zu klein gewählt wird. Aufgrund des zusätzlichen Cluster-Schritts und der Berechnung von PCSS bzw. PCF ist die Geschwindigkeit langsamer als bei der Referenzlösung. Allerdings kann dieser Fall einfach umgangen werden, wenn die Anzahl der Cluster mit der Anzahl der Lichtquellen verglichen und ein anderer Renderpfad gewählt wird.

Sponza	**Zeit [ms]**		**Prozent**	
Schritt	PCF	PCSS	PCF	PCSS
Clustering	0.05	0.05	<0.1%	<0.1%
G-Buffer	0.53	0.53	0.9%	0.8%
Schatten	49.20	59.42	86.3%	88.4%
Beleuchtung	7.05	7.05	12.3%	10.4%
Darstellung	0.20	0.20	0.4%	0.3%
Gesamt	57.03	67.25	100%	100%

Tabelle 5.2.: Dauer der einzelnen Schritte im Algorithmus bei der Sponza Szene mit 44 Clustern.

Die Dauer der einzelnen Schritte des Algorithmus wird in Tabelle 5.1, für die Sponza Szene mit 44 Clustern, dargestellt. Es kann beobachtet werden, dass der Flaschenhals des Algorithmus die Schattenberechnung ist. Das Clustering macht hingegen nur etwa 0.1% der gesamten Geschwindigkeit aus.

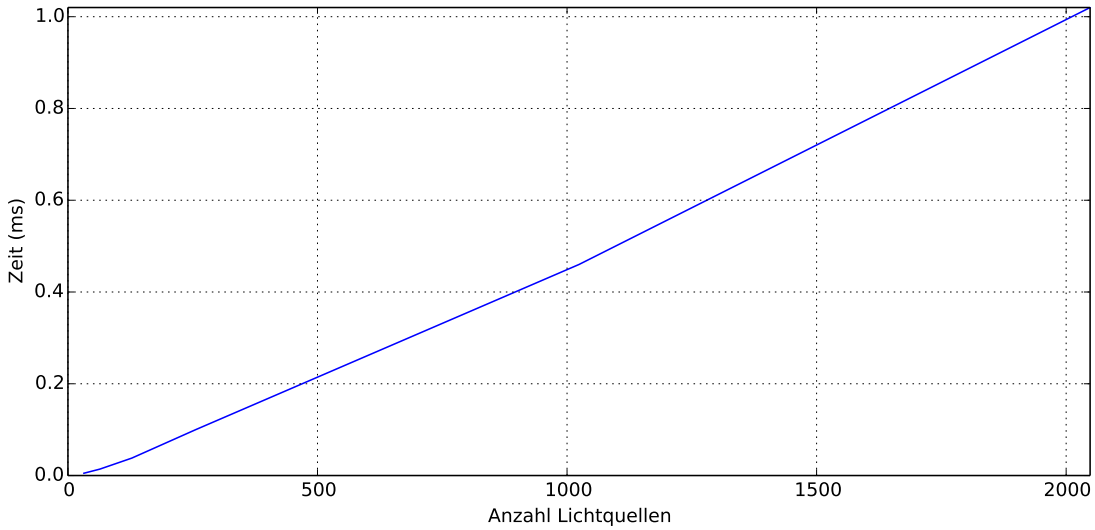


Abbildung 5.3.: Dauer des Clusterings in Abhängigkeit von der Anzahl der Lichtquellen.

Abbildung 5.3 zeigt die Geschwindigkeit des Clusterings, wenn die Anzahl der Lichtquellen variiert wird. Der Aufwand des Clusterns steigt linear mit der Anzahl der Lichtquellen an. Allerdings dauert das Clustern bei 2048 Lichtquellen nur einer Millisekunde.

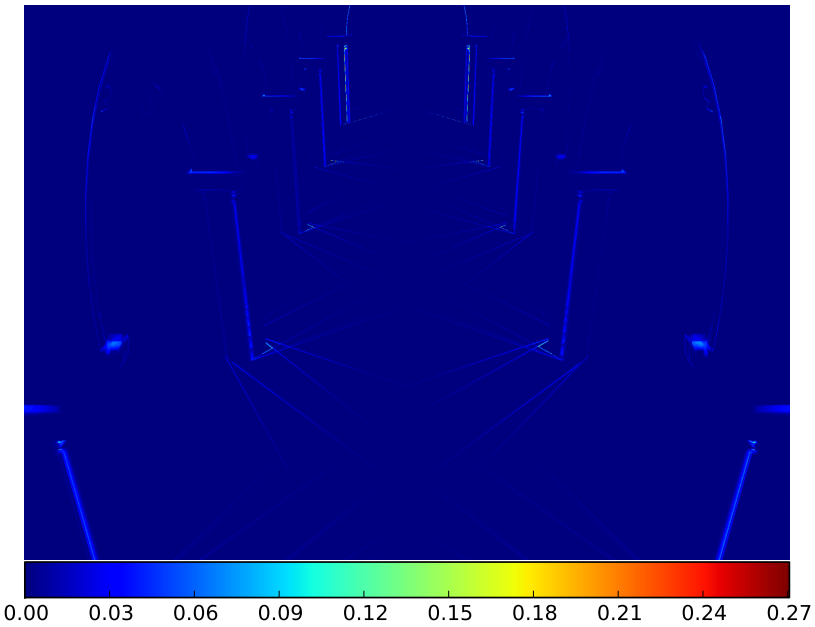


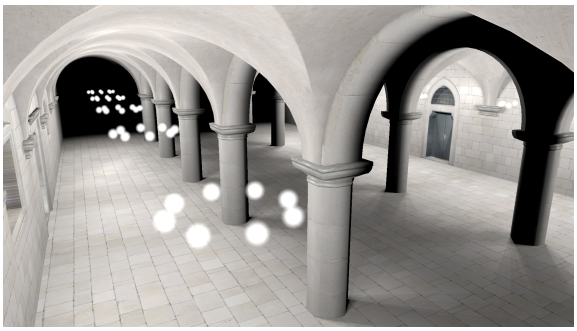
Abbildung 5.4.: Vergleich der Schatten zwischen PCSS und PCF.

5.1.5.5. Fehlerabschätzung

Um den Fehler im Vergleich zur Referenzlösung abzuschätzen, werden Differenzbilder benutzt. Eine dunkelblaue Farbe zeigt einen kleinen Fehler und eine rote Farbe einen großen Fehler an.

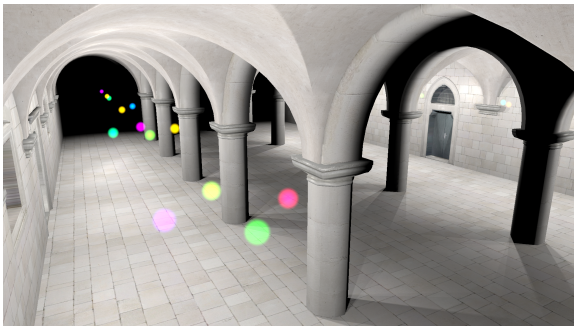
Abbildung 5.4 zeigt das Differenzbild zwischen einer Schattendarstellung mit PCSS und PCF. Der größte Unterschied kann in den Bereichen beobachtet werden, bei denen eine variable Halbschattenbreite vorhanden ist, beispielsweise an den Säulen. Daher erzeugt PCSS einen realistischeren Eindruck der Schatten. Allerdings kann PCF dazu benutzt werden, die Geschwindigkeit der Schattenberechnung zu erhöhen. Für die weiteren Vergleiche wird PCSS eingesetzt.

Abbildung 5.5 zeigt das Ergebnis unseres Algorithmus bei der Sponza Szene. Abbildung 5.5a und 5.5b zeigt die Verteilung der Lichtquellen und das Ergebnis der Referenzlösung. Für den ersten Vergleich wird eine anfängliche Distanz von $d_{init} = 0.5$ benutzt, welche in 26 Clustern resultiert. Die Platzierung der Cluster wird in Abbildung 5.5c dargestellt. Im Differenzbild 5.5g wird deutlich, dass ein kleiner Fehler bei den Schattenkanten und ein größerer Fehler im hinteren Bereich der Szene entsteht. Im hinteren Bereich der Szene werden durch die Distanz-Skalierung weniger Cluster platziert. Im zweiten Fall wird eine anfängliche Distanz von $d_{init} = 1.6$ benutzt, welche in 10 Clustern resultiert. Es werden also jeweils acht Lichtquellen durch einen Repräsentanten ersetzt. Die auftretenden Fehler sind dabei im Ergebnisbild 5.5f und im Differenzbild 5.5h klar sichtbar. Allerdings bleibt der allgemeine Eindruck des Schattens in der Szene vorhanden und die Geschwindigkeit konnte um einen Faktor von 4.8 erhöht werden.

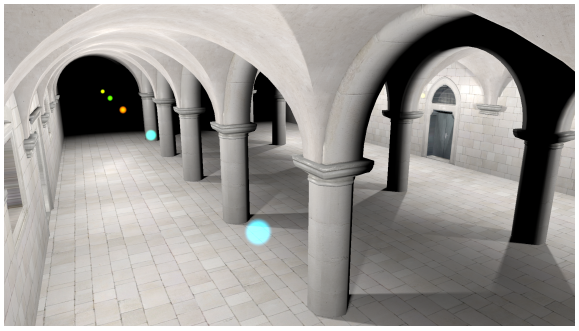


(a) Verteilung der 80 Lichtquellen.

(b) Referenzlösung mit 97.8 ms.



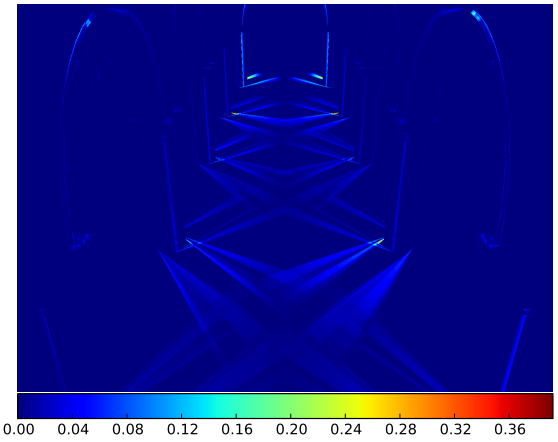
(c) Clustering für $d_{init} = 0.5$ mit 26 Cluster.



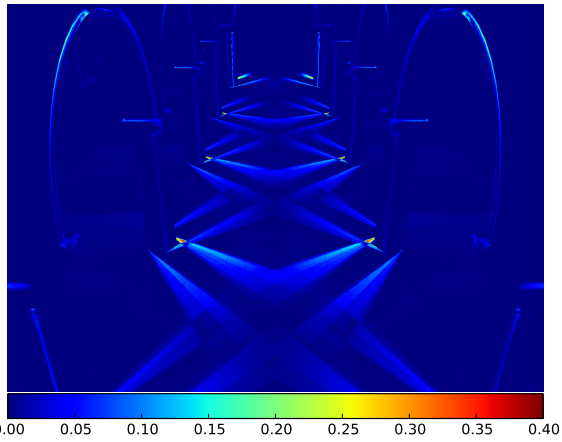
(d) Clustering für $d_{init} = 1.6$ mit 10 Cluster.

(e) Schatten für 26 Cluster bei 43.1 ms.

(f) Schatten für 10 Cluster bei 20.7 ms.



(g) Differenzbild von (b) und (e)



(h) Differenzbild von (b) und (f).

Abbildung 5.5.: Vergleich bei der Sponza Szene.

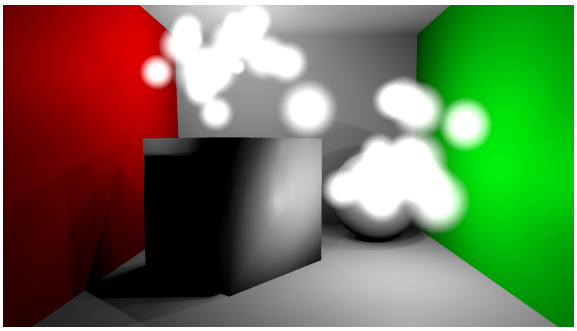
Die Ergebnisse der Cornell-Box Szene mit zufällig verteilten Punktlichtquellen werden im Abbildung 5.6 dargestellt. Die Verteilung der Lichter wird in Abbildung 5.6a und das Ergebnis der Referenzlösung in Abbildung 5.6b dargestellt. Im ersten Fall wird eine anfängliche Distanz von $d_{init} = 0.5$ benutzt, welche die 32 Lichtquellen zu fünf Clustern zusammenfasst. Dadurch entstehen Fehler bei den Schatten, die durch den linken Würfen und der Kugel geworfen werden (Abbildung 5.6e und 5.6g). Wenn die anfängliche Distanz weiter auf $d_{init} = 0.6$ erhöht wird, werden vier Cluster erzeugt. Dadurch wird der Fehler an der Wand hinter der Kugel erhöht (Abbildung 5.6f und 5.6h).

Bei der Restaurant Szene werden 14 Lichtquellen verteilt (Abbildung 5.7a). Das Ergebnis der Referenzlösung wird in Abbildung 5.7b dargestellt. Durch die räumliche Verteilung der Lichtquellen in der Szene können nur die Lichtquellen der Kerze und am Kronleuchter geclustert werden. In Abbildung 5.7e werden die Lichtquellen des Kronleuchters mit drei Clustern repräsentiert ($d_{init} = 0.3$). Das Differenzbild zeigt, dass dadurch ein Fehler am Schatten des Tisches entsteht. Wenn die anfängliche Distanz weiter auf $d_{init} = 0.4$ erhöht wird, werden die Lichtquellen des Kronleuchters durch zwei Repräsentanten repräsentiert. Dadurch steigt der Fehler weiter an.

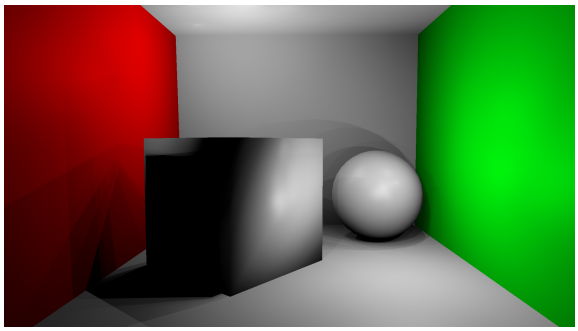
In Abbildung 5.8 wird die Wurzel des mittleren quadratischen Fehlers (RMSE) unseres Algorithmus, mit unterschiedlichen anfänglichen Distanzen, im Vergleich zur Referenzlösung dargestellt. Der Fehler steigt an, wenn sich die Anzahl der Cluster und somit die Anzahl der Schattenberechnungen ändert. Dies ist z.B. bei der Restaurant-Szene bei einem Anstieg der Distanz von 0.4 zu 0.5 ersichtlich. Andererseits ist der Fehler fast konstant, wenn die Lichtquellen anderen Clustern zugewiesen werden. Das ist dann der Fall, wenn die Distanz erhöht wird, aber keine neuen Cluster erzeugt werden. Dieses Verhalten ist beispielsweise in der Cornell-Box Szene bei einer Distanz zwischen 0.4 und 0.7 sichtbar.

5.1.6. Diskussion

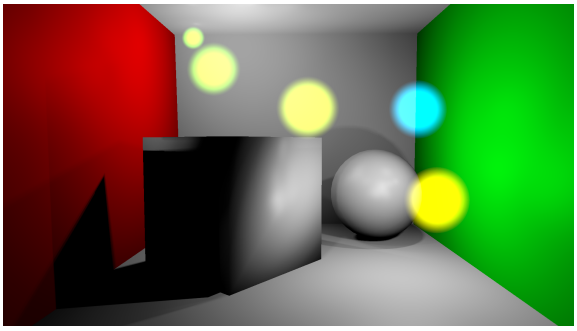
Wir benutzen einen Minimum-Distanz Clustering Algorithmus um ähnliche Lichtquellen zu Clustern zusammenzufassen. Im Vergleich zu einem k-means Clustering, besitzt ein Minimum-Distanz Clustering für diesen Einsatzzweck folgende Vorteile: Die Anzahl der Cluster muss nicht im Vorfeld bekannt sein. Dadurch muss keine Heuristik für eine Abschätzung der Anzahl der Cluster integriert werden, was das Clustering vereinfacht. Ein weiterer Vorteil besteht darin, dass der Repräsentant immer eine vorhanden Lichtquelle in der Szene ist. Dadurch ist der Schatten für eine Lichtquelle des Clusters immer korrekt. Hingegen wird beim k-means Clustering der Schwerpunkt des Clusters als Repräsentant benutzt. Bei lokalen Punktlichtquellen kann es daher vorkommen, dass der Repräsentant durch eine Geometrie verdeckt wird, z.B. sich dieser innerhalb einer Wand befindet. Dong et al. [25] löst dieses Problem, indem die Normale als zusätzliches Cluster-Kriterium herangezogen wird. Allerdings ist das bei unseren Ansatz nicht möglich, da die Lichtquellen frei im Raum platziert werden. Um dieses



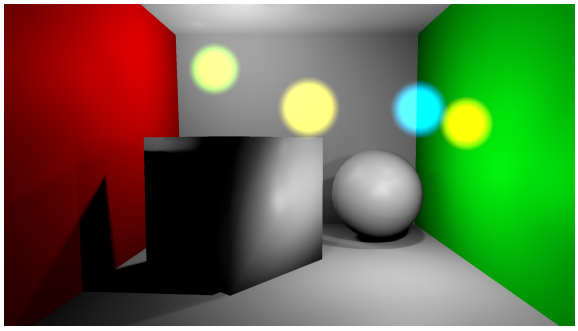
(a) Verteilung der 32 Lichter.



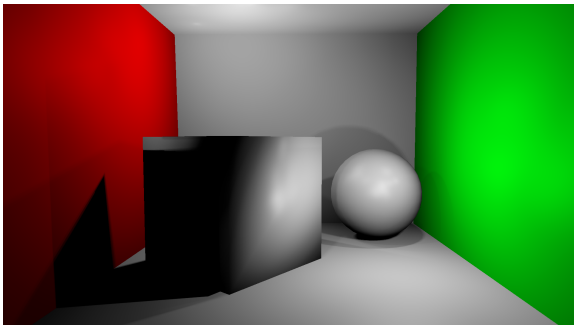
(b) Referenzlösung mit 20.0 ms.



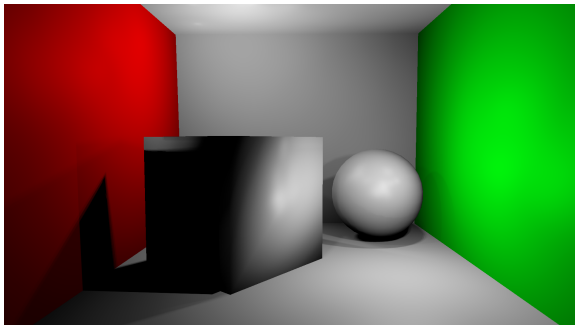
(c) Clustering für $d_{init} = 0.5$ mit 5 Clustern.



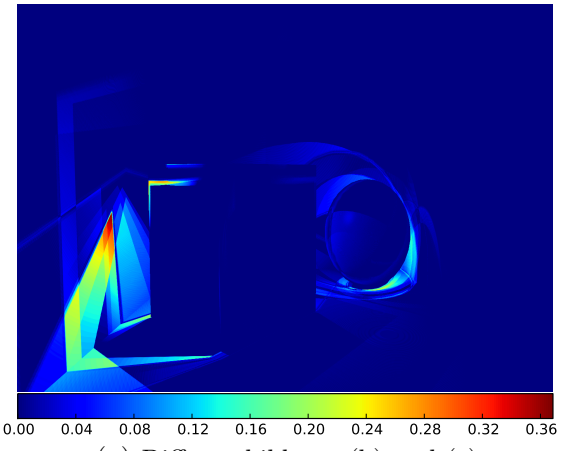
(d) Clustering für $d_{init} = 0.6$ mit 4 Clustern.



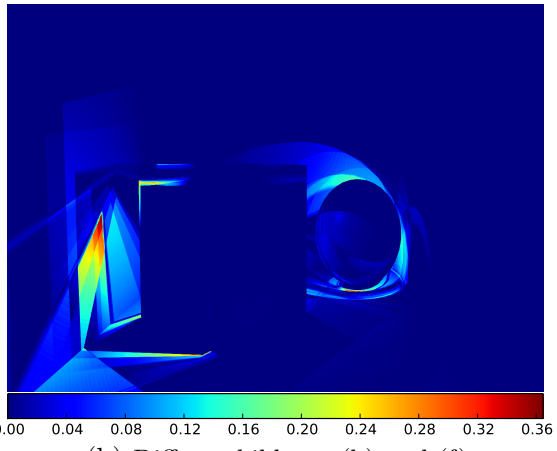
(e) Schatten für 5 Cluster bei 12.1 ms.



(f) Schatten für 4 Cluster bei 11.1 ms.



(g) Differenzbild von (b) und (e).



(h) Differenzbild von (b) und (f).

Abbildung 5.6.: Vergleich bei der Cornell-Box Szene.

(a) Verteilung der Lichter.

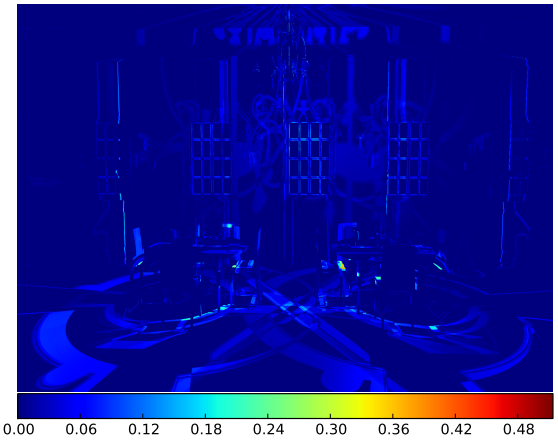
(b) Referenzlösung mit 39.3 ms.

(c) Clustering für $d_{init} = 0.3$ mit 9 Clustern.

