|  |
| --- |
| Rendering Denoising  Path Tracing Denoising mit neuronalen Netzen  **Bachelorthesis**  [Kurztext einfüngen, falls gewünscht] |
| Studiengang: CPCVR  Autor: Pascal Cornu  Betreuer: Marcus Hudritsch  Experten: Harald Studer  Datum: 02.06.2025 |

Management Summary

Diese Arbeit untersucht, wie die Bildqualität beim physikalisch korrekten Rendern (Path Tracing) effizient verbessert werden kann. Path Tracing ist ein verbreitetes Verfahren zur realistischen Bildsynthese in der Computergrafik. Es simuliert den Weg von Lichtstrahlen in einer virtuellen Szene sehr genau, ist jedoch mit hohem Rechenaufwand verbunden. Besonders bei Szenen mit indirekter oder komplexer Beleuchtung entsteht deutlich sichtbares Rauschen, wenn nicht genügend Rechenzeit aufgewendet wird. Dieses Rauschen äussert sich in Form von körnigen oder fleckigen Bildern.

Ziel der Arbeit war es, ein Verfahren zu entwickeln, das solche verrauschten Bilder automatisch verbessern kann – und zwar mithilfe Künstlicher Intelligenz. Dafür wurde ein eigener Trainingsdatensatz erstellt, bestehend aus verrauschten und hochwertigen Vergleichsbildern. Anschliessend wurde ein künstliches neuronales Netz trainiert, das lernt, wie ein Bild mit wenig Rechenzeit aussieht und wie es idealerweise aussehen sollte. Es kann dann auf neue Bilder angewendet werden, um diese deutlich zu verbessern.

Die Bildqualität konnte dadurch messbar gesteigert werden: Der durchschnittliche Unterschied zur Originalqualität wurde um rund 70 % verringert. Gleichzeitig verbesserte sich die visuelle Ähnlichkeit zu den perfekten Bildern deutlich. Subjektiv bedeutet das: Die vom Modell bearbeiteten Bilder wirken deutlich sauberer, detailreicher und natürlicher als die ursprünglichen verrauschten Versionen – und das ohne zusätzliche Rechenzeit beim Rendern.

Damit zeigt diese Arbeit das Potenzial von Deep Learning zur Beschleunigung realistischer Bildsynthese. Besonders in Anwendungsfeldern wie Animation, Produktvisualisierung oder Computerspielen kann dies in Zukunft eine wichtige Rolle spielen.

Inhaltsverzeichnis

[Path Tracing Denoising mit neuronalen Netzen 1](#_Toc199957606)

[Management Summary 2](#_Toc199957607)

[Inhaltsverzeichnis 3](#_Toc199957608)

[1 Einleitung 5](#_Toc199957609)

[1.1 Path Tracing 6](#_Toc199957610)

[1.1.1 Rendering-Gleichung 6](#_Toc199957611)

[1.1.2 Monte-Carlo-Integration beim Path Tracing 6](#_Toc199957612)

[1.1.3 Path Tracing Algorithmus 7](#_Toc199957613)

[1.1.4 Path Tracing Rauschen 8](#_Toc199957614)

[1.2 Denoising 9](#_Toc199957615)

[1.2.1 Klassische Filtermethoden 9](#_Toc199957616)

[1.2.2 Statistische Modelle 9](#_Toc199957617)

[1.2.3 Lernbasierte Methoden 9](#_Toc199957618)

[2 Projektmanagement 10](#_Toc199957619)

[2.1 Projektziele 10](#_Toc199957620)

[2.1.1 Theoretische Untersuchung von Path Tracing und Denoising 10](#_Toc199957621)

[2.1.2 Datensatzgenerierung 10](#_Toc199957622)

[2.1.3 Entwicklung und Training eines Denoising-Modells 10](#_Toc199957623)

[2.1.4 Vergleich mit bestehenden Lösungen 10](#_Toc199957624)

[2.2 Meilensteine 10](#_Toc199957625)

[2.2.1 Planung 10](#_Toc199957626)

[2.2.2 Forschung 11](#_Toc199957627)

[2.2.3 Datensatz 11](#_Toc199957628)

[2.2.4 Modell 11](#_Toc199957629)

[2.2.5 Präsentation und Dokumentation 11](#_Toc199957630)

[2.3 Projektablauf 12](#_Toc199957631)

[3 Implementation 13](#_Toc199957632)

[3.1 Erstellung des Datensatzes 13](#_Toc199957633)

[3.1.1 Szenenvorbereitung in Blender 13](#_Toc199957634)

[3.1.2 Automatisierte Bildgenerierung per Python-Skript 13](#_Toc199957635)

[3.1.3 Bildausschnitte 15](#_Toc199957636)