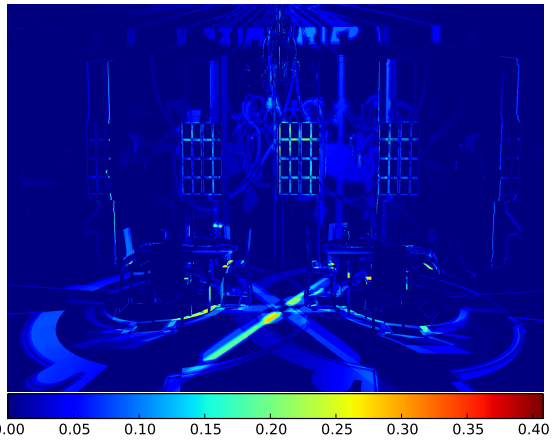
(d) Clustering für $d_{init} = 0.4$ mit 8 Clustern.

(e) Schatten für 9 Cluster mit 32.9 ms.

(f) Schatten für 8 Cluster mit 30.5 ms.



(g) Differenzbild zwischen (b) und (e).



(h) Differenzbild zwischen (b) und (f).

Abbildung 5.7.: Vergleich bei der Restaurant Szene.

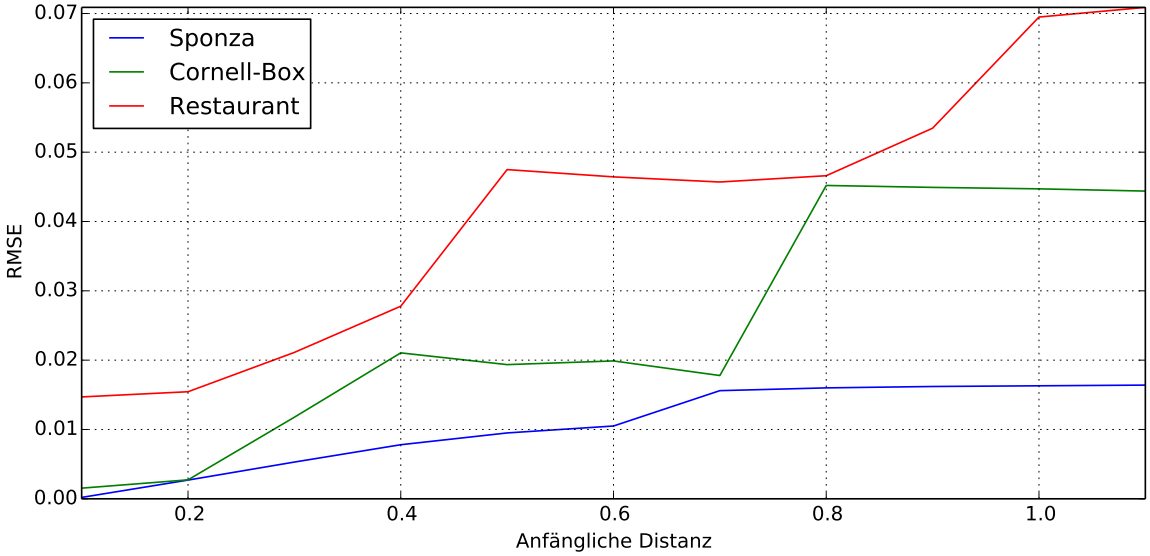


Abbildung 5.8.: Wurzel des mittleren quadratischen Fehlers unseres Algorithmus im Vergleich zur Referenzlösung.

Problem zu lösen, müsste der Repräsentant auf eine Verdeckung getestet werden. Beim Minimum-Distanz Clustering muss dies nicht durchgeführt werden.

Unser Ansatz kann jedoch einige Artefakte erzeugen. Wenn die Kamera bewegt oder Lichtquellen animiert werden, kann es zu temporären Aliasing kommen. Diese Artefakte können reduziert werden, indem die Skalierung der Cluster-Distanz basierend auf dem Abstand zur Kamera deaktiviert wird. Für animierte Lichtquellen ist es zusätzlich sinnvoll, diese als Repräsentanten zu wählen.

Die Ergebnisse haben gezeigt, dass unser Algorithmus dazu benutzt werden kann, zwischen der Qualität und der Geschwindigkeit bei der Schattenberechnung für viele Lichtquellen abzuwägen. Durch die Interpretation eines Clusters als scheibenförmige Flächenlichtquelle können die Fehler im Schatten reduziert werden.

5.1.7. Zusammenfassung und weitere Arbeiten

Wir haben einen Ansatz vorgestellt, der Repräsentanten aus einer Menge an Lichtquellen auswählt und diese für die Berechnung von Schatten verwendet. Durch den Fokus auf hochfrequente Schatten einer direkten Beleuchtung, haben wir einen modifizierten Minimum-Distanz Clustering Algorithmus entwickelt, der verschiedene Metriken kombiniert. Dieser wird benutzt um eine variable Anzahl an Repräsentanten zu bestimmen und die restlichen Lichtquellen diesen Repräsentanten zuzuordnen. Die Cluster werden anschließend als scheibenförmige Flächenlichtquellen interpretiert und die Lichtgröße anhand der Clustergröße skaliert. Die Schatten für jeden Cluster werden mit einem Algorithmus für weiche Schatten gerendert. Durch die anfängliche Distanz beim Clustern kann die Geschwindigkeit gegenüber der Qualität abgewogen werden. Die Ergebnisse

haben gezeigt, dass unser Algorithmus zur Beschleunigung der Schattenberechnung benutzt werden kann, während eine akzeptable Qualität der Schatten gewahrt bleibt.

In weiteren Arbeiten wollen wir die Heuristik zur Auswahl der Repräsentanten erweitern, indem der Einfluss auf das View-Frustum genauer berücksichtigt wird. Dazu kann die Fläche des Hüllkörpers als Sortierkriterium herangezogen werden. Des Weiteren kann die Abschätzung der beleuchteten Bereiche durch ein Occlusion-Culling mit Hilfe eines hierarchischen Z-Buffers [33] weiter verbessert werden. Des Weiteren möchten wir andere Methoden für die Schattenberechnung von lokalen Punktlichtquellen implementieren, wie z.B. Paraboloid Shadow Mapping [14] und Tetrahedron Mapping [42].

5.2. Weiche Schatten mit Hilfe eines Erosionsoperators

Bei aktuellen Verfahren für Schatten mit variabler Halbschattenbreite gibt es einige Einschränkungen, die den praktischen Einsatz erschweren. Summed-Area Variance Shadow Maps von Lauritzen [52] ist nur für kleine Shadow Map Auflösungen geeignet, da sonst ein Float-Überlauf entsteht. Percentage Closer Soft Shadows [30] benötigt für jeden sichtbaren Pixel eine Blockersuche, die viele Texturzugriffe erfordert. Screen Space Soft Shadows [35] arbeitet direkt auf den Shadow Maps wodurch die Geschwindigkeit mit der Anzahl der Shadow Maps sinkt.

In diesem Abschnitt stellen wir einen Algorithmus vor, der Schatten mit variabler Halbschattenbreite mit Hilfe eines Erosionsoperators erzeugt. Unser Algorithmus ist eine Erweiterung von Shadow Mapping und arbeitet direkt im Screen Space. Des Weiteren ist er geeignet für hohe Shadow Map Auflösungen und mehrere Shadow Maps.

Unsere Beiträge sind:

- Ein Algorithmus der Schatten mit variabler Halbschattenbreite über einen Erosionsoperator erzeugt.
- Eine Technik zur Reduzierung von Diskretisierungsfehlern bei weichen Schatten über eine Geradenanpassung mit Hilfe der Methode der kleinsten Quadrate.

5.2.1. Stand der Technik

Die Arbeit von Arvo et al. [9] ähnelt unserer Arbeit. Sie bestimmen die Halbschattenbereiche mit Hilfe einer Kantenerkennung bei harten Schatten und propagieren einen Sichtbarkeitsfaktor mittels einem Floodfill-Algorithmus. Rong und Tan [85] beschleunigen diese Methode mit einem Jump Floodfill-Algorithmus. Gumbau et al. [35] ersetzen die Blockersuche bei PCSS durch eine Dilatation einer Shadow Map und bestimmen den Schattenfaktor durch einen separierbaren Gauß-Filter.

Unsere Methode zählt zu den Screen-Space Verfahren für weiche Schatten. Robison und Shirley [84] benutzen eine Screen-Space Distance Map, um die Halbschattenbereiche zu bestimmen. Diese wird dazu genutzt um harte Schatten weichzuzeichnen. Die Arbeit von Hanjun und Huali [36] ist sehr ähnlich zu unserer Arbeit. Sie propagieren einen Schattenfaktor mit Hilfe eines Erosions- und Dilatationsfilters. Allerdings bestimmen wir Schatten mit einer variablen Halbschattenbreite und präsentieren eine Methode um das Aliasing zu reduzieren, welches durch eine Dilatation des Halbschattens entsteht. Aguado und Montiel [3] präsentieren einen Ansatz bei dem ein Halbschatten über ein MipMap Floodfill propagiert und der Schattenfaktor mit einem Gauß-Filter im Screen-Space bestimmt wird. Allerdings leidet dieser Ansatz unter Light-Leaks, die aber durch mehrere Shadow Maps reduziert werden können. MohammandBagher et al. [61] projizieren eine Shadow Map in den Screen-Space um die Halbschattenbereiche abzuschätzen und erzeugen einen Schattenfaktor durch eine Tiefpassfilterung von harten Schatten.

5.2.2. Algorithmus

Die Vorgehensweise unseres Algorithmus besteht aus vier Schritten (Abbildung 5.9). Zuerst werden harte Schatten gerendert. Anschließend werden Kanten in den harten Schatten gesucht und die Kanten-Pixel, zusammen mit der Distanz zwischen Schattenspender und Schattenempfänger, sowie der Distanz zur Kamera, in eine Textur gespeichert. Mit Hilfe dieser Werte kann eine Halbschattenbreite abgeschätzt werden. Im nächsten Schritt werden die Kanten-Pixel mit einen Filterkern erodiert, der mit der Halbschattenbreite skaliert wird. Damit können die Bereiche des Halbschattens abgeschätzt werden. Im letzten Schritt wird dann ein Halbschatten erzeugt. Dazu werden zwei Lösungen vorgestellt. Die erste Lösung benutzt einen Percentage Closer Filter [79] um den Halbschatten zu erzeugen und die zweite Lösung berechnet den Halbschatten direkt bei der Erosion.

5.2.2.1. Rendern von harten Schatten

Im ersten Schritt des Algorithmus werden harte Schatten gerendert und Zusatzinformationen, zur Bestimmung der Halbschattenbreite, ermittelt. Wir nehmen an, dass die Shadow Map bereits gerendert wurde. Die Szene wird von der Betrachterposition erneut gezeichnet und ein Vergleich der Tiefenwerte durchgeführt. Das binäre Ergebnis wird, zusammen mit der Distanz zwischen Schattenspender und Schattenempfänger sowie der Distanz zur Kamera, in eine Textur gespeichert. Des Weiteren werden in einem Geometry-Buffer (G-Buffer) die notwendigen Daten für die Beleuchtungsrechnung gespeichert.

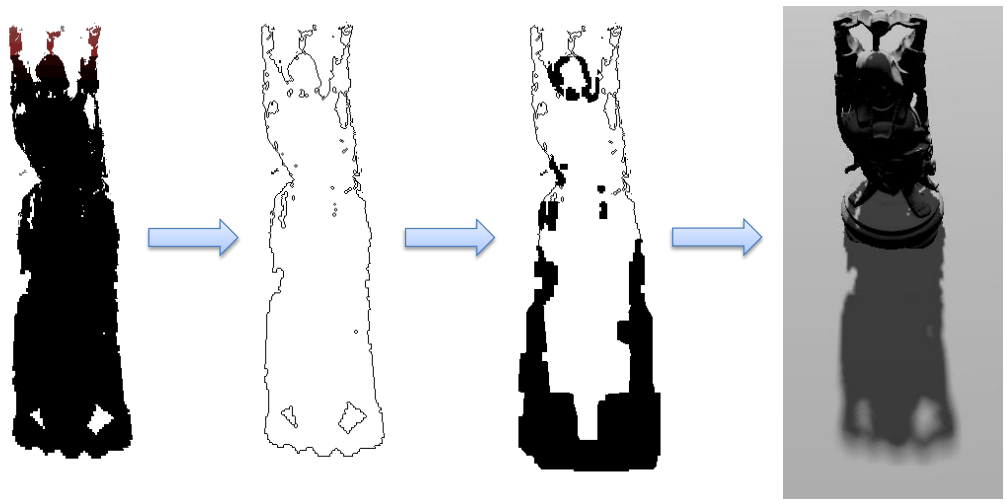


Abbildung 5.9.: Ablauf des Algorithmus. Zuerst werden harte Schatten gerendert. In den harten Schatten wird anschließend eine Kantenerkennung durchgeführt und die Kanten mit einem Erosionsfilter mit variabler Filterkerngröße erweitert. Im letzten Schritt wird der Schattenfaktor erzeugt.

5.2.2.2. Kantenerkennung

Im zweiten Schritt werden die Bereiche des Halbschattens abgeschätzt, indem eine Kantenerkennung auf den harten Schatten durchgeführt wird. Da die harten Schatten nur Binärinformationen beinhalten, können die Ecken über einen 3 x 3 Laplace Filter bestimmt werden. Der Filterkern besitzt folgende Form:

$$\begin{pmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{pmatrix}$$

Für jeden Kantenpixel wird anschließend die Halbschattenbreite nach Fernando [30] mit Hilfe des Strahlensatzes ermittelt. Zusätzlich wird die Halbschattenbreite mit der Distanz zur Kamera skaliert (vgl. [35]):

$$\omega_{penumbra} = \frac{(d_{receiver} - d_{blocker})\omega_{light}}{d_{blocker} \cdot d_{observer}}$$

Mit der Lichtgröße ω_{light}. Die Halbschattenbreite wird in dem zweiten Texturkanal der Kantentextur gespeichert.

5.2.2.3. Erosion

Nach der Kantenerkennung wird die berechnete Halbschattenbreite an den Kantenpixel mit Hilfe eines Erosionsfilters, mit variabler Filterkerngröße, erodiert. Die Erosion

wird dabei mit Hilfe einer Min-Max MipMap [24, 44] durchgeführt. Die Kantentextur speichert eine Null für jede Kante im ersten Texturkanal und die Halbschattenbreite im zweiten Texturkanal. Die MipMap Hierarchie wird erzeugt, indem eine Minimum Operation auf den ersten Texturkanal und eine Maximum-Operation auf den zweiten Texturkanal in einer 3 x 3 Umgebung durchgeführt wird.

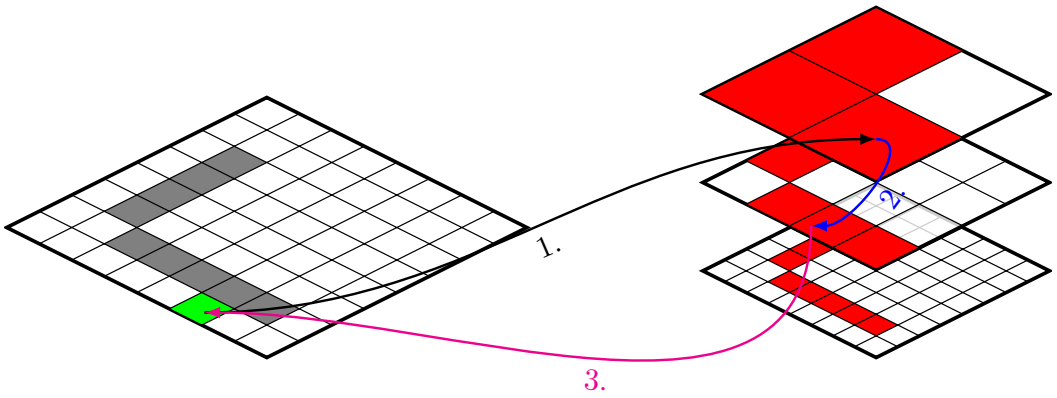


Abbildung 5.10.: Erosion mit einer Min-Max MipMap Hierarchie. Als erstens wird eine MipMap Stufe mit Hilfe eines maximalen Suchradius bestimmt und die Halbschattenbreite aus der MipMap Hierarchie gelesen. Anhand dieser Halbschattengröße wird erneut eine MipMap Stufe bestimmt und die Textur auf einen Kantenpixel getestet. Wenn ein Kantenpixel vorhanden ist, wird die Halbschattenbreite im aktuellen Pixel gespeichert.

Der Ablauf der Erosion wird in Abbildung 5.10 dargestellt. Für die Erosion wird zuerst ein maximaler Suchradius, für die gegebene Lichtgröße und die Distanz zur Kamera, bestimmt. Der Suchradius wird in einen MipMap-Level umgewandelt, indem ein Logarithmus mit der Basis 2 auf den Suchradius angewandt wird. Auf die MipMap-Hierarchie wird anschließend in dieser Stufe zugegriffen und überprüft, ob ein Kantenpixel vorhanden ist. Falls kein Kantenpixel vorhanden ist, kann die Erosion beendet werden. Andernfalls wird die Halbschattenbreite aus dem zweiten Texturkanal ausgelesen und wiederholt eine zugehörige MipMap Stufe bestimmt. Auf die MipMap Hierarchie wird nun erneut zugegriffen und auf einen Kantenpixel getestet. Wenn dieser Test positiv ausfällt, wird die Halbschattenbreite in die Ergebnistextur gespeichert. Andernfalls wird der Pixel verworfen.

5.2.2.4. Schattenfaktor mit PCF bestimmen

Im letzten Schritt wird ein Schattenfaktor mit Hilfe eines Percentage Closer Filters [79] bestimmt. Dazu wird die Halbschattenbreite für jeden Pixel aus der erodierten Kantentextur ausgelesen. Der PCF-Filterkern wird anhand der Halbschattenbreite skaliert und das Ergebnis mit den harten Schatten kombiniert.

5.2.2.5. Schattenfaktor während der Erosion bestimmen

Eine weitere Möglichkeit um einen Schattenfaktor zu bestimmen besteht darin, diesen direkt während der Erosion zu ermitteln. Um diese Idee umzusetzen, sind einige Anpassungen am Algorithmus notwendig.

Da die harten Schatten aus der Sicht des Betrachters erzeugt werden, kann es vorkommen, dass Objekte den Schatten verdecken. Das kann dazu führen, dass Kanten erkannt werden, die allerdings zu keinem Halbschatten gehören. Nach Arvo et al. [9] kann dieses Problem gelöst werden, indem die Shadow Map auf eine Silhouette für jeden Kantenpixel getestet wird. Dazu benutzen wir den gleichen 3 x 3 Laplace Filter wie bei der Kantenerkennung und vergleichen das Ergebnis mit einem Schwellenwert. Wenn das Ergebnis kleiner als der Schwellenwert ist, ist der Kantenpixel gültig und wird für die folgenden Schritte benutzt. Andernfalls wird der Kantenpixel verworfen.

Um einen Schattenfaktor zu bestimmen, wird die Erosion nach einem Gathering Schema implementiert. Zuerst wird eine maximale Filterkerngröße bestimmt und nach Kantenpixeln in diesem Bereich gesucht. Ist ein Kantenpixel vorhanden, wird dessen Distanz mit der gespeicherten Halbschattenbreite vergleichen. Dieser Vorgang wird solange wiederholt, bis der Kantenpixel mit der kleinsten Distanz zum aktuellen Pixel gefunden wird. Der Schattenfaktor des äußeren Halbschattens kann dann wie folgt bestimmt werden:

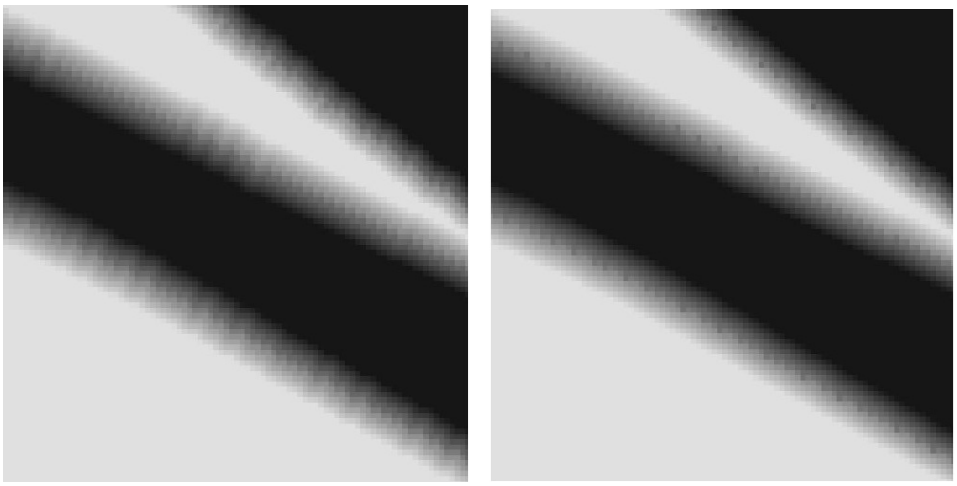
$$s_{outer} = \frac{\omega_{penumbra} - d_{min}}{2 \cdot \omega_{penumbra}}$$

Wobei d_{min} die kleinste Distanz zur Kantenpixel und $\omega_{penumbra}$ die Halbschattenbreite ist. Der innere Halbschatten kann mit $s_{inner} = 1 - s_{outer}$ bestimmt werden.

Ein Problem bei dieser Vorgehensweise ist allerdings, dass mögliche Aliasing Artefakte im harten Schatten auf den weichen Schatten übertragen werden. Abbildung 5.11a zeigt ein Beispiel solcher Aliasing Artefakte.

Um das visuelle Ergebnis zu verbessern, werden die Kantenpixel durch eine Gerade ersetzt, die mit Hilfe einer Geradenanpassung nach der Methode der kleinsten Quadrate bestimmt wird. Diese Geraden werden in einer diskreten Umgebung um jeden Kantenpixel ermittelt und in eine zusätzliche Textur gespeichert. Während der Erosion werden die Parameter der Geraden aus der Textur geladen und die Distanz zum Kantenpixel durch die Distanz zur Geraden ersetzt (Abbildung 5.12).

Abbildung 5.11b zeigt das Ergebnis des Schattenfaktors mit Hilfe der Gerade. Allerdings behebt diese Vorgehensweise nicht alle Artefakte. Im Übergangsbereich vom inneren zum äußeren Halbschatten werden noch einige Pixel falsch dargestellt. Das liegt daran, dass der Abstand d_{min} an diesen Pixel geringfügig kleiner oder größer als bei Pixel in der Umgebung ist.



(a) Artefakte die durch Aliasing im harten Schatten entstehen

(b) Ergebnis nach Korrektur der minimalen Distanz über eine angepasste Gerade.

Abbildung 5.11.: Artefakte bei der direkten Berechnung eines Schattenfaktors bei der Erosion.

5.2.3. Ergebnisse

Wir vergleichen unseren Algorithmus gegen eine Referenzlösung und eine PCSS Implementierung. Die Referenzlösung nähert eine Flächenlichtquelle durch 512 Punktlichtquellen an, rendert für jede Punktlichtquelle einen harten Schatten und mittelt das Ergebnis über alle Lichtquellen. Die PCSS Implementierung benutzt eine Poisson-Disk für die Blockersuche und das PCF. Die Anzahl der Abtastpunkte für die Blockersuche wird dabei variiert und die Geschwindigkeit mit unseren Algorithmus verglichen.

Sowohl die PCSS Implementierung als auch unser Algorithmus benutzen 32 Abtastpunkte für die Erzeugung eines Schattenfaktors mit PCSS. Die Bildschirmauflösung

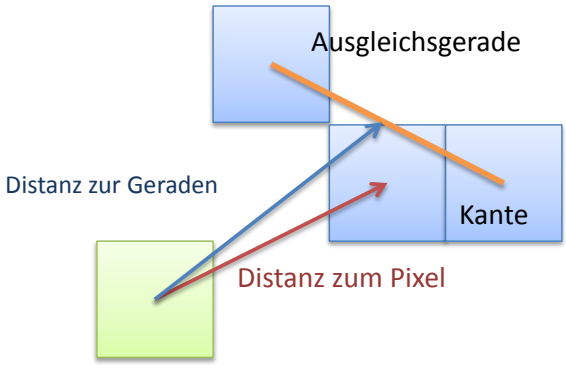


Abbildung 5.12.: Reduzierung von Artefakten bei der Bestimmung des Schattenfaktors. Die Distanz zum Pixel wird durch eine minimale Distanz zur Geraden ersetzt.

beträgt 1920 x 1080 und die Shadow Map Größe 2048 x 2048.

Abbildung 5.13 und 5.14 zeigen die visuellen Ergebnisse. Bei 8 Abtastpunkten für die Blockersuche bei PCSS sind im Bild deutliche Artefakte zu erkennen. Das liegt daran, dass Schattenspender durch die wenigen Abtastpunkte nicht gefunden werden und somit der Algorithmus vorzeitig terminiert. Unser Algorithmus hingegen ist von diesem Problem nicht betroffen. Allerdings erzeugt unser Algorithmus nicht den gleichen Schatten wie PCSS. Das liegt daran, dass die Halbschattenbreite nur für Kantenpixel des harten Schattens berechnet wird. Durch die Erosion mit Hilfe der MipMap-Hierarchie wird anschließend das Maximum dieser Halbschattenbreite in die Nachbarschaft propagiert. Nichtsdestotrotz ist der optische Eindruck des Schattens vergleichbar mit PCSS und der Referenzlösung.

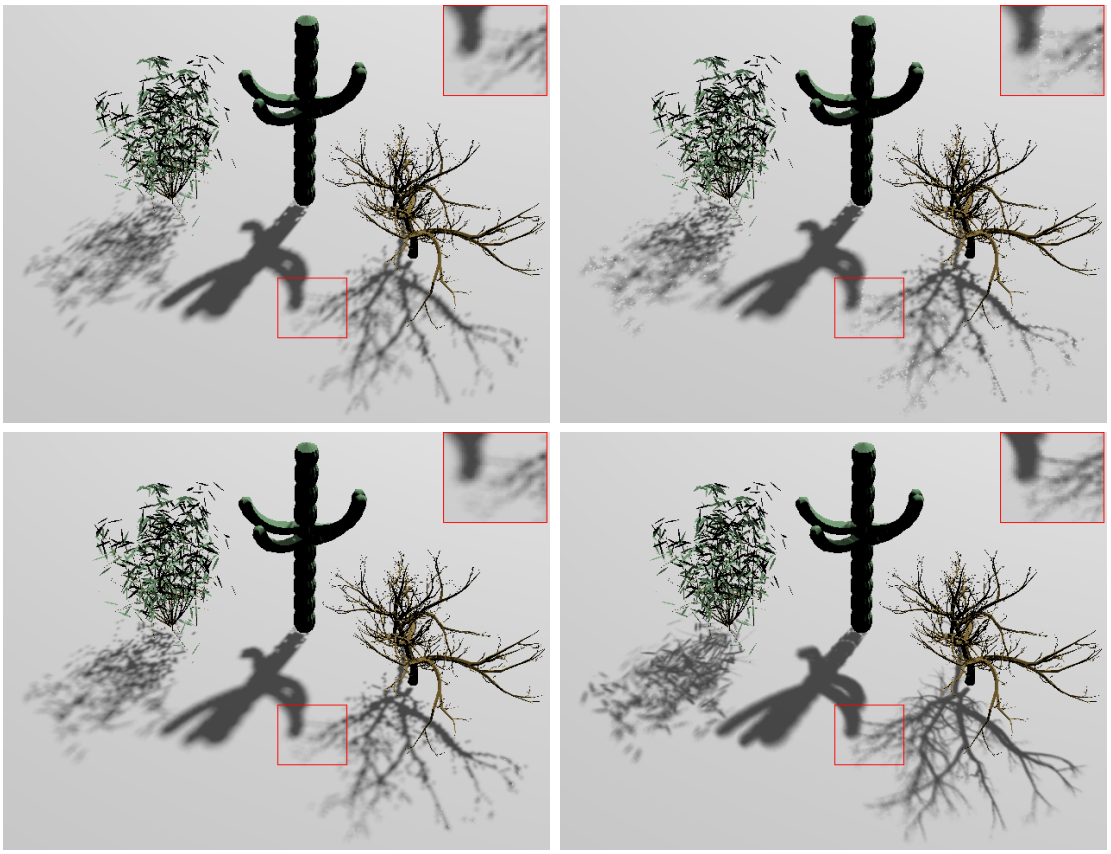


Figure 5.13.: Visuelle Ergebnisse des Schattens in der Kaktus Szene. Oben links: Unser Algorithmus. Oben rechts: PCSS mit 8 Abtastpunkten bei der Blockersuche. Aufgrund der wenigen Abtastpunkte entstehen Artefakte im Schatten weil Schattenspender nicht gefunden werden. Unten links: PCSS mit 64 Abtastpunkten bei der Blockersuche. Unten rechts: Referenzlösung

Tabelle 5.3 stellt die Geschwindigkeit im Vergleich zu PCSS dar. Die Geschwindigkeit wurde auf einen Intel Xeon E5620 Prozessor mit 2,4 GHz, 8 GB RAM und einer NVIDIA GeForce GTX 680 mit 2048 MB Speicher ermittelt. Tabelle 5.4 schlüsselt die Dauer der einzelnen Schritte im Algorithmus auf.

	Kaktus (188K Dreiecke)	**Hairball** (2.88M Dreiecke)	**Buddha** (1.08M Dreiecke)
Erosion	1.62	12.56	5.77
PCSS 8	1.58	11.51	4.68
PCSS 32	1.60	11.89	5.11
PCSS 64	1.66	12.44	5.69

Tabelle 5.3.: Geschwindigkeitsergebnisse im Vergleich zu PCSS mit 8, 32 und 64 Abtastpunkten bei der Blockersuche. Angaben in Millisekunden.

Schritt	**Zeit**
Shadow Map	2.2 ms
Harte Schatten	2.5 ms
Kantenerkennung	0.3 ms
Erosion	0.3 ms
Shading	0.4 ms

Tabelle 5.4.: Dauer der einzelnen Schritte bei der Buddha-Szene.

5.2.4. Diskussion

Der Flaschenhals bei PCSS ist die Blockersuche, die für jeden sichtbaren Pixel durchgeführt werden muss. Unser Ziel bestand darin, die Blockersuche durch einen Algorithmus zu ersetzen, der nur auf den Silhouetten des harten Schattens arbeitet. Allerdings ist unser Algorithmus komplexer und ein Geschwindigkeitsvorteil, aufgrund der gestiegenen Anzahl an Textureinheiten und des größeren Texturcaches einer NVIDIA 680 GTX Grafikkarte, nicht ausmachbar. Des Weiteren sinkt die Geschwindigkeit durch die Erosion wenn der Halbschatten große Bereiche des Bildes ausmacht. Im Gegensatz zu PCSS verursacht unsere Methode bei fein strukturierter Geometrie keine Artefakte im Bild wenn eine die Blockersuche nur mit wenigen Abtastpunkten ausgeführt wird.

Eine möglicher Einschränkung bei der Methode von Gumbau et al. [35] besteht darin, dass der Algorithmus direkt auf Shadow Maps arbeitet. Im Gegensatz dazu arbeitet unser Algorithmus auf den harten Schatten, was ihn attraktiv für Anwendungen mit mehr als einer Shadow Map macht.

Im Vergleich zu Hanjun und Huali [36] ermöglicht unser Algorithmus die Darstellung von Schatten mit variabler Halbschattenbreite. Des Weiteren haben wir eine mögliche Lösung vorgestellt, wie die Artefakte bei der Berechnung eines Schattenfaktors während der Erosion reduziert werden können. Durch den Einsatz von PCF stellen wir eine zweite Möglichkeit vor einen Schattenfaktor zu erzeugen, der in einen vereinfachten Algorithmus und verbesserter Schattenqualität resultiert.

Nichtsdestotrotz hat diese Technik einige Einschränkungen. Da unsere Methode auf PCSS basiert, besitzt sie dieselben Einschränkungen, wie z.B. eine Überschätzung der Halbschattenbreite. Des Weiteren ist die Größe des Erosionsfilters beschränkt,

wo durch relevante Schattenspender ausgelassen werden können. Durch die MipMap Erosion und die Skalierung der Halbschattenbreite, basierend auf der Distanz zur Kamera, verursacht unsere Methode Aliasing wenn die Kamera bewegt wird. Eine weitere Einschränkung besteht darin, dass die Qualität des Halbschattens sehr stark von der Qualität der harten Schatten abhängt. Daher sollten Verfahren zur Reduzierung von Aliasing eingesetzt werden, wie z.B. Cascaded Shadow Maps [29] oder Light Space Perspective Shadow Maps [101].

5.2.5. Fazit und Ausblick

Wir haben einen Algorithmus vorgestellt, der Schatten mit variabler Halbschattenbreite im Screen Space berechnet. Wie alle Screen Space Verfahren, ist diese Technik am besten für kleine Halbschattenbreiten geeignet und kann dazu genutzt werden, Anwendungen mit Shadow Mapping zu erweitern. Des Weiteren haben wir zwei Lösungen vorgestellt, wie ein Schattenfaktor generiert werden kann. Während die Erosion der Schattenkanten und die Erzeugung eines Schattenfaktors mit PCF vergleichbare Ergebnisse wie PCSS liefert, verursacht die Generierung des Schattenfaktors während der Erosion noch einige Artefakte.

In zukünftigen Arbeiten wollen wir die Geradenberechnung mit der Methode der kleinsten Quadrate durch einen Tiefpassfilter ersetzen. Des Weiteren wollen wir die verbleibenden Artefakte bei der Schattenfaktorberechnung während der Erosion reduzieren.

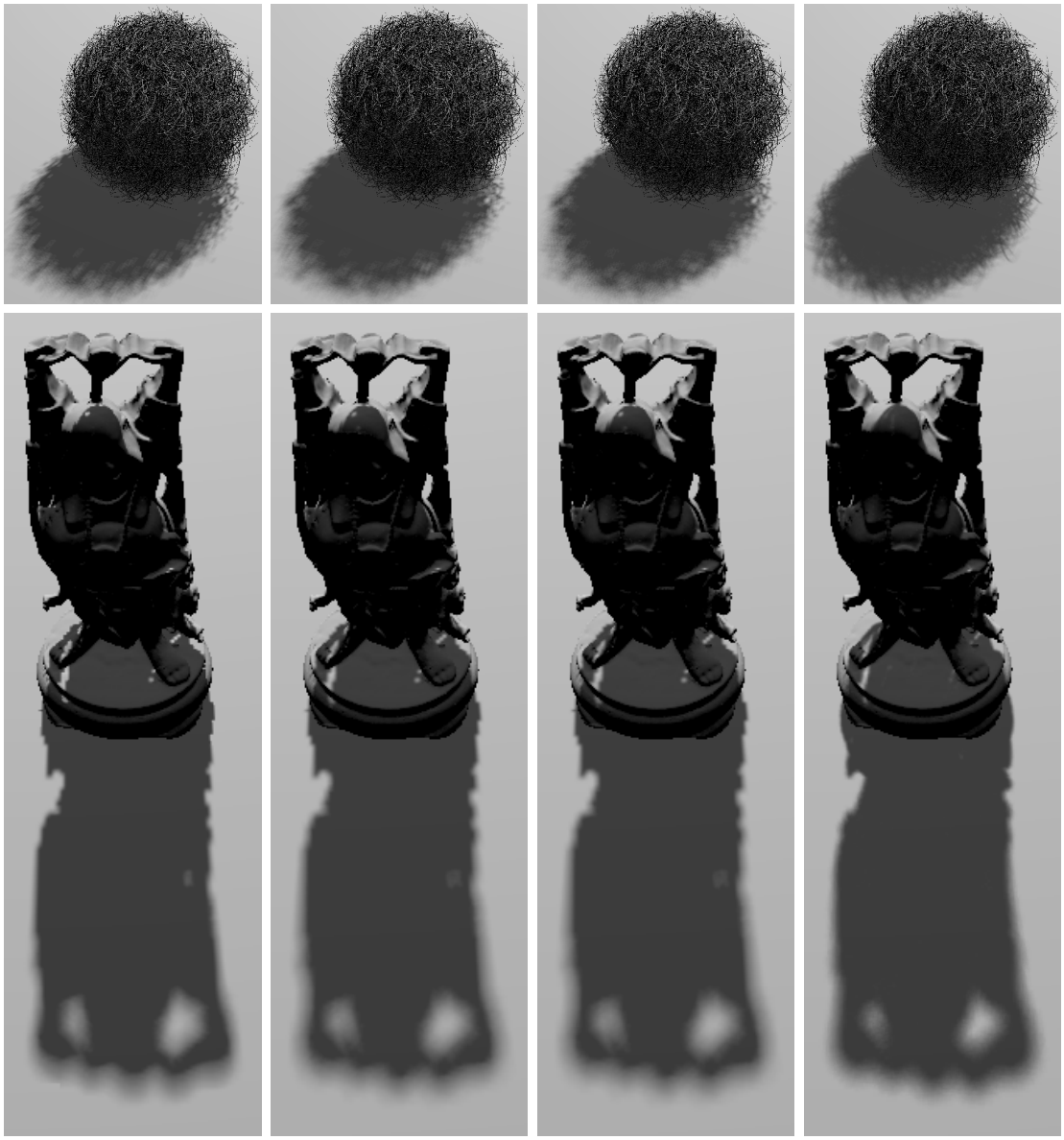


Abbildung 5.14.: Visuelle Ergebnisse der Schatten in der Hairball und Buddha Szene. Von links nach rechts: Unser Algorithmus, PCSS mit 32 Abtastpunkten bei der Blockersuche, PCSS mit 64 Abtastpunkten bei der Blockersuche und die Referenzlösung.

6. Thermische Schatten mit atmosphärischer Verdeckung

In diesem Kapitel wird die Darstellung von thermischen Schatten um einen dynamischen indirekten Wärmebeitrag erweitert. Dazu wird zuerst die Idee und der Algorithmus vorgestellt. Anschließend wird genauer auf die Implementierung eingegangen. Zum Abschluss wird eine Fehler- und Geschwindigkeitsanalyse durchgeführt.

6.1. Idee

Neben der direkten Wärmeeinstrahlung, liefert die indirekte Wärmeeinstrahlung aus der Umgebung einen entscheidenden Beitrag zur Wärmebilanz. Bei dem im Kapitel 4 vorgestellten Algorithmus für thermische Schatten wird die gesamte Umgebungsstrahlung als eine Konstante angesehen, die in der Strahlungs- bzw. Schattentemperatur enthalten ist. Das Ziel dieses Ansatzes besteht darin die indirekte Wärmeeinstrahlung aus der Umgebung dynamisch zu berücksichtigen (Abbildung 6.1).

Allerdings erfordert dies einen neuen Algorithmus zur Berechnung der Ausgleichstemperatur. Unsere Idee besteht darin, aus den Strahlungsbeiträgen eine Umgebungstemperatur abzuleiten und von dieser, über ein Strahlungsgleichgewicht, eine Ausgleichstemperatur zu ermitteln. Im Gegensatz zu vorberechneten Temperaturen, besitzt diese Vorgehensweise den Vorteil, dass dadurch sowohl die Orientierung der Oberfläche als auch mehrere Strahlungsquellen berücksichtigt werden können.

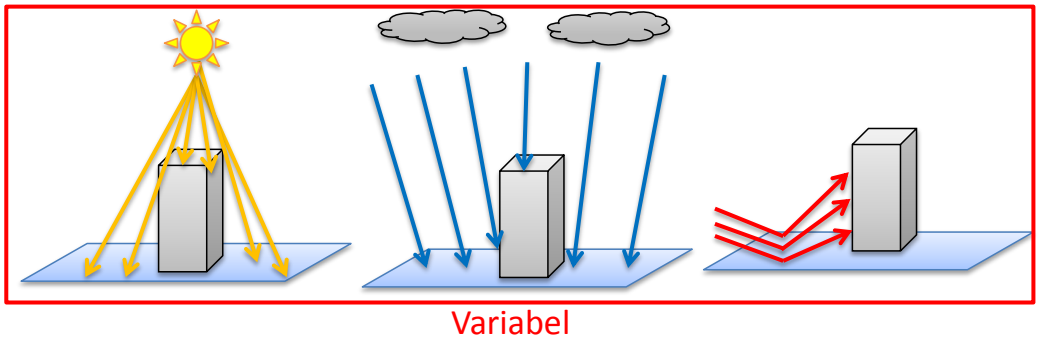


Abbildung 6.1.: Neben dem direkten Wärmestrom wird der indirekte Wärmestrom der Atmosphäre und der Umgebung berücksichtigt.

6.2. Algorithmus

Nach Schätz [73] kann eine Umgebungstemperatur definiert werden, indem die Energieeinträge auf der Oberfläche eines schwarzen Körpers betrachtet werden. Die Strahlungsbeiträge sind im Einzelnen:

- Der direkte Wärmestrom S_d einer Wärmequelle $\left[\frac{W}{m^2}\right]$.
- Der indirekte Wärmestrom S_i einer Wärmequelle $\left[\frac{W}{m^2}\right]$.
- Der indirekte Wärmestrom R_e der Umgebung $\left[\frac{W}{m^2}\right]$.
- Der indirekte Wärmestrom R_{LW} der Atmosphäre $\left[\frac{W}{m^2}\right]$.

Weitere Wärmeströme, wie z.B. durch Konvektion oder Strahlung, werden in diesem Algorithmus vernachlässigt. Für die Bestimmung der Umgebungstemperatur T_a wird ein Strahlungsgleichgewicht gebildet:

$$\varepsilon\sigma T_a^4 = S_d f_d\,(1-\alpha)\,(1-\beta) + S_i f_i\,(1-\alpha) + (1-f_i)\,R_e + \varepsilon f_i R_{LW} \tag{6.1}$$

Die restlichen Symbole sind der Tabelle 6.1 zu entnehmen.

Symbol	**Bezeichnung**
α	Albedo der Oberfläche
β	Bewölkungsgrad
ε	Emissionsgrad der Oberfläche
σ	Stefan-Bolzmann Konstante $\left[\frac{W}{m^2K^4}\right]$
f_d	Schattenfaktor der direkten Strahlung
f_i	Schattenfaktor der indirekten Strahlung

Tabelle 6.1.: Bedeutung der Symbole für die Energieeinträge eines schwarzen Körpers.

Zur Berechnung der Umgebungstemperatur wird die Gleichung 6.1 nach T_a umgestellt:

$$T_a = \sqrt[4]{\frac{1}{\varepsilon\sigma}\left(S_d f_d\,(1-\alpha)\,(1-\beta) + S_i f_i\,(1-\alpha) + (1-f_i)\,R_e + \varepsilon f_i R_{LW}\right)} \tag{6.2}$$

Im Folgenden wird genauer auf die einzelnen Wärmeströme eingegangen:

6.2.1. Direkter Wärmestrom

Der direkte Wärmestorm ist abhängig von der Ausrichtung der Oberfläche [73]:

$$S_d = S_o cos(N \cdot L) \tag{6.3}$$

Wobei S_o die direkte Einstrahlung $\left[\frac{W}{m^2}\right]$, N der Normalenvektor der Oberfläche sowie L der Richtungsvektor von der Oberfläche zum Strahler ist.

6.2.2. Indirekter Wärmestrom

Der indirekte Wärmestrom fasst den reflektieren Wärmestrom des Strahlers aus der Umgebung zusammen. Es ist wie folgt definiert [73]:

$$S_i = \int_{\Omega} S_{i0} cos(\gamma) d\Omega \tag{6.4}$$

Wobei S_{io} die indirekten Einstrahlung $\left[\frac{W}{m^2 sr}\right]$, Ω die Kugel um den Oberflächenpunkt und γ der Einfallswinkel der Strahlung ist. Bei einer homogenen Strahlungsdichte wird das Integral über die Kugel durch $S_i = S_{io} 4\pi$ ersetzt [73].

Neben den indirekten Wärmestrom eines Strahlers werden weitere reflektierte Wärmeströme aus der Umgebung R_e, sowie der langwellige Wärmestrom der Atmosphäre R_{LW} berücksichtigt. Die reflektierten Wärmeströme können durch eine numerische Simulation, wie z.B. mit RadTherm, ermittelt werden. Letzterer Anteil wird über ein Atmosphärenmodell, wie z.B. MODTRAN [6], bestimmt.

6.2.3. Ausgleichstemperatur

Für die Bestimmung der Ausgleichstemperatur wird angenommen, dass eine Oberfläche nur aus einer unendlich ausgedehnten Schicht besteht. Die Ausgleichstemperatur T_s ist nach Schätz [73] durch die Gleichheit von Strahlung und Bodenwärmestrom definiert:

$$\varepsilon\sigma(T_a^4 - T_s^4) = \lambda \frac{T_s - T_c}{d} \tag{6.5}$$

Wobei ε der Emissionsgrad der Oberfläche, σ die Stefan-Bolzmann Konstante, T_a die Umgebungstemperatur, λ die Wärmeleitfähigkeit, d die Schichtdicke und T_c die Kerntemperatur ist. Die Kerntemperatur ist die Temperatur einer tiefliegenden Schicht, die im Simulationszeitraum als Konstante angesehen werden kann [73].

Eine analytische Lösung der Ausgleichstemperatur ist möglich, indem Gleichung 6.5 nach T_s aufgelöst wird [73]:

$$T_s = \frac{T_b}{2}\left(\sqrt{\frac{2\sqrt{3H}}{\sqrt{3H^2 - 4\theta_c - 4\theta_a^4}} - H + \frac{4\theta_c + 4\theta_a^4}{3H}} - \frac{\sqrt{3H^2 - 4\theta_c - 4\theta_a^4}}{\sqrt{3H}}\right) \tag{6.6}$$

Mit den Termen:

$$b = \frac{d\sigma\varepsilon}{\lambda} \tag{6.7}$$

$$T_b = \frac{1}{\sqrt[3]{b}} \tag{6.8}$$

$$N = \sqrt{256\frac{T_c^3}{T_b^3} + 768\frac{T_a^4 T_c^2}{T_b^6} + 768\frac{T_a^8 T_c}{T_b^9} + 256\frac{T_a^{12}}{T_b^{12}} + 27} \tag{6.9}$$

$$H = \left(\frac{N}{2 \cdot 3^{\frac{3}{2}}} + \frac{1}{2}\right)^{\frac{1}{3}} \tag{6.10}$$

$$\theta_c = \frac{T_c}{T_b} \tag{6.11}$$

$$\theta_a = \frac{T_a}{T_b} \tag{6.12}$$

6.2.4. Vorgehensweise

Die Vorgehensweise des Algorithmus lautet wie folgt [73]:

1. Schattenfaktor der direkten f_d und indirekten f_i bestimmen.
2. Albedo α und Emissivität ε der Oberfläche auslesen.
3. Umgebungstemperatur T_a nach Gleichung 6.2 ermitteln.
4. Ausgleichstemperatur T_s über Formel 6.6 berechnen.
5. Oberflächentemperatur mit Hilfe des thermischen Modells (Kapitel 3.4) anpassen.

6.3. Implementierung

Im folgenden Abschnitt wird genauer auf die Implementierung eingegangen. Diese Implementierung baut auf der im Kapitel 4 vorgestellten Implementierung auf.

6.3.1. Parameterbestimmung

Zur Bestimmung der materialabhängigen Zeitkonstanten und Gewichtungsfaktoren des thermischen Modells wird analog wie im Kapitel 4 vorgegangen. Für die gegebenen Strahlungsdaten wird die Ausgleichstemperatur nach Formel 6.6 bestimmt. Aus dieser kann über das Strahlungsgleichgewicht (Formel 4.5) der Wärmeübergangskoeffizient und die Biot-Zahl hergeleitet werden. Anschließend wird eine Fourier-Reihe numerisch gelöst und eine Kurvenanpassung mit dem thermischen Modell vorgenommen. Alternativ kann eine Kurvenanpassung gegen eine numerische Lösung von RadTherm erfolgen.

6.3.2. Bestimmung des Sonnenwärmestroms

Der Wärmestrom der Sonne wird mit einem Atmosphärenmodell, wie z.B. MODTRAN [6], berechnet. Allerdings liefert MODTRAN nur die stündliche, totale Sonneneinstrahlung. Nach de Miguel et al. [23] kann daraus allerdings die direkte und die indirekte Einstrahlung über die atmosphärische Transmissivität abgeleitet werden. Für eine Transmissivität von mehr als 0.76 folgt beispielsweise $S_{i0} = S_{Gesamt} \cdot 0.18$ und $S_o = S_{Gesamt} - S_{i0}$.

6.3.3. Wahl des Ambient Occlusion Verfahrens

Zusätzlich zu den direkten Schatten einer Strahlungsquelle wird bei diesem Verfahren ein Verdeckungsfaktor der Atmosphäre berechnet. Dieser Verdeckungsfaktor wird mit einem Ambient Occlusion Verfahren bestimmt. Dabei werden folgende Algorithmen betrachtet:

- Screen Space Ambient Occlusion (SSAO)
- Horizon Based Ambient Occlusion (HBAO)
- Volumetric Obscurance (VO)

6.3.3.1. Screen Space Ambient Occlusion

Bei Screen Space Ambient Occlusion [59] wird eine Kugel um jeden Pixel gelegt und überprüft, wie viel Prozent der Kugel verdeckt sind. Der resultierende Verdeckungsfaktor ergibt sich direkt aus dem Verhältnis der verdeckten Punkte zur Gesamtzahl der Abtastpunkte. Um einen weichen Grauwerteübergang zu gewährleisten, sollten daher mindestens 32 Abtastpunkte benutzt werden. Alternativ kann der Verdeckungsfaktor über einen Tiefpassfilter weichgezeichnet werden (siehe [59]). Ein Nachteil dieses Verfahrens besteht darin, dass der Kugelradius, indem die Abtastpunkte verteilt werden, durch den Benutzer gewählt wird. Ein falsch gewählter Radius kann dazu führen, dass falsche Verdeckungen entstehen oder relevante Schattenspender übersprungen werden.

6.3.3.2. Horizon Based Ambient Occlusion

Bei Horizon Based Ambient Occlusion (HBAO) [11] wird der Verdeckungsfaktor über den sichtbaren Öffnungswinkel des Horizonts abgeschätzt. Um den Öffnungswinkel zu ermitteln, wird ein Ray Marching auf der Tiefenkarte durchgeführt. Durch die Berechnung des Verdeckungsfaktors über den Öffnungswinkel wird ein weicher Grauwerteübergang generiert.

Das Ray Marching wird durch den Benutzer mit der Vorgabe einer Schrittweite und der Anzahl der Raumrichtungen konfiguriert. Auch hier können falsche Verdeckungen

entstehen oder relevante Schattenspender übersprungen werden. Durch die Ermittelung des minimalen Öffnungswinkels ist das Verfahren allerdings robuster gegenüber falsch gewählter Parameter.

6.3.3.3. Volumetric Obscurance

Bei Volumetric Obscurance (VO) [55] wird der Verdeckungsfaktor über das Volumen der Sichtkugel abgeschätzt. Dazu werden Line Samples einer Kugel erstellt und jedem Line Sample ein Volumen der Sichtkugel zugeordnet. Durch diese Vorgehensweise kann selbst mit wenig Abtastpunkten ein weicher Grauwerteübergang generiert werden. Eine weitere Besonderheit dieses Verfahrens besteht darin, dass durch gepaarte Abtastpunkte, sehr effektiv verdeckte Abtastpunkte ausgeschlossen werden können. Allerdings ist dieses Verfahren, wie SSAO, anfällig für falsch gewählte Kugelradien.

6.3.3.4. Auswahl des Verfahrens

Nach Abwägung der Vor- und Nachteile der einzelnen Verfahren wird Volumetric Obscurance eingesetzt. Der Hauptvorteil dieses Verfahrens besteht darin, dass mit wenig Abtastpunkten ein weicher Grauwerteübergang generiert werden kann. Im Worst-Case muss der Verdeckungsfaktor für jeden Zeitschritt der Simulation berechnet werden. Dazu ist ein schnelles Verfahren notwendig. Des Weiteren können falsche Verdeckungen bei der Abtastung effizient ausgeschlossen werden.

6.3.4. Implementierung des Algorithmus

Als Basis wird auf die Implementierung des thermischen Schattenalgorithmus aus Kapitel 4 zurückgegriffen. Die Implementierung wird um einen Renderpass für Ambient Occlusion erweitert. Das Ablaufdiagramm der Implementierung wird in Abbildung 6.2 dargestellt.

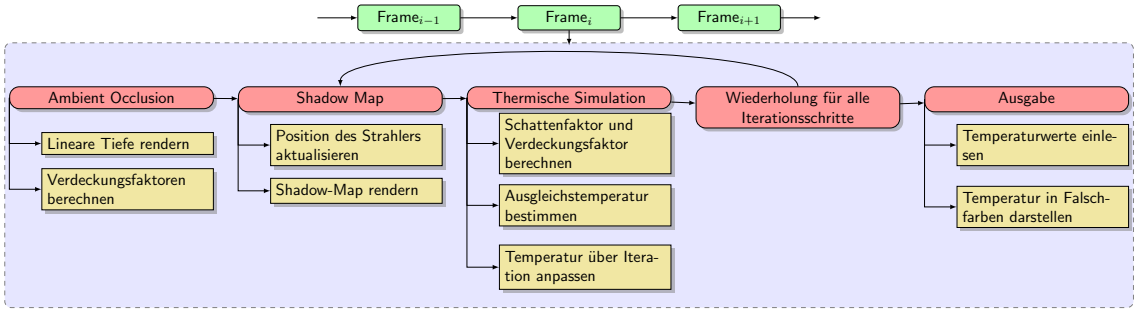


Abbildung 6.2.: Ablaufdiagramm der Implementierung

Die Implementierung wird, wie zuvor, als Forward Renderer und als Deferred Renderer umgesetzt. Da die Verdeckungsfaktoren von der Geometrie abhängen, muss Ambient Occlusion jedes Mal neu berechnet werden, wenn sich die Geometrie der Szene ändert.

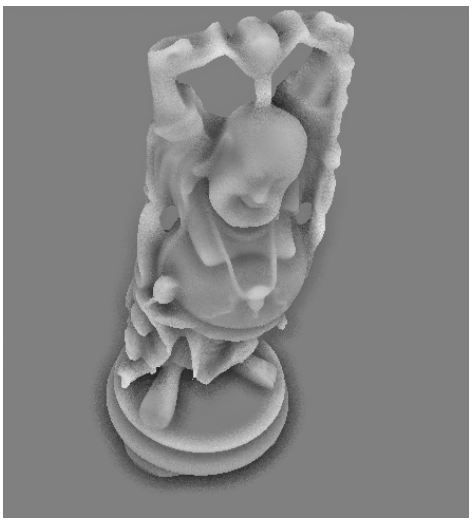


Abbildung 6.3.: Verdeckungsfaktoren bei der Buddha-Szene

6.3.4.1. Ambient Occlusion

Um die Verdeckungsfaktoren zu berechnen wird zuerst ein linearer Tiefenwert aus der Betrachtersicht gerendert und in eine Textur gespeichert. Beim Deferred Renderer kann dieser Renderpass eingespart werden, da der linearisierte Tiefenwert bereits im G-Buffer gespeichert wird. Anschließend wird für jeden Pixel der Verdeckungsfaktor über Volumetric Obscurance bestimmt und in eine Textur gespeichert.

Die Volumetric Obscurance Implementierung folgt dem Vorgehen von Ownby et al. [69]. Die Abtastpunkte werden durch eine zweidimensionale Gleichverteilung erstellt. Die dazugehörigen Volumina werden anschließend über eine Integration des Kugelvolumens mit Hilfe eines Voronoi-Diagrams bestimmt. Bei der Berechnung des Verdeckungsfaktors wird jeder Abtastpunkt als Paar ausgewertet, um verdeckte Abtastpunkte zu invalidieren.

Im Gegensatz zur Implementierung im visuellen Spektrum, wird für die indirekte Wärmestrahlung der Verdeckungsfaktor so skaliert, dass unverdeckte Pixel einen Verdeckungsfaktor von 0.5 besitzen. Dies bedeutet, dass die Hemisphäre vollkommen sichtbar ist. Ein Faktor von < 0.5 bedeutet, dass die Hemisphäre verdeckt und ein Faktor von > 0.5 das mehr als die Hälfte der Sphäre sichtbar ist. Dies ist beispielsweise an 90 Grad Kanten der Fall. Abbildung 6.3 zeigt ein Beispiel der resultierenden Verdeckungsfaktoren bei der Buddha-Szene.

6.3.4.2. Thermische Simulation

Bei der thermischen Simulation wird zuerst, für jeden Pixel, der Schattenfaktor über PCSS bestimmt und der Verdeckungsfaktor aus der Textur geladen. Anschließend wird die Umgebungstemperatur nach Formel 6.2 berechnet und dabei die Materialparameter, wie z.B. die Albedo, aus der Lookup-Tabelle geladen. Bei der Berücksichtigung

der Oberflächenorientierung ist ein gemittelter Normalenvektor der angrenzenden Flächen erforderlich, um einen weichen Verlauf der Temperaturen zu ermöglichen. Aus der Umgebungstemperatur wird die Ausgleichstemperatur nach Formel 6.6 bestimmt. Anschließend werden die Zeitkonstanten und die Gewichtungsfaktoren des thermischen Modells aus der Lookup-Tabelle gelesen und die Oberflächentemperatur angepasst.

6.3.4.3. Anpassung des Materialsystems

Um den Algorithmus umzusetzen ist eine Anpassung des Materialsystems notwendig. Ein Material besteht aus den folgenden Feldern:

- Schichtdicke d.
- Albedo α.
- Emissivität ϵ.
- Wärmeleitfähigkeit λ.
- Kerntemperatur T_c.
- Einen zeitlichen Verlauf der Zeitkostanten und Gewichtungsfaktoren des thermischen Modells.

Des Weiteren wird ein Strahlungsprofil referenziert, dass die Wärmeströme aus Tabelle 6.1 beinhaltet. Beim Erstellen des Szenengraphen legt ein Node-Visitor diese Parameter in die UserData Felder des Szenengraphen ab. Während des Einlesens des Szenengraphen für die thermische Simulation werden die UserData Felder ausgewertet. Für den Forward-Renderer werden die Materialparameter als Uniform-Variable angelegt. Der zeitliche Verlauf der Zeitkonstanten und Gewichtungsfaktoren wird in eine 1D-Textur übertragen. Beim Deferred-Renderer werden die alle Parameter in ein 1D-Texture Array abgelegt und der Index in dieses Array in den G-Buffer gespeichert.

6.3.5. Mehrere Wärmequellen

Ein Vorteil dieses Algorithmus besteht darin, dass mehrere Wärmequellen in der Szene unterstützt werden. Damit ist es möglich Abgasstrahlen von Motoren oder ein Feuer zu modelliert. Diese Wärmequellen werden wie Lichtquellen bei der Beleuchtungsrechnung behandelt. Sie können gerichtet oder lokal sein und durch Abschwächungsfaktoren begrenzt werden. Ebenso lässt sich ein Schatten der Wärmequelle durch einen Schattenfaktor realisieren. Die Beiträge dieser Wärmequellen müssen bei der Umgebungstemperatur berücksichtigt werden.

In Abbildung 6.4 wird ein Beispiel einer lokalen Wärmequelle mit 1000 Watt und einer Kältequelle mit 500 Watt dargestellt, die jeweils über einen quadratischen Abschwächungsfaktor begrenzt werden.

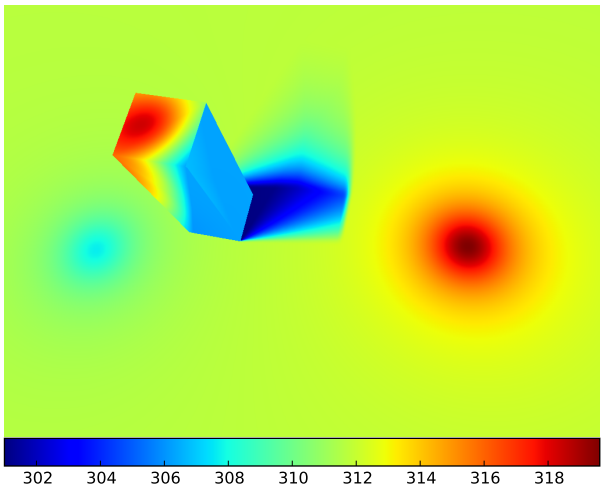


Abbildung 6.4.: Lokale Wärmequelle mit 1000 Watt und Kältequelle mit 500 Watt.

6.4. Ergebnisse

Im Folgenden wird auf die Ergebnisse des Algorithmus eingegangen. Dazu wird zuerst eine Fehlerabschätzung durchgeführt und anschließend die Geschwindigkeit und der Speicherverbrauch analysiert.

6.4.1. Fehlerabschätzung

Eine Fehlerquelle besteht darin, dass eine homogene Umgebungsstrahlung für alle Objekte angenommen wird. Dadurch werden gerichtete, lokale Aufheizungseffekte aus der Umgebung vernachlässigt, wie z.B. ein heißes Objekt in der Umgebung. Diese Aufheizungseffekte können allerdings durch eine Modellierung mit lokalen Wärmequellen simuliert werden. Des Weiteren wird in Kapitel 7 eine Idee vorgestellt um diesen Fehler zu minimieren.

Eine weitere Fehlerquelle liegt in der Berechnung der Ausgleichstemperatur. Dabei wird die Annahme getroffen, dass eine Oberfläche nur aus einer Schicht besteht. Dadurch wird der Wärmeaustausch mit unteren Schichten vernachlässigt. Dies führt vorallem bei längeren Simulationen zu einem Fehler. Daher ist der Algorithmus nur für kurze Simulationszeiten geeignet. Eine sinnvolle Simulationszeit kann über die Eindringtiefe der Wärme abgeschätzt werden:

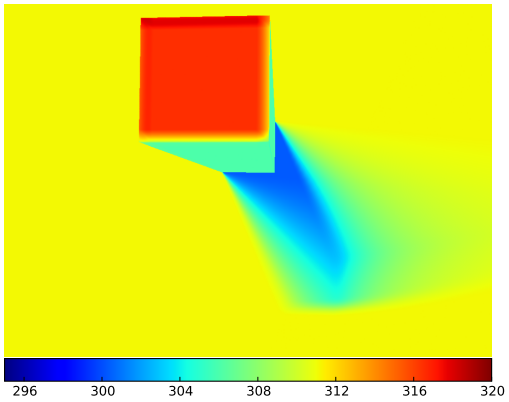
$$t = \frac{d^2 \rho c}{2 \cdot \lambda}$$

Wobei d die Eindringtiefe, λ die Wärmeleitfähigkeit, ρ die Dichte und c die Wärmekapazität ist. Wenn die Eindringtiefe die Schichtdicke übersteigt, muss von einem Einfluss der unteren Oberflächenschichten ausgegangen werden.

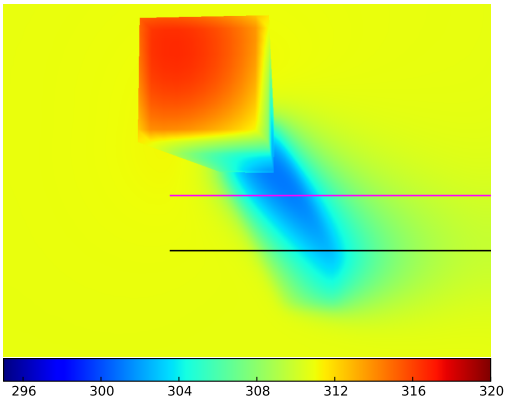
6.4.2. Vergleich der absoluten Temperaturen

In diesem Vergleich werden die absoluten Temperaturen gegenüber einer Referenzlösung mit RadTherm verglichen. Für beide Simulationen werden die gleichen Atmosphärendaten benutzt. Die Parameter für das thermische Modell werden über eine Kurvenanpassung gegen die RadTherm Lösung ermittelt. Wie zuvor im Kapitel 4.4.1 wird die Simulationszeit zwischen 8:00 - 11:30 Uhr mit einen Zeitschritt von 60 Sekunden gewählt. Die RadTherm Lösung wird mit einer Simulationszeit von 0:00 bis 11:30 Uhr berechnet. Die Normalenvektoren des Quaders werden innerhalb eines Winkels von 100° gemittelt.

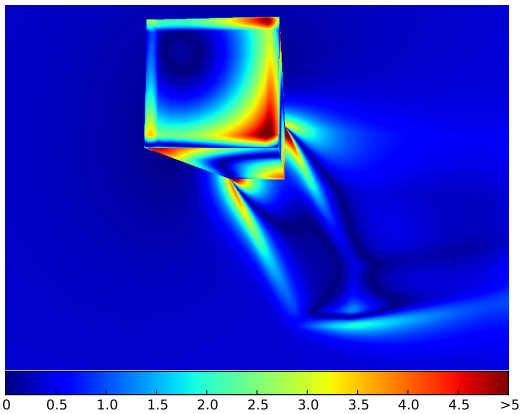
Für die Asphaltoberfläche wird eine Wärmeleitfähigkeit von $\lambda = 0.7$, eine Schichtdicke von 15 cm, ein Albedo von 0.75 und ein Emissionsgrad von $\varepsilon = 0.93$ angenommen. Die Wärmeleitfähigkeit für die Zementoberfläche wurde mit $\lambda = 1.28$ definiert, eine Schichtdicke von 10 cm, ein Albedo von 0.6 sowie ein Emissionsgrad von $\varepsilon = 0.88$ gewählt.



(a) Ergebnis unserer Approximation.



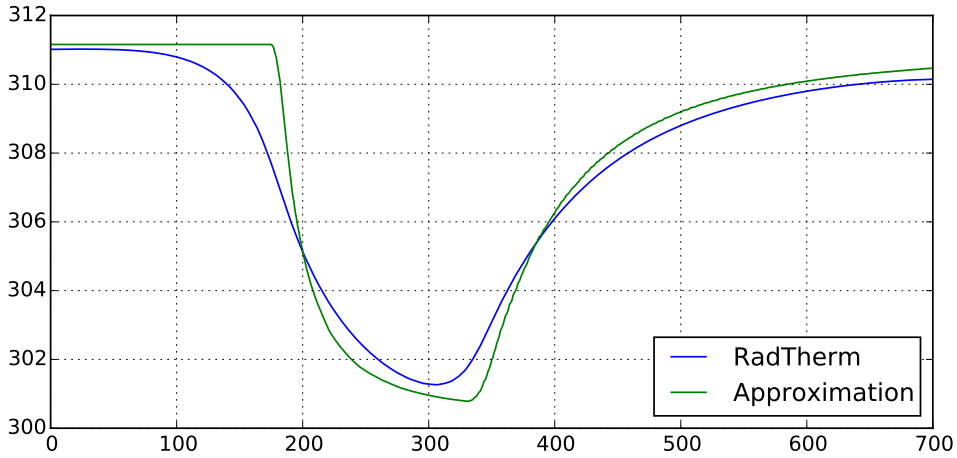
(b) Ergebnis von RadTherm. Die Linien zeigen die Stellen für die Temperaturkurven an.



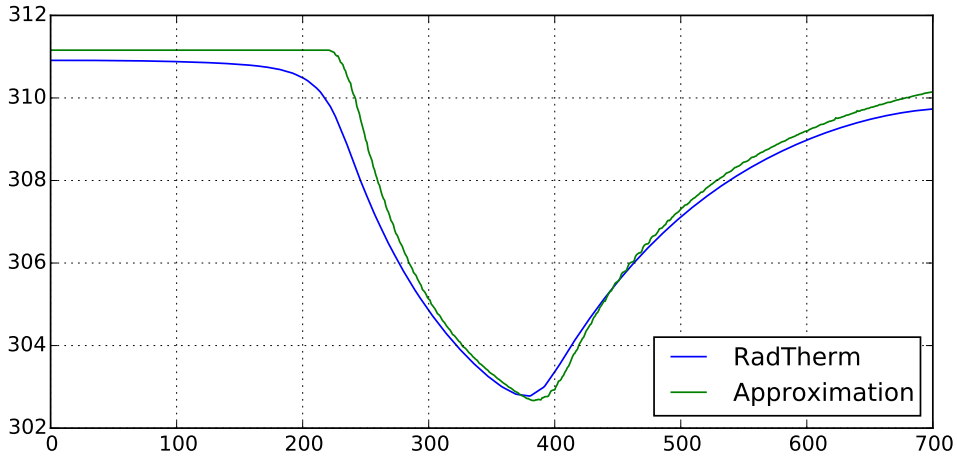
(c) Differenzbild zwischen (a) und (b).

Abbildung 6.5.: Ergebnis unserer Approximation im Vergleich zur RadTherm Lösung.

Abbildung 6.5 vergleicht das Ergebnis unserer Approximation mit der RadTherm-Lösung und stellt die Temperaturunterschiede als Differenzbild dar. Beim Vergleich der Bilder fällt auf, dass nun Temperaturübergänge durch den Einfluss der Oberflächenorientierung entstehen. Allerdings sinkt die Temperatur im Schatten gegenüber der Referenzlösung weiter ab. Das liegt daran, dass in diesem Modell angenommen wird, dass eine Oberfläche nur aus einer Schicht besteht. Die Wärme durchdringt die Asphaltschicht bereits nach etwa 3.2 Stunden komplett und gleicht sich mit den tieferen Schichten aus. Dies wird in der Ausgleichstemperatur nicht berücksichtigt. Beim Algorithmus aus Kapitel 4 ist dieser Beitrag in der vorberechneten Schattentemperatur enthalten. Daher ist die Ausgleichstemperatur im Gegensatz zur RadTherm-Lösung zu kalt.



(a) Temperaturkurve in der magentafarbenen Bildzeile.



(b) Temperaturkurve in der schwarzen Bildzeile.

Abbildung 6.6.: Temperaturvergleich in den markierten Bildzeilen aus Abbildung 6.5b

Abbildung 6.7.: An Kanten ist ein größerer Raumwinkel der Atmosphäre sichtbar, daher steigt die Temperatur schneller an. An Vertiefungen ist dieser Effekt umgekehrt.

In Abbildung 6.6 werden die Temperaturkurven dargestellt. Die Temperaturen im Kernschatten unterscheiden sich um bis zu 2.5 K. Nichtsdestotrotz kann die Approximation die Referenzlösung mit einer akzeptablen Abweichung repräsentieren.

Diese Auswertung wird für weitere Materialien im Anhang A.5 durchgeführt.

6.4.3. Visuelle Effekte der atmosphärischen Verdeckung

Abbildung 6.7 zeigt die visuellen Effekte der atmosphärischen Verdeckung. Wenn die Umgebungstemperatur wärmer als die Oberflächentemperatur ist, heizen sich die Kanten schneller auf. Das liegt an einem größeren Beitrag des indirekten Wärmestroms durch einen sichtbaren Raumwinkel von mehr als 2π sr der Sichtkugel. Analog sind Vertiefungen kühler als die Umgebung, da der indirekte Wärmestrom durch einen kleineren Raumwinkel abgeschwächt wird.

6.4.4. Geschwindigkeitsanalyse

Die Geschwindigkeitsanalyse wird auf einen Intel i7-3820 Prozessor mit 3,6 GHz, 16 GB RAM und einer NVIDIA GeForce GTX Titan mit 6144 MB Speicher durchgeführt. Die Schatten werden mit PCSS mit 8 Abtastpunkten für die Blockersuche und 32 Abtastpunkte für PCF gerendert. Die Auflösung beträgt 1024x768 Pixel und die Shadow Map Auflösung ist 1024x1024. Für Volumetric Obscurance werden 13 gepaarte Abtastpunkte benutzt. Die Angaben zur Rechendauer sind jeweils die Mittelwerte aus 1000 Bildern.

Tabelle 6.2 zeigt die Rechendauer bei einer gleichmäßigen Abtastung. Zusätzlich wird für jede Implementierung der Worst-Case gemessen, bei dem das Ambient Occlusion in

jedem Zeitschritt erneut gerendert werden muss. Beim Deferred-Renderer kann, durch den G-Buffer, das Rendern des linearen Tiefenwerts für Ambient Occlusion eingespart werden. Dadurch ist die Implementierung für den Worst-Case Fall im Vergleich zum Forward-Renderer um 30 ms schneller.

Szene	**Forward**	**Deferred**	**Forward (Worst-Case)**	**Deferred (Worst-Case)**
Quader	23.6 ms	25.2 ms	74.2 ms	84.1 ms
Buddha	96.4 ms	56.8 ms	175.4 ms	145.0 ms

Tabelle 6.2.: Rechendauer bei den Testszenen und einer gleichmäßigen Abtastung mit 90 Schritten.

In Tabelle 6.3 wird die Rechendauer für eine exponentielle Abtastung einer Simulationsdauer von 9:00 - 12:00 Uhr mit einen Zeitschritt von zwei Minuten dargestellt. Die Halbschattenbreite wird ebenfalls exponentiell angepasst. Für die 27 Simulationsschritte erreicht der Deferred Renderer bei der Buddha-Szene im besten Fall eine Framerate von 54 Bildern pro Sekunde. Das Verfahren erfüllt mit den vorgegebenen Testparametern daher nicht die geforderten, harten Echtzeitbedingungen von 60 Bildern pro Sekunde. Die Geschwindigkeit kann allerdings durch eine Mehrfachbenutzung von Shadow Maps erhöht werden (siehe Kapitel 4.3.10).

Szene	**Forward**	**Deferred**	**Forward (Worst-Case)**	**Deferred (Worst-Case)**
Quader	7.7 ms	8.4 ms	22.5 ms	25.5 ms
Buddha	29.7 ms	18.2 ms	52.4 ms	43.6 ms

Tabelle 6.3.: Rechendauer bei den Testszenen und einer exponentiellen Abtastung mit 27 Schritten.

Die Rechendauer der einzelnen Renderpässe wird in Tabelle 6.4 dargestellt. Der Ambient Occlusion Renderpass benötigt beim Forward Renderer 1.8 ms. Beim Deferred-Renderer wird diese Dauer, durch das Rendern der linearen Tiefe des G-Buffers, reduziert. Im Vergleich zur Implementierung der thermischen Schatten aus Kapitel 4 fällt auf, dass die thermische Simulation nun länger rechnet. Das liegt zum einen daran, dass je Material jetzt insgesamt acht Parameter aus den Lookup-Tabellen geladen werden müssen. Zusätzlich muss der indirekte Verdeckungsfaktor aus einer Textur geladen werden. Zum anderen erfordert die Berechnung der Umgebungs- und Ausgleichstemperatur (Formel 6.2 und 6.6) deutlich mehr Rechenoperationen.

6.4.5. Speicherverbrauch auf der Grafikkarte

Eine Lookup-Tabelle ist nun 14 Kilobyte groß. In Tabelle 6.5 wird der Speicherverbrauch des Algorithmus dargestellt.

Für die atmosphärische Verdeckung sind drei zusätzliche Texturen für das Ambient Occlusion notwendig. Die erste Textur beinhaltet einen linearisierten Tiefenwert. Diese

Schritt	Forward	Deferred
G-Buffer	-	0.7 ms
Ambient Occlusion	1.8 ms	1.1 ms
Shadow Maps	36.2 ms	36.2 ms
Thermische Simulation	57.7 ms	18.1 ms
Darstellung	0.7 ms	0.7 ms
Gesamt	96.4 ms	56.8 ms

Tabelle 6.4.: Dauer der einzelnen Renderpässe beim Forward und Deferred Rendering Ansatz in der Buddha-Szene mit 90 Simulationsschritten.

	Speicherverbrauch [MB]			
Implementierung	**Shadow-Maps**	**Texturen**	**Lookup-Tabellen**	**Gesamt**
Forward(osgShadow)	87	42.03	0.03	129.06
Forward (Shadow-Map)	3	42.03	0.03	45.06
Deferred (osgShadow)	87	63.03	0.03	150.06
Deferred (Shadow-Map)	3	63.03	0.03	66.06

Tabelle 6.5.: Speicherverbrauch bei 29 Simulationsschritten für zwei Materialien bei einer Bildschirmauflösung von 1024x768 und einer Shadow-Map Auflösung von 1024x1024.

kann jedoch im Deferred-Rendering Ansatz eingespart werden. In der zweiten Textur werden die Verdeckungsfaktoren in eine 32-Bit Float 2D-Textur gespeichert. Die dritte Textur ist eine 64x64 32-Bit RG Float 2D-Textur, die für die Zufallszahlen benötigt wird. Des Weiteren sind die Lookup-Tabellen größer, da nun die Strahlungsbeiträge sowie die Parameter des thermischen Modells gespeichert werden.

6.5. Diskussion

In diesem Kapitel wurde ein Algorithmus für die Berechnung von thermischen Schatten unter der Berücksichtigung einer atmosphärischen Verdeckung realisiert. Im direkten Vergleich zum Algorithmus aus Kapitel 4 kann das Fazit gezogen werden, dass die direkte Berechnung der Ausgleichstemperatur aus den Strahlenbeiträgen deutliche Vorteile, gegenüber einer Vorberechnung von Temperaturwerten, bietet. So können zum einen sehr einfach neue Materialien erzeugt werden, in dem die Materialparameter aus der Literatur verwendet werden. Das gilt insbesondere für komplexe Geometrie, da nun die Oberflächennormale berücksichtigt wird. Zum anderen können weitere Wärmequellen berücksichtigt und dadurch lokale Aufheizungseffekte simuliert werden.

Die Annahme, dass die Oberfläche nur aus einer Schicht besteht, schränkt den Algorithmus jedoch für kurze Simulationszeiten ein. Ebenfalls werden die lateralen und konvektiven Wärmetransportmechanismen vernachlässigt. Wenn diese Mechanismen

überwiegen, ist mit einer deutlichen Abweichung der Oberflächentemperaturen zu rechnen. Allerdings können die Temperaturunterschiede, durch mehr Freiheiten bei der Modellierung der Materialien, minimiert werden. Eine starke Wärmeleitfähigkeit kann beispielsweise durch eine Erhöhung der Kerntemperatur realisiert werden (siehe Anhang A.5). Eine weitere Möglichkeit um diese Abweichung zu reduzieren, besteht bei eine Erweiterung des Algorithmus um Temperaturgradienten. Ein Konzept dazu wird in Kapitel 7 vorgestellt.

Im Geschwindigkeitsvergleich zum Algorithmus aus Kapitel 4, wird mehr Rechenzeit pro Simulationsschritt benötigt. Das liegt vor allem daran, dass nun ein Ambient Occlusion zusätzlich berechnet wird und die Berechnung der Ausgleichstemperatur aufwändiger ist. Das hat zur Folge, dass dadurch weniger Zeitschritte in Echtzeitanwendungen berechnet werden können. Allerdings können alle vorgestellten Techniken zur Geschwindigkeitsoptimierung, wie z.B. die Mehrfachbenutzung von Shadow Maps oder die exponentielle Abtastung, integriert werden. Da die Vorgehensweise auf dem vorherigen Ansatz basiert, treffen die Einschränkungen bezüglich animierter Objekte zu.

7. Fazit und weitere Arbeiten

Diese Dissertation beschäftigt sich mit der Frage, ob thermische Schatten für 3D-Szenen unter Echtzeitbedingungen simuliert werden können. Diese Frage kann mit Ja beantwortet werden. Allerdings müssen dabei einige Einschränkungen gemacht werden.

Für die Simulation von thermischen Schatten ist eine Approximation der Wärmebilanzrechnung notwendig. In dieser Arbeit wurde ein thermisches Modell erstellt, dass das Temperaturverhalten von Oberflächen mit zwei Exponentialfunktionen beschreibt. Dieses thermische Modell kann mit, für diesen Einsatzzweck, akzeptabler Genauigkeit eine Referenzlösung nachbilden. Um die notwendige Kurvenanpassung durchzuführen, muss allerdings für jedes Material eine Referenzlösung berechnet werden. Dies kann, je nach berücksichtigten Energiebeiträgen, sehr zeitintensiv ausfallen. Des Weiteren entsteht die größte Abweichung oft in der Kurzzeitlösung. Also bei der Lösung, die zu einer schnellen Temperaturveränderung, und daher zu einer schnellen Kontrastveränderung im Bild, führt. Bei Tracking- und Navigationsalgorithmen könnte dies zu einer Störung führen. Allerdings werden thermische Schatten in Echtzeitanwendungen oft vorberechnet oder vernachlässigt. Daher überwiegt der allgemeine Gewinn an Simulationsgenauigkeit dieser Einschränkung.

Eine weitere Einschränkung, bei der durchgeführten Wärmebilanzrechnung, besteht in den Ansätzen zur Berechnung der Ausgleichstemperatur. Bei der Simulation von thermischen Schatten mit vorberechneter Strahlungs- und Schattentemperatur wird nur der direkte Wärmestrom dynamisch variiert. Die anderen Anteile, wie z.B. der indirekte Wärmestrom der Atmosphäre, fließen in die vorberechneten Temperaturen ein. Ein Nachteil dieser Vorgehensweise ist allerdings, dass keine weiteren Wärmequellen dynamisch berücksichtigt werden können. Die Auswertung hat gezeigt, dass mit diesem Ansatz ein thermischer Schatten mit einer geringen Abweichung zu einer Referenzlösung mit RadTherm dargestellt werden kann. Daher ist dieser Ansatz besonders für den direkten Schattenwurf einer Geometrie auf eine Oberfläche geeignet. Im zweiten Ansatz wurde die indirekte Umgebungsstrahlung durch eine atmosphärische Verdeckung berücksichtigt. Dabei wird allerdings angenommen, dass die indirekte Umgebungsstrahlung homogen ist. Des Weiteren nimmt dieses Modell nur eine Oberflächenschicht an. Allerdings erlaubt dieser Ansatz mehrere Wärmequellen und berücksichtigt die Orientierung der Oberfläche. Die Auswertungen haben gezeigt, dass auch in diesen Ansatz eine akzeptable Temperaturabweichung zu einer Referenzlösung mit RadTherm entsteht. Durch mehr Freiheiten bei der Modellierung von Materialien können Temperaturunterschiede weiter reduziert werden.

Die Analysen bezüglich der Rechendauer haben gezeigt, dass der Flaschenhals der vorgestellten Ansätze jeweils in den Computergrafik-Algorithmen zur Schattenberechnung liegt. Für jeden Zeitschritt der Simulation muss eine Shadow Map erzeugt werden. Die Berechnung kann beschleunigt werden, wenn Shadow Maps für mehrere Simulationsschritte genutzt werden. Dieses Konzept wurde durch ein allgemeines Clustering von

Lichtquellen erweitert. Eine weitere Möglichkeit, um die Berechnung zu beschleunigen, besteht bei der exponentiellen Abtastung des Simulationszeitraums. Damit werden die Simulationsschritte vorwiegend in der jüngeren Vergangenheit berechnet.

Zusammenfassend kann das Fazit gezogen werden, dass mit diesen Ansätzen bereits jetzt eine Simulation von thermischen Schatten für 3D-Szenen mit dynamischer Geometrie unter Echtzeitbedingungen möglich ist. Dabei müssen allerdings die Anzahl der Simulationsschritte beschränkt werden. Durch die steigende Leistung neuer Grafikkartengenerationen ist davon auszugehen, dass diese Beschränkung in Zukunft gelockert werden kann.

Weitere Arbeiten

Da der Flaschenhals der vorgestellten Ansätze in der 3D-Schattenberechnung liegt, könnte ein komplexeres thermisches Modell eingesetzt werden. Beispielsweise kann eine Fourier-Reihe mit wenigen Reihengliedern benutzt werden. Diese Vorgehensweise hätte den Vorteil, dass sämtliche Stoffwerte von Materialien direkt benutzt werden können und somit eine Kurvenanpassung entfällt.

Eine weitere Arbeit besteht darin, die Temperaturgradienten der Oberfläche bei der thermischen Simulation zu berücksichtigen. Dadurch kann das Temperaturverhalten in den unteren Schichten genauer berücksichtigt werden. Die Temperaturgradienten können bei der Vorberechnung der Strahlungs- und Schattentemperatur, oder bei einer RadTherm Simulation, berechnet werden. Diese werden anschließend in den Materialdaten abgelegt und das Temperaturverhalten abhängig vom Gradienten angepasst werden. Dadurch könnte auch ein weiterer offener Punkt berücksichtigt werden und zwar, ob die Parameter der Kurzzeitlösung, wie die Parameter der Langzeitlösung, direkt durch die Stoffwerte der Oberfläche ermittelbar sind. Das Kurzzeitverhalten der Oberfläche hängt stark von den Temperaturgradienten ab. Wenn die Temperaturgradienten bekannt sind, können diese dazu genutzt werden die Exponentialfunktion der Langzeitlösung zu modifizieren.

Um die Einschränkungen bezüglich eines manuellen Wärmeeintrags in eine Szene zu lösen, kann der Wärmeeintrag über ein Splatting-Verfahren berücksichtigt werden. Dazu wird, analog zum Photon Mapping, eine Wärmekarte der Szene über einen kd-Baum erzeugt. Jeder Wärmeeintrag kann nun in die Wärmekarte abgelegt und aktualisiert werden. Bei der Berechnung der Ausgleichstemperatur wird auf die Wärmekarte zugegriffen und der Betrag zur Strahlungsbilanz addiert. Dadurch ist es auch möglich, zeitliche Effekte zu realisieren, die außerhalb des Kamerasichtfelds stattfinden.

Die visuellen Ergebnisse haben gezeigt, dass die thermische Simulation oft Ungenauigkeiten durch eine fehlerhafte Schattendarstellung erzeugt. Die Qualität der Schattendarstellung kann in weiteren Arbeiten erhöht werden, wie z.B. mit Exponential Soft Shadow Mapping [89]. Eine weitere Arbeit besteht darin, einen Algorithmus zur Reduzierung von Biasing-Artefakten des Shadow Mappings zu implementieren. Dou et al. [26] präsentieren dazu ein adaptives Verfahren, das einen optimalen Bias erzeugt.

Eine Möglichkeit, um die Genauigkeit bei der Simulation der atmosphärischen Verdeckung zu erhöhen, besteht in der Berücksichtigung einer inhomogenen Umgebungsstrahlung. Dazu kann ein Directional Occlusion Verfahren [83] eingesetzt werden, dass die Umgebungsstrahlung der Hemisphäre mit Hilfe einer Environment Map repräsentiert. Diese Environment Map muss vorab berechnet werden. Dadurch können dann Effekte, wie z.B. die Strahlung einer heißen Hauswand auf eine andere Oberfläche, realisiert werden. Diese Vorgehensweise kann durch ein Radiosity-Verfahren erweitert werden, um reflektierte Wärmestrahlung zu berücksichtigen. Allerdings eignen sich diese Verfahren im Allgemeinen nicht für Echtzeitanwendungen.

Als Alternative zum Shadow Mapping und Ambient Occlusion könnte, in einer weiteren Arbeit, Voxel Cone Tracing [20] eingesetzt werden. Die Idee besteht darin, eine Szene in jedem Frame in grober Auflösung zu voxelisieren und globale Beleuchtungseffekte über Cone Tracing zu realisieren. Die Cones können dabei mit Hilfe von MipMapping effizient umgesetzt werden. Ein Vorteil dieses Ansatzes besteht darin, dass damit auch Reflexionen umgesetzt werden können. Dadurch könnte der indirekte Wärmestrom der Umgebung genauer mit einbezogen werden. Des Weiteren lässt eine Voxel-Struktur eine Berechnung einer instationären Wärmeleitung mittels der Finiten-Volumen-Methode zu [56].

Danksagungen

Die Sponza, Hairball, Buddha und Cornell-Box Szene wurde vom Computer Graphics Archive [58] heruntergeladen. Die Restauraunt-Szene basiert auf dem Restaurant-Modell von [43].

Literaturverzeichnis

[1] Abramowitz, Milton und Stegun, Irene A. *Handbook of Mathematical Functions: With Formulas, Graphs, and Mathematical Tables*. Dover Publications, 1972. ISBN 9780486612720.

[2] Agrawala, Maneesh, Ramamoorthi, Ravi, Heirich, Alan, und Moll, Laurent. *Efficient image-based methods for rendering soft shadows*. In *Proceedings of the 27th annual conference on Computer graphics and interactive techniques*, SIGGRAPH '00, Seiten 375–384. ACM Press/Addison-Wesley Publishing Co., New York, NY, USA, 2000. ISBN 1-58113-208-5. doi:10.1145/344779.344954.

[3] Aguado, Alberto und Montiel, Eugenia. *Mipmapped screen space soft shadows*. In Wolfgang Engel, Herausgeber, *GPU Pro 2 - Advanced Rendering Techniques*, Seiten 257–272. A.K. Peters, 2011. ISBN 1568817185.

[4] Alfred Nischwitz, Peter Haberäcker und Gudrun Socher, Max Fischer. *Computergrafik und Bildverarbeitung*. Vieweg und Teubner Verlag, 2012. ISBN 9783834813046.

[5] AMD. *CodeXL: Powerful Debugging, Profiling and Analysis*, 2014. URL `http://developer.amd.com/tools-and-sdks/opencl-zone/opencl-tools-sdks/codexl/`. Version 1.4. Zugegriffen am 08.07.2014.

[6] Anderson, Gail P., Berk, Alexander, Acharya, Prabhat K., Matthew, Michael W., Bernstein, Lawrence S., Chetwynd, James H., Jr., Dothe, H., Adler-Golden, Steven M., Ratkowski, Anthony J., Felde, Gerald W., Gardner, James A., Hoke, Michael L., Richtsmeier, Steven C., Pukall, Brian, Mello, Jason B., und Jeong, Laila S. *Modtran4: radiative transfer modeling for remote sensing. Proc. SPIE*, 3866:2–10, 1999. doi:10.1117/12.371318.

[7] Andersson, Johan. *Parallel graphics in frostbite - current and future*. In *Beyond programmabel shading, SIGGRAPH 2009*. 2009.

[8] Annen, Thomas, Mertens, Tom, Seidel, Hans-Peter, Flerackers, Eddy, und Kautz, Jan. *Exponential shadow maps*. In *Proceedings of Graphics Interface 2008*, GI '08, Seiten 155–161. Canadian Information Processing Society, Toronto, Ont., Canada, Canada, 2008. ISBN 978-1-56881-423-0.

[9] Arvo, Jukka, Hirvikorpi, Mika, und Tyystjärvi, Joonas. *Approximate soft shadows win an image-space flood-fill algorithm. Computer Graphics Forum*, 23(3):271–279, 2004. ISSN 1467-8659. doi:10.1111/j.1467-8659.2004.00758.x.

[10] Baehr, Hans Dieter und Stephan, Karl. *Heat and Mass Transfer*. Springer, Berlin, Deutschland, 2006. ISBN 3540295267.

[11] Bavoil, Louis, Sainz, Miguel, und Dimitrov, Rouslan. *Image-space horizon-based ambient occlusion*. In *ACM SIGGRAPH 2008 Talks*, SIGGRAPH '08, Seiten

22:1–22:1. ACM, New York, NY, USA, 2008. ISBN 978-1-60558-343-3. doi: 10.1145/1401032.1401061.

[12] Biesel, Heiner und Rohlfing, Tom. *Real-Time Simulated Forward Looking Infrared (FLIR) Imagery For Training. Proc. SPIE*, 0781:71–80, 1987. doi: 10.1117/12.940535.

[13] Bonnans, Frédéric J., Gilbert, Jean Charles, Lemaréchal, Claude, und Sagastizábal, Claudia A. *Numerical Optimization: Theoretical and Practical Aspects (Universitext)*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2006. ISBN 354035445X.

[14] Brabec, Stefan, Annen, Thomas, und Seidel, Hans-Peter. *Shadow mapping for hemispherical and omnidirectional light sources*. In John Vince und Rae Earnshaw, Herausgeber, *Advances in Modelling, Animation and Rendering*, Seiten 397–407. Springer London, 2002. ISBN 978-1-4471-1118-4. doi:10.1007/978-1-4471-0103-1_25.

[15] Cathcart, M. J. *Thermal Shadow Modelling within a Physically-Based Scene Simulation. SPIE*, vol. 1967, 1993.

[16] CEDIP Infrared Systems. *JADE-UC IR Uncooled Camera*. Spezifikation, 2003.

[17] Christophe Balestra, Pål-Kristian Engstad. *The technology of uncharted: Drake's fortune*. In *Game Developers Conference (GDC)*. 2008.

[18] Cook, Robert L., Porter, Thomas, und Carpenter, Loren. *Distributed ray tracing*. In *Proceedings of the 11th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '84, Seiten 137–145. ACM, New York, NY, USA, 1984. ISBN 0-89791-138-5. doi:10.1145/800031.808590.

[19] Cozzi, Patrick und Riccio, Christophe. *Opengl pipeline map*. In Patrick Cozzi und Christophe Riccio, Herausgeber, *OpenGL Insights*, Seiten First Page–Last Page. CRC Press, Juli 2012. ISBN 978-1439893760. `http://www.openglinsights.com/`.

[20] Crassin, Cyril, Neyret, Fabrice, Sainz, Miguel, Green, Simon, und Eisemann, Elmar. *Interactive indirect illumination using voxel cone tracing. Computer Graphics Forum*, 30(7):1921–1930, 2011. ISSN 1467-8659. doi:10.1111/j.1467-8659.2011.02063.x.

[21] Dachsbacher, Carsten, Křivánek, Jaroslav, Hašan, Miloš, Arbree, Adam, Walter, Bruce, und Novák, Jan. *Scalable realistic rendering with many-light methods. Computer Graphics Forum*, 33(1):88–104, 2014. ISSN 1467-8659. doi:10.1111/cgf.12256.

[22] Dachsbacher, Carsten und Stamminger, Marc. *Reflective shadow maps*. In *Proceedings of the 2005 Symposium on Interactive 3D Graphics and Games*, I3D '05, Seiten 203–231. ACM, New York, NY, USA, 2005. ISBN 1-59593-013-2. doi:10.1145/1053427.1053460.

[23] de Miguel, A., Bilbao, J., Aguiar, R., Kambezidis, H., und Negro, E. *Diffuse solar irradiation model evaluation in the north mediterranean belt area. Solar Energy*, 70(2):143–153, 2001. ISSN 0038-092X. doi:10.1016/S0038-092X(00)00135-3.

[24] Dimitriev, Kirill und Uralsky, Yury. *Soft shadows using hierarchical min-max shadowmap.* In *Game Developers Conference (GDC)*. 2007.

[25] Dong, Zhao, Grosch, Thorsten, Ritschel, Tobias, Kautz, Jan, und Seidel, Hans-Peter. *Real-time indirect illumination with clustered visibility.* In *Vision, Modeling, and Visualization Workshop*, Seiten 187–196. 2009.

[26] Dou, Hang, Yan, Yajie, Kerzner, Ethan, Dai, Zeng, und Wyman, Chris. *Adaptive depth bias for shadow maps.* In *Proceedings of the 18th Meeting of the ACM SIGGRAPH Symposium on Interactive 3D Graphics and Games*, I3D '14, Seiten 97–102. ACM, New York, NY, USA, 2014. ISBN 978-1-4503-2717-6. doi:10.1145/2556700.2556706.

[27] Dumont, Reynald, Robert, Thomas, Verdavaine, Yan, Lapierre, Fabian, Acheroy, Marc, und Marcel, Jean-Paul. *SAFIR, a Testbed for Maritime Simulation.* In *3rd International IR Target, Background Modelling & Simulation (ITBMS) Workshop*. Toulouse, France, 2007.

[28] Eisemann, Elmar, Schwarz, Michael, Assarsson, Ulf, und Wimmer, Michael. *Real-Time Shadows.* A K Peters / CRC Press, Dezember 2011.

[29] Engel, Wolfgang. *Cascaded shadow maps.* In Wolfgang Engel, Herausgeber, *Shader X5: Advanced Rendering Techniques*. Charles River Media, 2006. ISBN 1584504994.

[30] Fernando, Randima. *Percentage-Closer Soft Shadows. ACM SIGGRAPH 2005 Sketches*, 2005. doi:10.1145/1187112.1187153.

[31] Fernando, Randima, Fernandez, Sebastian, Bala, Kavita, und Greenberg, Donald P. *Adaptive shadow maps.* In *Proceedings of the 28th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '01, Seiten 387–390. ACM, New York, NY, USA, 2001. ISBN 1-58113-374-X. doi:10.1145/383259.383302.

[32] Golub, Gene H. und Pereyra, Victor. *The differentiation of pseudoinverses and nonlinear least squares problems whose variables separate.* Technischer Bericht, Stanford, CA, USA, 1972.

[33] Greene, Ned, Kass, Michael, und Miller, Gavin. *Hierarchical z-buffer visibility.* In *Proceedings of the 20th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '93, Seiten 231–238. ACM, New York, NY, USA, 1993. ISBN 0-89791-601-8. doi:10.1145/166117.166147.

[34] Grigull, Ulrich, Bach, Josef, und Sandner, Heinrich. *Näherungslösungen der nichtstationären Wärmeleitung. Forschung im Ingenieurwesen*, 32(1):11–18, 1966.

[35] Gumbau, Jesus, Chover, Miguel, und Sbert, Mateu. *Screen space soft shadows.* In Wolfgang Engel, Herausgeber, *GPU Pro - Advanced Rendering Techniques*, Seiten 477–490. A.K. Peters, 2010. ISBN 1568814720.

[36] Hanjun, Jin und Huali, Sun. *Rendering fake soft shadows based on the erosion and dilation.* In *Computer Engineering and Technology (ICCET), 2010 2^{nd} International Conference on*, Band 6, Seiten V6–234–V6–236. April 2010. doi:10.1109/ICCET.2010.5486285.

[37] Hanke-Bourgeois, Martin. *Grundlagen der Numerischen Mathematik und des Wissenschaftlichen Rechnens.* Vieweg+Teubner Verlag, 2009. ISBN 9783834807083.

[38] Hargreaves, Shawn und Harris, Mark. *6800 leagues under the sea - deferred shading*, 2004. URL `http://http.download.nvidia.com/developer/presentations/2004/6800_Leagues/6800_Leagues_Deferred_Shading.pdf`.

[39] Hašan, Miloš, Pellacini, Fabio, und Bala, Kavita. *Matrix row-column sampling for the many-light problem.* In *ACM SIGGRAPH 2007 Papers*, SIGGRAPH '07. ACM, New York, NY, USA, 2007. doi:10.1145/1275808.1276410.

[40] Hašan, Miloš, Velázquez-Armendariz, Edgar, Pellacini, Fabio, und Bala, Kavita. *Tensor clustering for rendering many-light animations.* In *Proceedings of the Nineteenth Eurographics Conference on Rendering*, EGSR'08, Seiten 1105–1114. Eurographics Association, Aire-la-Ville, Switzerland, Switzerland, 2008. doi:10.1111/j.1467-8659.2008.01248.x.

[41] Higham, Nicholas J. *Accuracy and Stability of Numerical Algorithms.* Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, second Auflage, 2002. ISBN 0-89871-521-0.

[42] Hung-Chien, Liao. *Shadow mapping for omnidirectional light using tetrahedrom mapping.* In Wolfgang Engel, Herausgeber, *GPU Pro - Advanced Rendering Techniques*, Seiten 455–475. A.K. Peters, 2010. ISBN 1568814720.

[43] IDST-Render. *Restaurant model*, 2013. URL `http://idst-render.com/scenes.html`. Zugegriffen am 27.05.2013.

[44] Isidoro, John R. *Shadow-mapping: GPU-based tips and techniques.* In *Game Developers Conference (GDC)*. 2006.

[45] Joly, A. *Virtual rays for EO, RF & AD operational simulation.* In *5th International IR Target, Background Modelling & Simulation (ITBMS) Workshop*. Toulouse, France, 2009.

[46] Kajiya, James T. *The rendering equation.* In *Proceedings of the 13^{th} Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '86, Seiten 143–150. ACM, New York, NY, USA, 1986. ISBN 0-89791-196-2. doi:10.1145/15922.15902.

[47] Keller, Alexander. *Instant radiosity.* In *Proceedings of the 24^{th} Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '97, Seiten

49–56. ACM Press/Addison-Wesley Publishing Co., New York, NY, USA, 1997. ISBN 0-89791-896-7. doi:10.1145/258734.258769.

[48] Klein, Andreas, Nischwitz, Alfred, und Obermeier, Paul. *Contact hardening soft shadows using erosion.* In *20th WSCG International Conference on Computer Graphics, Visualization and Computer Vision. Communication proceedings*, Seiten 53–58. Union Agency, Pilsen, Czech Republic, 2012. ISSN 1213-6972.

[49] Klein, Andreas, Nischwitz, Alfred, Schätz, Peter, und Obermeier, Paul. *Incorporation of thermal shadows into real-time infrared three-dimensional image generation. Optical Engineering*, 53(5):053113, 2014. doi:10.1117/1.OE.53.5.053113.

[50] Laine, Samuli, Saransaari, Hannu, Kontkanen, Janne, Lehtinen, Jaakko, und Aila, Timo. *Incremental instant radiosity for real-time indirect illumination.* In *Proceedings of the 18th Eurographics Conference on Rendering Techniques*, EGSR'07, Seiten 277–286. Eurographics Association, Aire-la-Ville, Switzerland, Switzerland, 2007. ISBN 978-3-905673-52-4. doi:10.2312/EGWR/EGSR07/277-286.

[51] Lapierre, Fabian D. und Acheroy, Marc. *Performance enhancement and validation of the open-source software for modeling of ship infrared signatures (OSMOSIS). Journal of Computational and Applied Mathematics*, 234(7):2342–2349, 2010. ISSN 0377-0427. doi:10.1016/j.cam.2009.08.091. Fourth International Conference on Advanced COmputational Methods in ENgineering (ACOMEN 2008).

[52] Lauritzen, Andrew. *Summed-area variance shadow maps.* In Hubert Nguyen, Herausgeber, *GPU Gems 3*, Seiten 157–182. Addison-Wesley, 2007. ISBN 9780321545428.

[53] Lauritzen, Andrew. *Deferred rendering for current and future rendering pipelines.* In *Beyond Programmable Shading Course, SIGGRAPH 2010*. 2010.

[54] Lauritzen, Andrew und McCool, Michael. *Layered variance shadow maps.* In *Proceedings of Graphics Interface 2008*, GI '08, Seiten 139–146. Canadian Information Processing Society, Toronto, Ont., Canada, Canada, 2008. ISBN 978-1-56881-423-0.

[55] Loos, Bradford James und Sloan, Peter-Pike. *Volumetric obscurance.* In *I3D '10: Proceedings of the 2010 ACM SIGGRAPH symposium on Interactive 3D Graphics and Games*, Seiten 151–156. ACM, New York, NY, USA, 2010. ISBN 978-1-60558-939-8. doi:10.1145/1730804.1730829.

[56] Maréchal, N., Guérin, E., Galin, E., Mérillou, S., und Mérillou, N. *Heat transfer simulation for modeling realistic winter sceneries. Computer Graphics Forum*, 29(2):449–458, 2010. ISSN 1467-8659. doi:10.1111/j.1467-8659.2009.01614.x.

[57] Marek, Rudi und Nitsche, Klaus. *Praxis der Wärmeübertragung: Grundlagen - Anwendungen - Übungsaufgaben.* Carl Hanser Verlag GmbH & Company KG, 2012. ISBN 9783446433205.

[58] McGuire, Morgan. *Computer graphics archive*, 2013. URL `http://graphics.cs.williams.edu/data`. Zugegriffen am 27.05.2013.

[59] Mittring, Martin. *Finding next gen: Cryengine 2.* In *ACM SIGGRAPH 2007 Courses*, SIGGRAPH '07, Seiten 97–121. ACM, New York, NY, USA, 2007. ISBN 978-1-4503-1823-5. doi:10.1145/1281500.1281671.

[60] Mittring, Martin. *A bit more deferred - cryengine 3.* In *Game Developers Conference (GDC)*. 2009.

[61] MohammadBagher, Mahdi, Kautz, Jan, Holzschuch, Nicolas, und Soler, Cyril. *Screen-space percentage-closer soft shadows.* In *ACM SIGGRAPH 2010 Posters*, SIGGRAPH '10, Seiten 133:1–133:1. ACM, New York, NY, USA, 2010. ISBN 978-1-4503-0393-4. doi:10.1145/1836845.1836987.

[62] Numpy. *Numerical Python*, 2014. URL `http://www.numpy.org`. Zugegriffen am 04.06.2014.

[63] NVIDIA. *NVIDIA GF100 Whitepaper.* http://www.nvidia.com/attach/3632670.html, 2010.

[64] NVIDIA. *NVIDIA GeForce GTX 680 Whitepaper*, 2012. URL `http://www.geforce.com/Active/en_US/en_US/pdf/GeForce-GTX-680-Whitepaper-FINAL.pdf`.

[65] O'Leary, Dianne P. und Rust, Bert W. *Variable projection for nonlinear least squares problems. Computational Optimization and Applications*, 54(3):579–593, 2013. ISSN 0926-6003. doi:10.1007/s10589-012-9492-9.

[66] Olsson, Ola und Assarsson, Ulf. *Tiled shading. Journal of Graphics, GPU, and Game Tools*, 15(4):235–251, 2011. doi:10.1080/2151237X.2011.621761.

[67] Olsson, Ola, Sintorn, Erik, Kämpe, Viktor, Billeter, Markus, und Assarsson, Ulf. *Efficient virtual shadow maps for many lights.* In *Proceedings of the ACM SIGGRAPH Symposium on Interactive 3D Graphics and Games*. ACM, 2014.

[68] Owens, J.D., Houston, M., Luebke, D., Green, S., Stone, J.E., und Phillips, J.C. *GPU computing. Proceedings of the IEEE*, 96(5):879–899, Mai 2008. ISSN 0018-9219. doi:10.1109/JPROC.2008.917757.

[69] Ownby, John-Paul, Hall, Robert, und Hall, Christopher. *Rendering techniques in toy story 3.* In *Advances in Real-Time Rendering in Games*, SIGGRAPH 2010 courses. ACM, New York, NY, USA, 2010.

[70] Paul Martz. *OpenSceneGraph Quick Start Guide*, 2011. URL `http://www.skew-matrix.com/OSGQSG/`.

[71] Peter Schätz. *Auswertung der Schattenmessung am 18.08.2011.* Studie der MBDA Deutschland GmbH, 2011.

[72] Peter Schätz. *Berechnung thermischer Schatten.* Studie der MBDA Deutschland GmbH, 2011.

[73] Peter Schätz. *Ambient Occlusion*. Studie der MBDA Deutschland GmbH, 2013.

[74] Pharr, Matt und Humphreys, Greg. *Physically based rendering: From theory to implementation*. Morgan Kaufmann, 2010. ISBN 0123750792.

[75] Piessens, Robert, de Doncker-Kapenga, Elise, Überhuber, Christoph W., und Kahane, David K. *Quadpack - A Subroutine Package for Automatic Integration*. Springer Berlin Heidelberg, 1983. ISBN 9783540125532.

[76] Poglio, Thierry, Savaria, Eric, und Wald, Lucien. *Outdoor Scene Synthesis in the Infrared Range for Remote Sensing Applications*. In *CISST '02*. 2002.

[77] Polifke, Wolfgang und Kopitz, Jan. *Wärmeübertragung*. Pearson Studium, München, Deutschland, 2009. ISBN 3827373492.

[78] Reda, Ibrahim und Andreas, Afshin. *Solar position algorithm for solar radiation applications*. *Solar Energy*, 76(5):577–589, 2004. ISSN 0038-092X. doi:10.1016/j.solener.2003.12.003.

[79] Reeves, William T., Salesin, David H., und Cook, Robert L. *Rendering antialiased shadows with depth maps*. In *SIGGRAPH '87*, Seiten 283–291. ACM, New York, USA, 1987. doi:10.1145/37401.37435.

[80] Ritschel, T., Grosch, T., Kim, M. H., Seidel, H.-P., Dachsbacher, C., und Kautz, J. *Imperfect shadow maps for efficient computation of indirect illumination*. *ACM Trans. Graph.*, 27(5):129:1–129:8, Dezember 2008. ISSN 0730-0301. doi:10.1145/1409060.1409082.

[81] Ritschel, Tobias, Dachsbacher, Carsten, Grosch, Thorsten, und Kautz, Jan. *The state of the art in interactive global illumination*. *Computer Graphics Forum*, 31(1):160–188, 2012. ISSN 1467-8659. doi:10.1111/j.1467-8659.2012.02093.x.

[82] Ritschel, Tobias, Eisemann, Elmar, Ha, Inwoo, Kim, James Dokyoon, und Seidel, Hans-Peter. *Making imperfect shadow maps view-adaptive: High-quality global illumination in large dynamic scenes*. *Computer Graphics Forum*, 30(8):2258–2269, 2011. doi:10.1111/j.1467-8659.2011.01998.x.

[83] Ritschel, Tobias, Grosch, Thorsten, und Seidel, Hans-Peter. *Approximating Dynamic Global Illumination in Screen Space*. In *Proceedings ACM SIGGRAPH Symposium on Interactive 3D Graphics and Games*. 2009.

[84] Robison, Austin und Shirley, Peter. *Image space gathering*. In *Proceedings of the Conference on High Performance Graphics 2009*, HPG '09, Seiten 91–98. ACM, New York, NY, USA, 2009. ISBN 978-1-60558-603-8. doi:10.1145/1572769.1572784.

[85] Rong, Guodong und Tan, Tiow-Seng. *Utilizing jump flooding in image-based soft shadows*. In *ACM Syposium on Virtual Reality Software and Technology*, Seiten 173–180. 2006.

[86] Salvi, Marco. *Rendering filtered shadows with exponential shadow maps*. In Wolfgang Engel, Herausgeber, *Shader X6: Advanced Rendering Techniques*. Charles River Media, 2008. ISBN 1584505443.

[87] Schott, John R., Raqueno, Rolando V., und Salvaggio, Carl. *Incorporation of a time-dependent thermodynamic model and a radiation propagation model into IR 3D synthetic image generation. Optical Engineering*, 31(7):1505–1516, 1992. doi:10.1117/12.57682.

[88] Scipy. *Scientific Computing Tools for Python*, 2014. URL `http://www.scipy.org`. Zugegriffen am 04.06.2014.

[89] Shen, Li, Feng, Jieqing, und Yang, Baoguang. *Exponential soft shadow mapping. Computer Graphics Forum*, 32(4):107–116, 2013. ISSN 1467-8659. doi:10.1111/cgf.12156. URL `http://dx.doi.org/10.1111/cgf.12156`.

[90] Shreiner, Dave, Sellers, Graham, Kessenich, John M., und Licea-Kane, Bill M. *OpenGL Programming Guide: The Official Guide to Learning OpenGL, Version 4.3.* Addison-Wesley Professional, 8th Auflage, 2013. ISBN 0321773039, 9780321773036.

[91] Swoboda, Mark. *Deferred lighting and post processing on playstation 3*. In *Game Developers Conference (GDC)*. 2009.

[92] Taylor, John R. *An Introduction to Error Analysis: The Study of Uncertainties in Physical Measurements.* University Science Books, 1997. ISBN 9780935702750.

[93] Thermo Analytics Inc. *RadTherm*, 2013. URL `http://www.thermoanalytics.com`. Zugegriffen am 09.12.2011.

[94] Thibieroz, Nicolas. *Deferred shading optimizations.* In *Game Developers Conference (GDC)*. 2011.

[95] Vollmer, Michael. *Newton's law of cooling revisited. European Journal of Physics*, 30(5):1063, 2009.

[96] Vollmer, Michael und Möllmann, Klaus-Peter. *Infrared Thermal Imaging: Fundamentals, Research and Applications.* John Wiley & Sons, 2010. ISBN 3527407170.

[97] Walter, Bruce, Fernandez, Sebastian, Arbree, Adam, Bala, Kavita, Donikian, Michael, und Greenberg, Donald P. *Lightcuts: A scalable approach to illumination. ACM Trans. Graph.*, 24(3):1098–1107, Juli 2005. ISSN 0730-0301. doi:10.1145/1073204.1073318.

[98] Whitted, Turner. *An improved illumination model for shaded display. Commun. ACM*, 23(6):343–349, Juni 1980. ISSN 0001-0782. doi:10.1145/358876.358882.

[99] Wiesenhütter, Daniel, Klein, Andreas, und Nischwitz, Alfred. *Lightcluster - clustering lights to accelerate shadow computation. Journal of WSCG*, 21(1):31–40, 2013. ISSN 1213-6972.

[100] Williams, Lance. *Casting curved shadows on curved surfaces.* In *SIGGRAPH '78*, Seiten 270–274. ACM, New York, NY, USA, 1978. doi:10.1145/800248.807402.

[101] Wimmer, Michael, Scherzer, Daniel, und Purgathofer, Werner. *Light space perspective shadow maps.* In *Proceedings of the Fifteenth Eurographics Conference*

on Rendering Techniques, EGSR'04, Seiten 143–151. Eurographics Association, Aire-la-Ville, Switzerland, Switzerland, 2004. ISBN 3-905673-12-6. doi: 10.2312/EGWR/EGSR04/143-151.

[102] Wolfowitz, Jacob. *The minimum distance method. The Annals of Mathematical Statistics*, Seiten 75–88, 1957.

A. Anhang

A.1. Fehlerabschätzung

A.1.1. Festlegung der Parameter über Eigenwerte für ein exponentielles Temperaturprofil

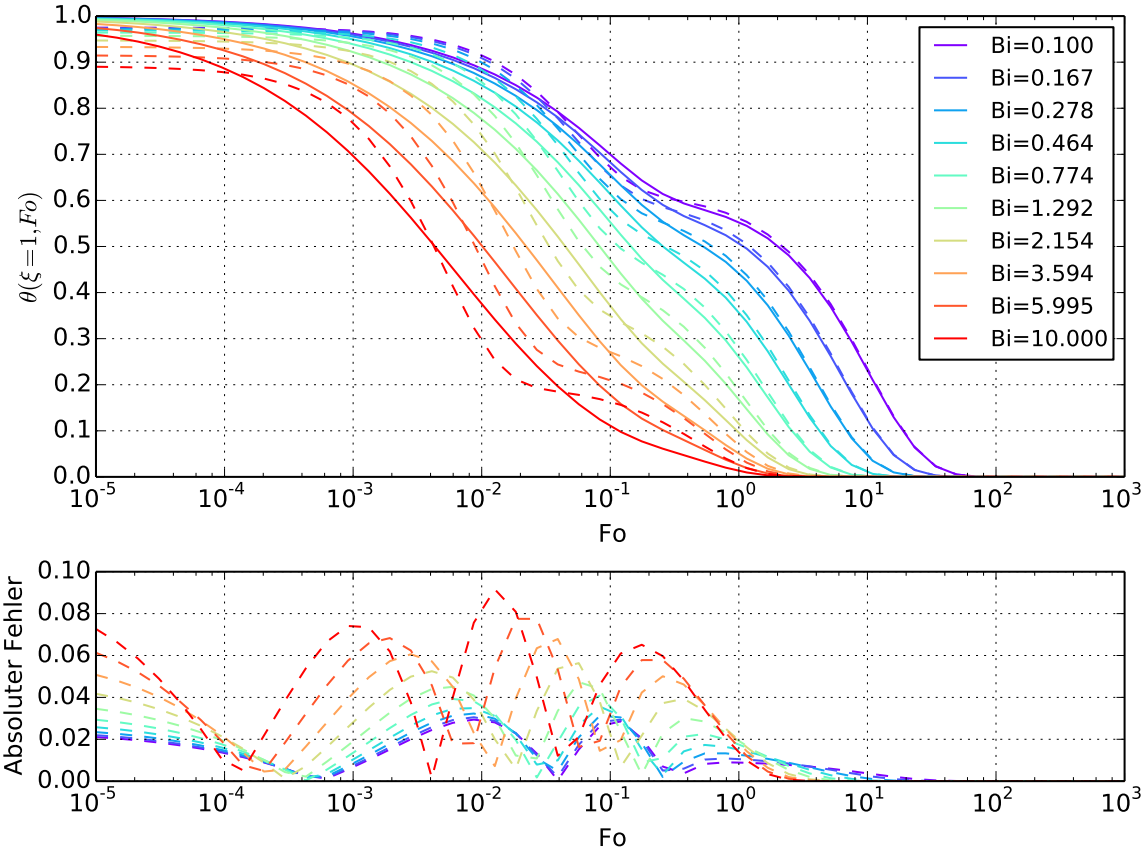


Abbildung A.1.: Vergleich der Fourier-Reihe (durchgezogene Linie) und der Approximation mit zwei Exponentialfunktionen (gestrichelte Linie) für ein exponentielles Temperaturprofil. Die Zeitkonstante τ_1 wurde mit Hilfe des ersten Eigenwert und der Parameter β_1 über den Koeffizienten C_1 bestimmt.

Abbildung A.1 zeigt die Fehlerabschätzung für ein exponentielles Temperaturprofil. Der Einfluss des exponentiellen Temperaturprofils tritt gegenüber dem Linearen zeitlich verzögert auf. Ansonsten verhält es sich wie das lineare Profil.

A.1.2. Kurvenanpassung der Zeitkonstante für ein lineares Temperaturprofil

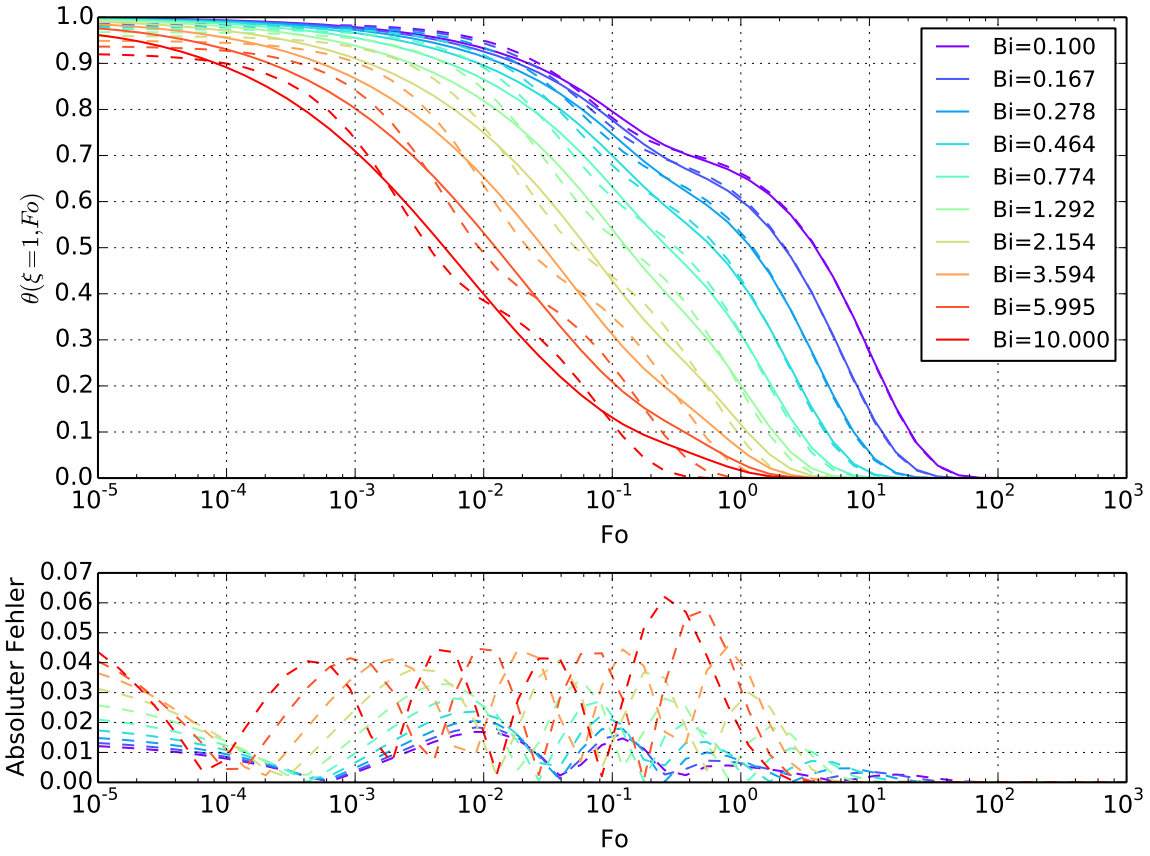


Abbildung A.2.: Vergleich der Fourier-Reihe (durchgezogene Linie) und der Approximation mit zwei Exponentialfunktionen (gestrichelte Linie) für ein lineares Temperaturprofil. Alle Parameter wurden durch eine Kurvenanpassung bestimmt.

A.1.3. Kurvenanpassung der Zeitkonstante für ein exponentielles Temperaturprofil

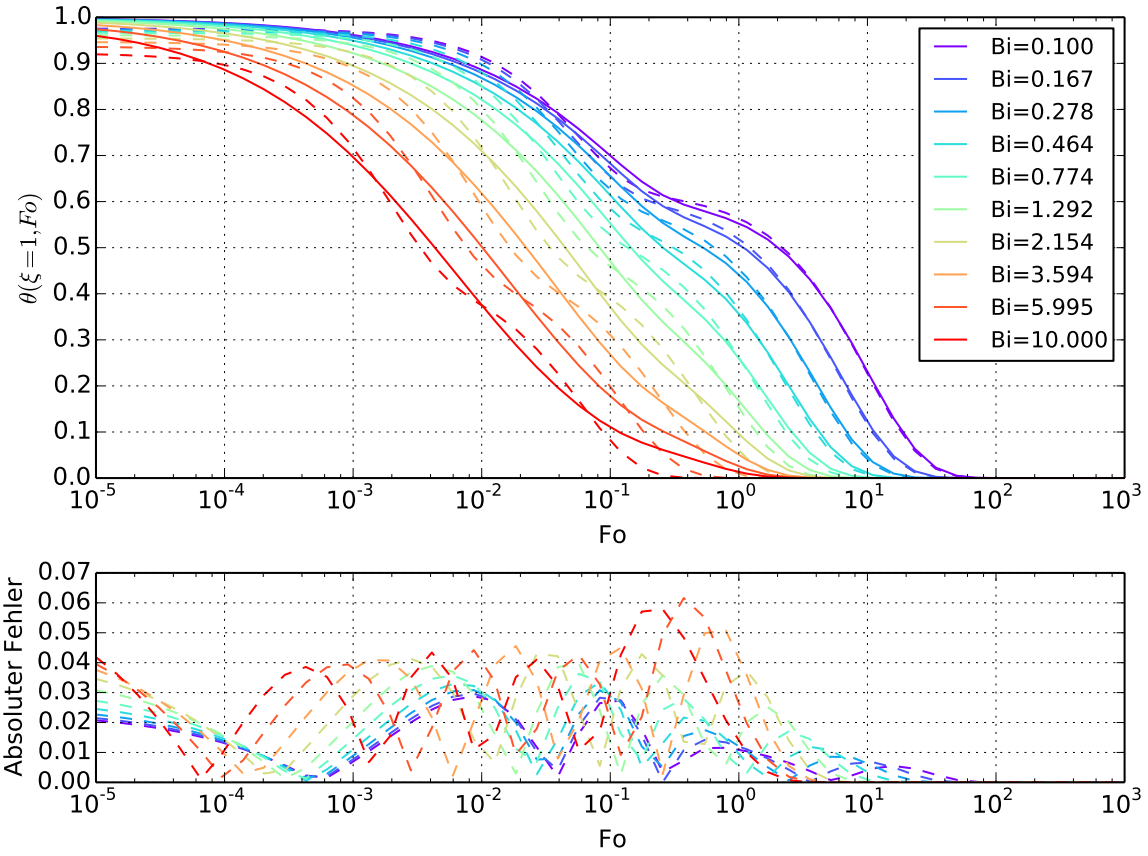


Abbildung A.3.: Vergleich der Fourier-Reihe (durchgezogene Linie) und der Approximation mit zwei Exponentialfunktionen (gestrichelte Linie) für ein exponentielles Temperaturprofil. Alle Parameter wurden durch eine Kurvenanpassung bestimmt.

A.1.4. Halbunendlicher Körper für ein exponentielles Temperaturprofil

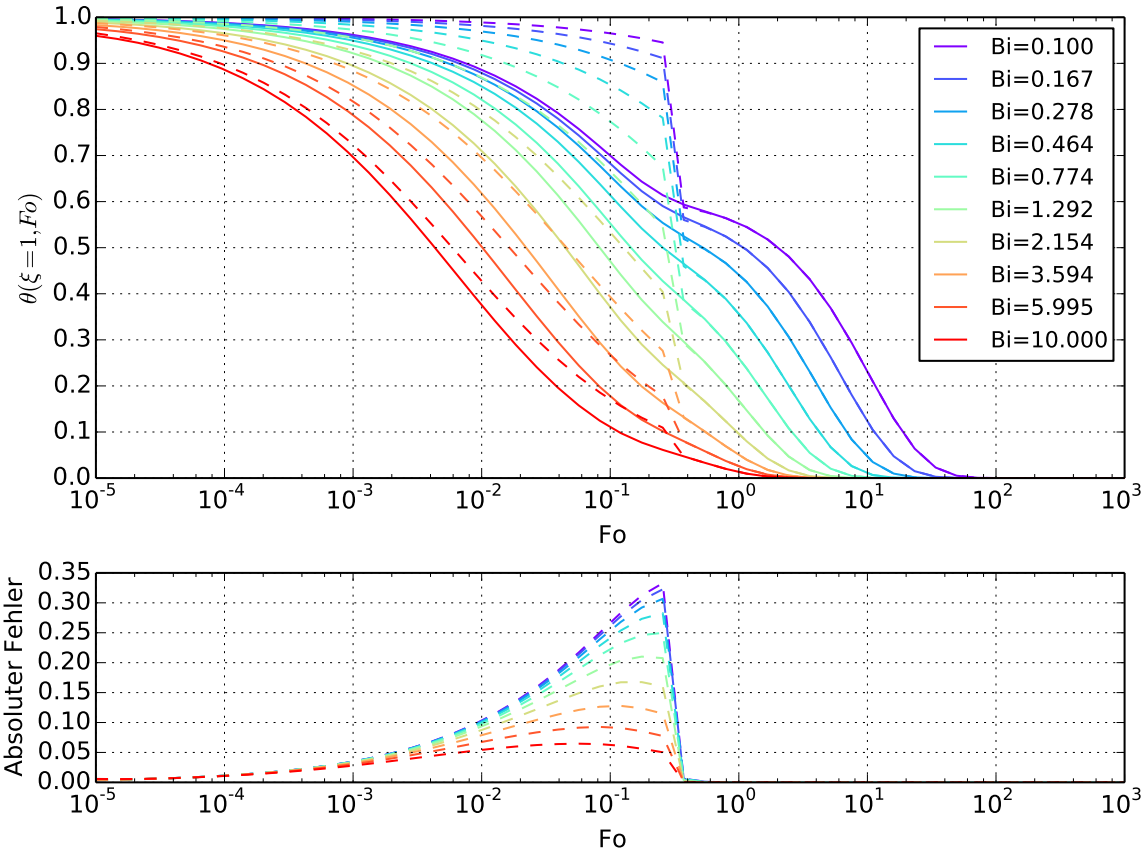


Abbildung A.4.: Vergleich der Fourier-Reihe (durchgezogene Linie) und der Approximation mit halbunendlichem Körper sowie ersten Reihenglied der Fourier-Reihe (gestrichelte Linie) für ein exponentielles Temperaturprofil. Der Wechsel auf die Langzeitlösung erfolgt bei $Fo = 0.3$.

A.2. Numpy-Implementierung der Variablenprojektion

```
def DecayWithExpExp(Bi,Fo,theta, a):
    A = zeros((Fo.shape[0],2))
    A[:,0] = exp(-a[0] * Fo)
    A[:,1] = exp(-a[1] * Fo)
    [c,res,rank,s] = linalg.lstsq(A,theta)
    return dot(A,c)
```

```
def FitExpExp(Bi, Fo, theta):
    f = lambda a, Fo, y  : y-DecayWithExpExp(Bi,Fo,theta, a)
    guess=array([Bi*2, Bi]);
    popt,pcov = leastsq(func=f,x0=guess, args=(Fo, theta))
    return DecayWithExpExp(Bi,Fo,theta,popt)
```

A.3. Numpy-Implementierung der Eigenwertbestimmung

```
def Eigenvalue(Bi, k):
    f = lambda x : tan(x) - Bi / x
    dk = fsolve(f, (k-1)*pi+spacing(1) + pi/4.0)
    return dk[0]
```

A.4. Temperaturvergleich von thermischen Schatten für weitere Materialien

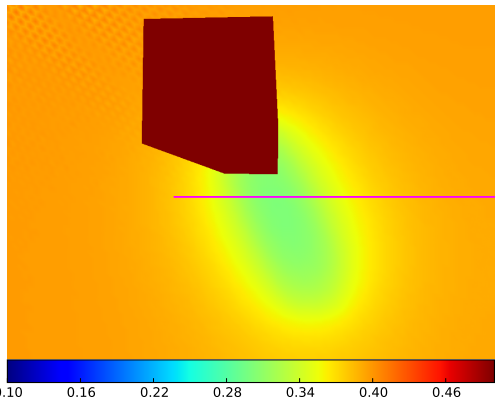
In diesem Abschnitt werden die absoluten Temperaturen unserer Approximation für weitere Materialien mit einer RadTherm Lösung verglichen. Dabei wird nur der Schatten in Betracht gezogen. Die Angaben zur Wärmeleitfähigkeit wurden aus der Materialdatenbank von RadTherm entnommen. Dabei wurden die gleichen Einstellungen wie in Kapitel 4.4.1.3 benutzt.

Folgende Materialien wurden untersucht: Aluminium, kurzes Gras und Eichenholz.

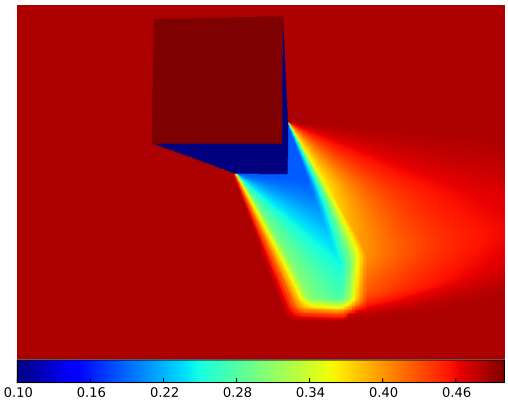
Aluminium ist ein Leiter mit einer hohen Wärmeleitfähigkeit von $\lambda = 201$. Dadurch überwiegen bei der Darstellung von Schatten die lateralen Wärmeströme. Diese werden bei unserer Approximation vernachlässigt. Des Weiteren ist die Schattentemperatur im Vergleich zur Strahlungstemperatur sehr gering. Dadurch ist die Ausgleichstemperatur durch eine Abschattung zu kalt.

Kurzes Gras zeichnet sich vor allem durch eine starke nicht lineare Abhängigkeit der Ausgleichstemperatur vom Schattenfaktor aus (Siehe Abbildung 4.14). Kurzes Gras kann dennoch mit unserer Lösung mit einer maximalen Abweichung von 2.0 K approximiert werden.

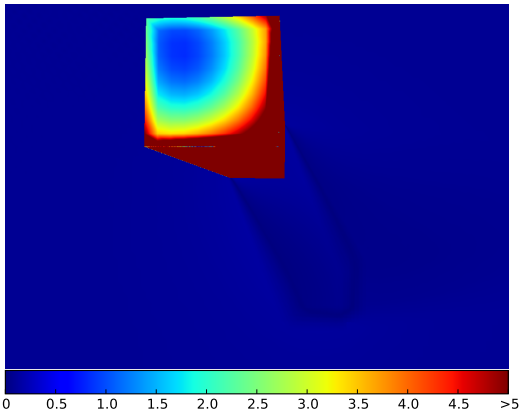
Eiche ist ein Isolator mit einer sehr geringen Wärmeleitfähigkeit von $\lambda = 0.21$. Dadurch entsteht ein geringer Wärmeaustausch zwischen den Oberflächenschichten. Es ist vor allem die Kurzzeitlösung relevant. Unsere Approximation kann den Halbschattenbereich nicht genau nachbilden. Daher entstehen in diesem Bereich Abweichungen von mehr als 5 K. Allerdings kann der Kernschatten exakt nachgebildet werden.



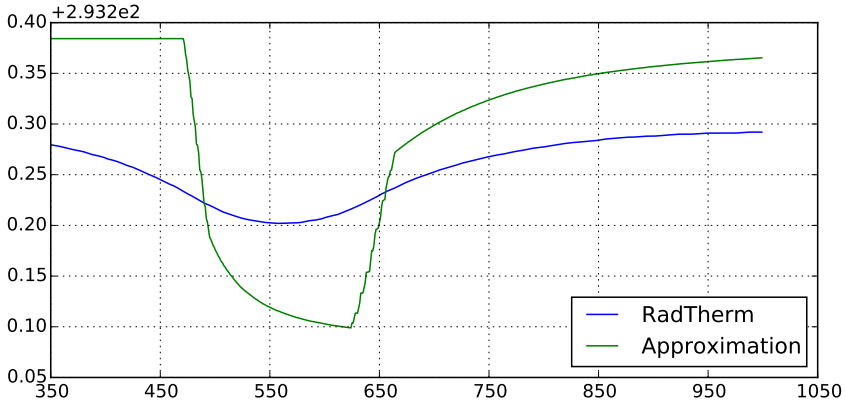
(a) RadTherm Lösung. Die Temperaturen wurden zwischen 293.2 K und 293.6 K skaliert.



(b) Unsere Lösung kann dieses Verhalten nicht nachstellen. Temperaturskalierung zwischen 293.2 K und 293.6 K.

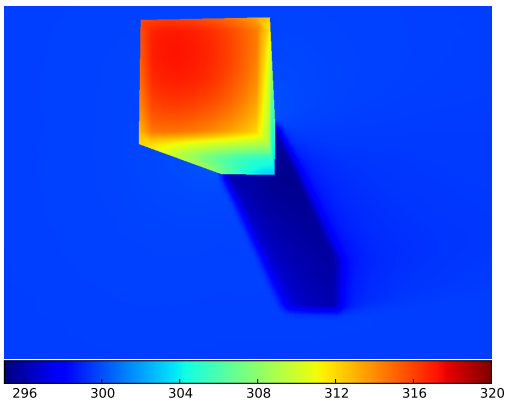


(c) Differenzbild zwischen (a) und (b).

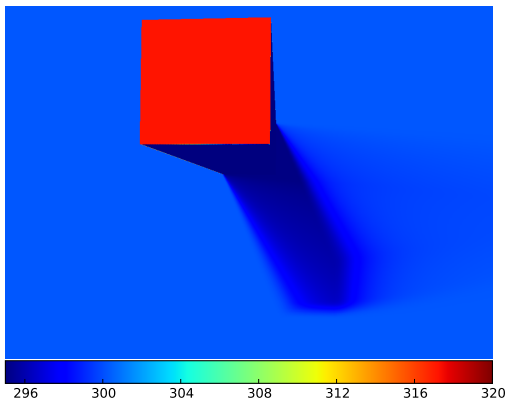


(d) Temperaturkurve der magentafarbenen Bildzeile.

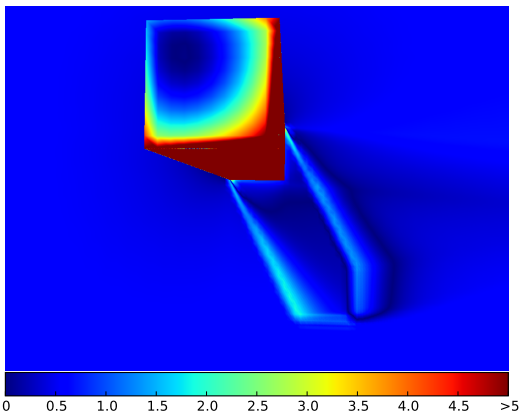
Abbildung A.5.: Vergleich der absoluten Temperaturen bei Aluminium. Temperaturen in K.



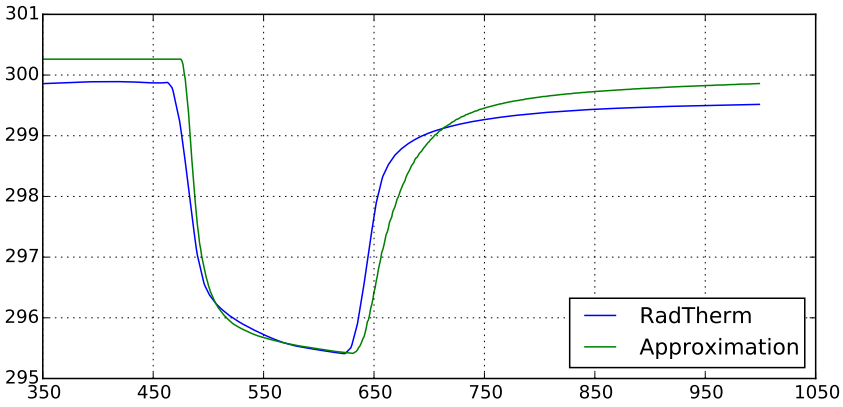
(a) RadTherm Lösung.



(b) Unsere Lösung.

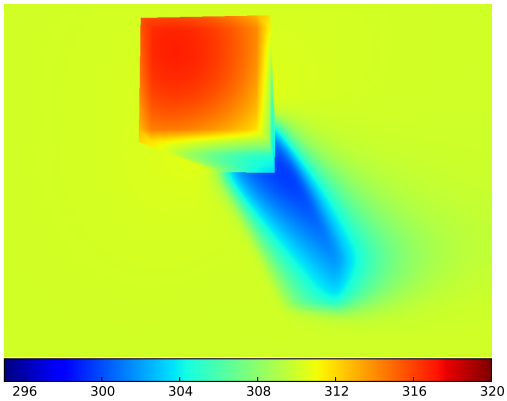


(c) Differenzbild zwischen (a) und (b).

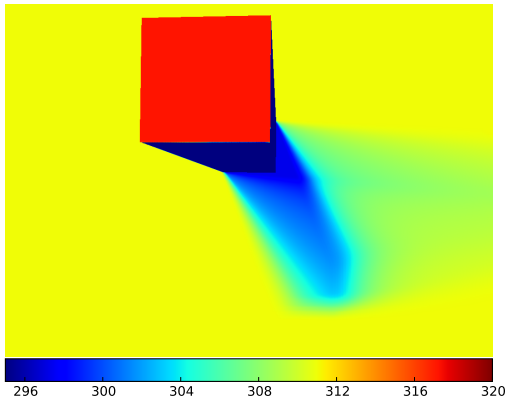


(d) Temperaturkurve der magentafarbenen Bildzeile.

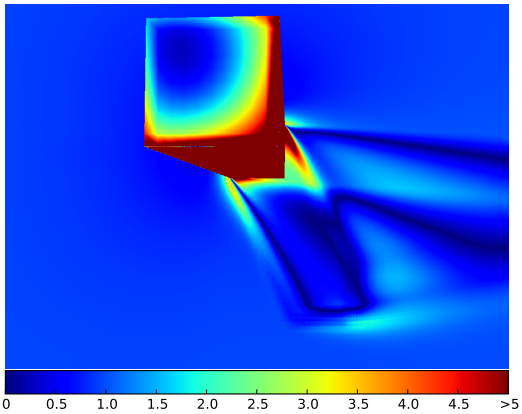
Abbildung A.6.: Vergleich der absoluten Temperaturen bei kurzem Gras. Temperaturen in K.



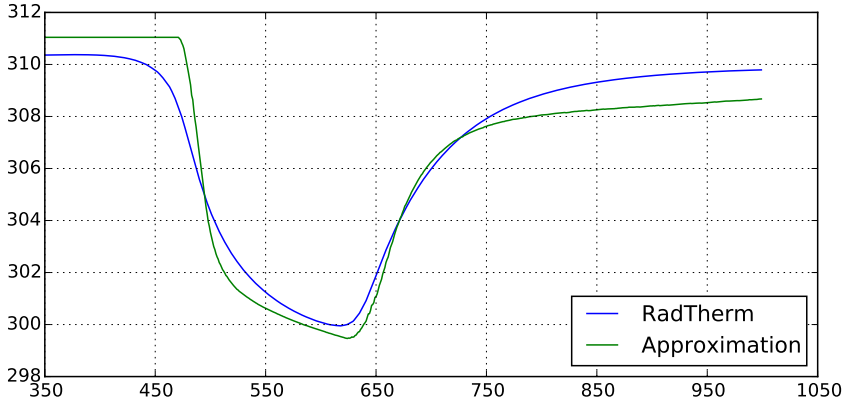
(a) RadTherm Lösung.



(b) Unsere Lösung.



(c) Differenzbild zwischen (a) und (b).



(d) Temperaturkurve der magentafarbenen Bildzeile.

Abbildung A.7.: Vergleich der absoluten Temperaturen bei Eichenholz. Temperaturen in K.

A.5. Temperaturvergleich der atmosphärischen Verdeckung für weitere Materialien

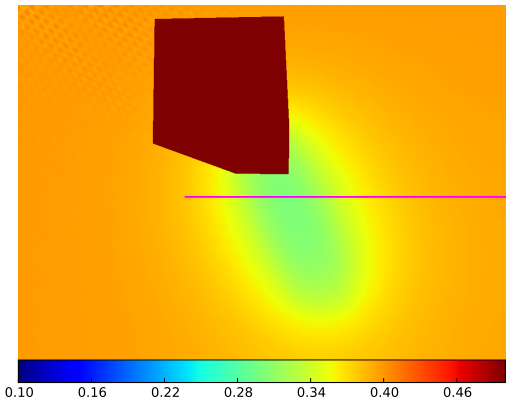
In diesem Abschnitt werden die absoluten Temperaturen unserer Approximation für weitere Materialien mit einer RadTherm Lösung verglichen. Dabei wird nur der Schatten in Betracht gezogen. Die Angaben zur Wärmeleitfähigkeit wurden aus der Materialdatenbank von RadTherm entnommen. Dabei wurden die gleichen Einstellungen wie in Kapitel 4.4.1.3 benutzt. Die Dicke der Oberfläche war jeweils 15 cm.

Folgende Materialien wurden untersucht: Aluminium, kurzes Gras und Eichenholz.

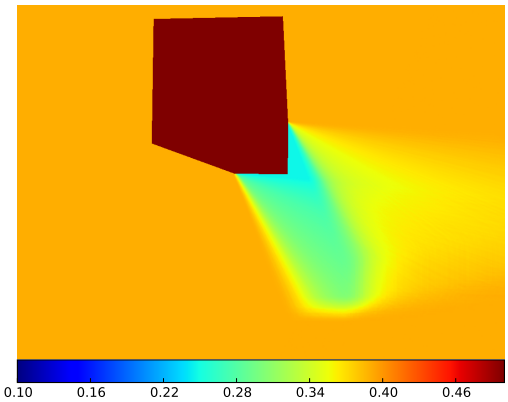
Aluminium ist ein Leiter mit einer hohen Wärmeleitfähigkeit von $\lambda = 201$, sowie einem Albedo von $\alpha = 0.49$ und einem Emissionsgrad von $\varepsilon = 0.22$. Durch die hohe Wärmeleitfähigkeit überwiegen die lateralen Wärmeströme. Diese werden bei unserer Approximation zwar vernachlässigt, allerdings kann die Temperaturdifferenz durch eine Wahl der Kerntemperatur T_c auf die Strahlungstemperatur von 292 K minimiert werden. Die Form des Schattens ist jedoch trotzdem nicht realisierbar.

Kurzes Gras wurde durch eine Wärmeleitfähigkeit von $\lambda = 1.9$, ein Albedo von 0.76 und einen Emissionsgrad von $\varepsilon = 0.95$ modelliert. Die Schattenhistorie ist bei unserer Lösung um etwa 0.5 K zu warm. Die größte Temperaturabweichung ist im Bereich des Halbschattens. Allerdings kann der Kernschatten mit minimaler Abweichung dargestellt werden.

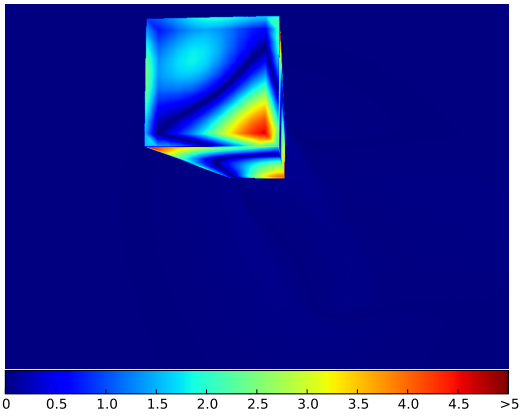
Eiche wurde mit einer Wärmeleitfähigkeit von $\lambda = 0.21$, ein Albedo von $\alpha = 0.78$ und einen Emissionsgrad von $\varepsilon = 0.9$ modelliert. Da die Halbschatten sehr weich werden, kann unsere Approximation die Schatten nur bedingt darstellen. Die Temperaturdifferenzen sind allerdings, bis auf in diesen Bereichen, gering.



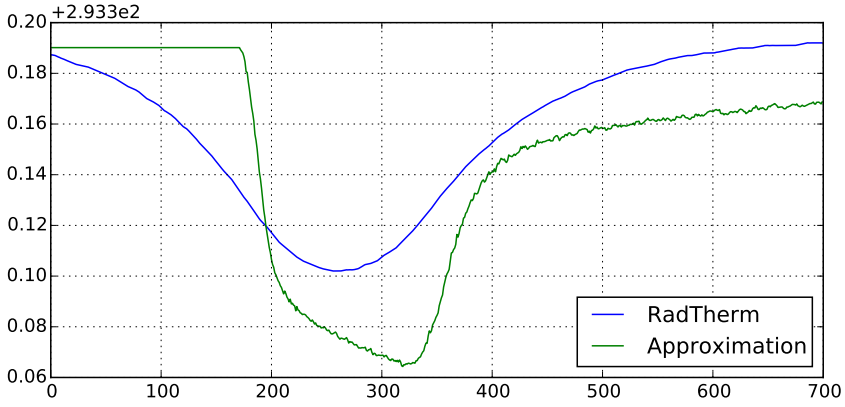
(a) RadTherm Lösung. Die Temperaturen wurden zwischen 293.2 K und 293.6 K skaliert.



(b) Unsere Lösung kann die Form des Schattens nicht reproduzieren. Temperaturskalierung zwischen 293.2 K und 293.6 K.

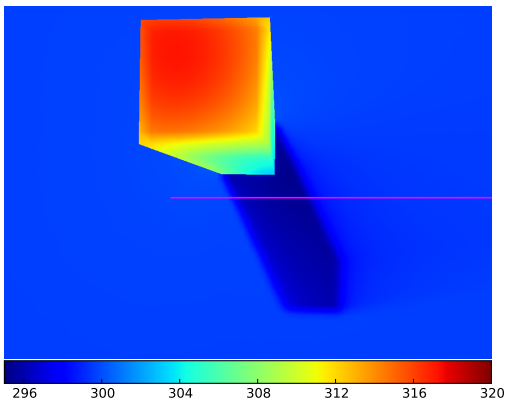


(c) Differenzbild zwischen (a) und (b).

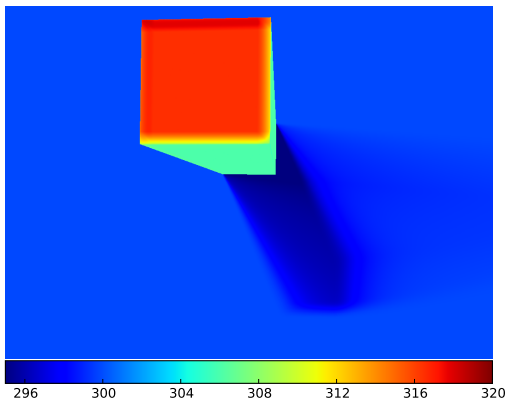


(d) Temperaturkurve der magentafarbenen Bildzeile.

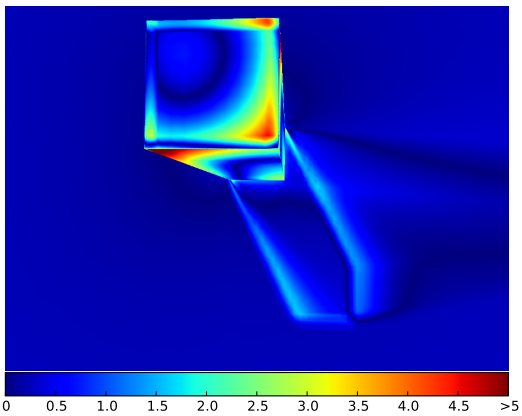
Abbildung A.8.: Vergleich der absoluten Temperaturen bei Aluminium. Temperaturen in K.



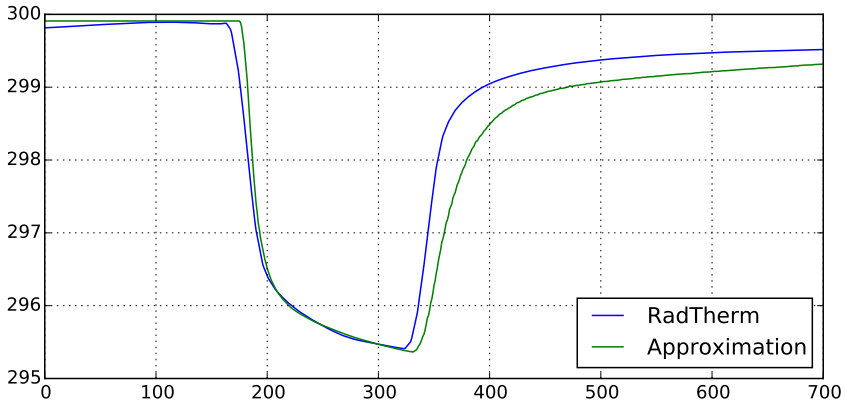
(a) RadTherm Lösung.



(b) Unsere Lösung.

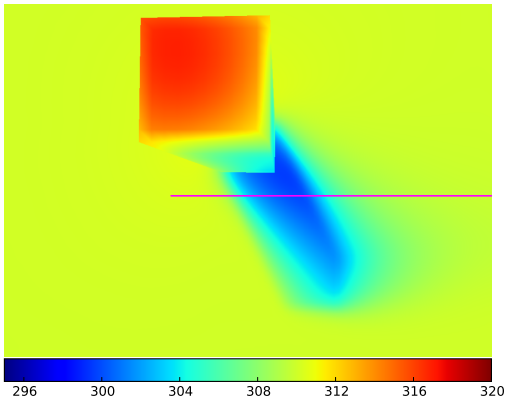


(c) Differenzbild zwischen (a) und (b).

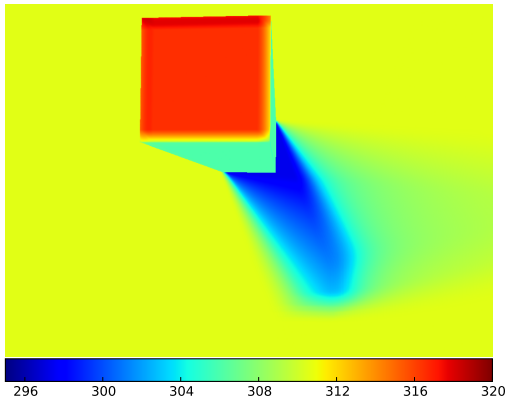


(d) Temperaturkurve der magentafarbenen Bildzeile.

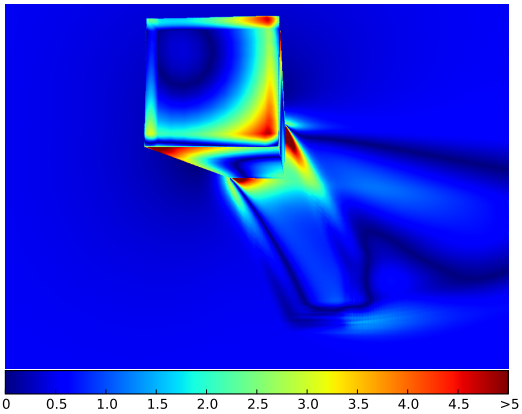
Abbildung A.9.: Vergleich der absoluten Temperaturen bei kurzem Gras. Temperaturen in K.



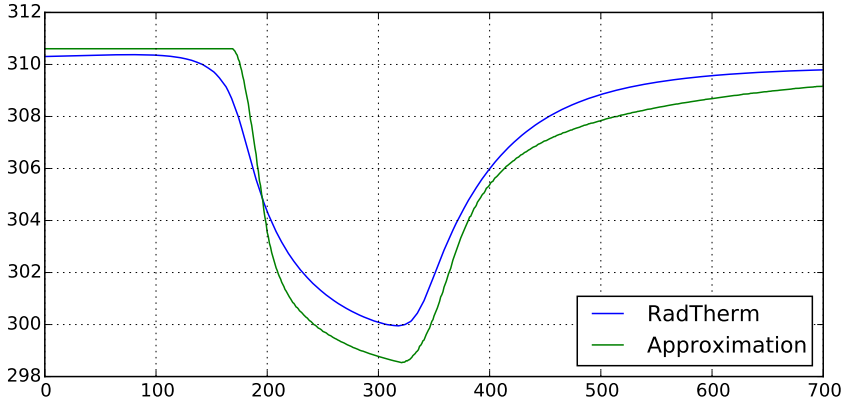
(a) RadTherm Lösung



(b) Unsere Lösung



(c) Differenzbild zwischen (a) und (b).



(d) Temperaturkurve der magentafarbenen Bildzeile.

Abbildung A.10.: Vergleich der absoluten Temperaturen bei Eichenholz. Temperaturen in K.