[3.1.4 Organisation und Verwendung des Datensatzes 15](#_Toc199957637)

[3.2 Erstellung des Modells 15](#_Toc199957638)

[3.2.1 Modell Architektur 15](#_Toc199957639)

[3.2.2 Optimierung der Hyperparameter 16](#_Toc199957640)

[3.2.3 Schutz vor Overfitting 16](#_Toc199957641)

[3.3 Modell Training 17](#_Toc199957642)

[3.4 Grafische Anwendung 18](#_Toc199957643)

[3.5 Test und Validierung 18](#_Toc199957644)

[4 Resultate 20](#_Toc199957645)

[5 Abbildungsverzeichnis 22](#_Toc199957646)

[6 Tabellenverzeichnis 22](#_Toc199957647)

[7 Glossar 22](#_Toc199957648)

[8 Literaturverzeichnis 22](#_Toc199957649)

[9 Anhang 23](#_Toc199957650)

[9.1 Meeting Protokoll 23](#_Toc199957651)

[9.1.1 Kick-off Meeting (24.02.2025 09:00) 23](#_Toc199957652)

[9.1.2 Abgleich 1 (13.03.2025 11:00) 23](#_Toc199957653)

[9.1.3 Abgleich 2 (24.03.2025 09:00) 23](#_Toc199957654)

[9.1.4 Expertentreff (07.04.2025 09:00) 24](#_Toc199957655)

[9.1.5 Abgleich 3 (22.04.2025 09:00) 24](#_Toc199957656)

[9.1.6 Abgleich 4 (05.05.2025 09:00) 24](#_Toc199957657)

[9.1.7 Abgleich 5 (19.05.2025 09:00) 25](#_Toc199957658)

[9.1.8 Abgleich 6 (02.06.2025 09:00) 25](#_Toc199957659)

[10 Selbständigkeitserklärung 26](#_Toc199957660)

# Einleitung

Path Tracing ist eine physikalisch genaue Rendering-Technik, die in der Computergrafik weit verbreitet ist, um das Verhalten von Licht zu simulieren. Sie ist in der Lage, äusserst realistische Bilder zu erzeugen, indem sie die Pfade einzelner Lichtstrahlen bei der Interaktion mit Oberflächen in einer Szene nachzeichnet. Diese Methode ist jedoch rechenintensiv und anfällig für Rauschen, insbesondere in Szenarien mit komplexen Beleuchtungseffekten wie globaler Beleuchtung, Kaustik und indirekter Beleuchtung.

Die Hauptquelle des Rauschens beim Path Tracing liegt in der stochastischen Natur der Monte-Carlo-Integration, die zur Annäherung an die Rendering-Gleichung verwendet wird. Da die Farbe jedes Pixels durch Mittelwertbildung aus einer endlichen Anzahl zufälliger Lichtpfade geschätzt wird, führt eine geringe Anzahl von Stichproben zu einer hohen Varianz, die sich im endgültigen Bild als körniges oder fleckiges Rauschen äussert. Eine Erhöhung der Anzahl der Abtastwerte pro Pixel kann das Rauschen zwar verringern, doch ist dieser Ansatz aufgrund der ansteigenden Rendering-Zeit oft nicht praktikabel.

Um dieses Problem zu lösen, wurden Entrauschungstechniken entwickelt, welche qualitativ hochwertige Bilder mit deutlich weniger Stichproben erzeugen. Diese Entrauschungsverfahren nutzen statistische, analytische und auf maschinellem Lernen basierende Methoden, um ein sauberes Bild aus einem verrauschten Eingangssignal zu rekonstruieren. Herkömmliche Entrauschungsmethoden beruhen auf Filtertechniken, die verrauschte Regionen glätten und dabei wichtige Details erhalten, während moderne, auf Deep Learning basierende Entrauschungsmethoden neuronale Netzwerke verwenden, die auf grossen Datensätzen trainiert wurden, um Rauschen effektiv vorherzusagen und zu entfernen. Durch die intelligente Unterscheidung von Rauschen und echten Details ermöglichen diese Verfahren schnellere Rendering-Workflows und machen Path Tracing für Echtzeitanwendungen wie interaktives Rendering und Spielgrafik möglich.

Diese Arbeit erforscht die Rauschunterdrückung beim Path Tracing mit Verwendung von neuronalen Netzwerken.

## Path Tracing

Path Tracing ist ein globaler Beleuchtungs-Rendering-Algorithmus, der die Physik des Lichttransports genau simuliert. Im Gegensatz zu rasterbasierten Methoden, die sich auf Heuristiken stützen, um Beleuchtungseffekte anzunähern, löst Path Tracing die Rendering-Gleichung direkt durch die Verfolgung von Lichtpfaden auf physikalische Weise. Aufgrund der Komplexität der Lichtinteraktionen in realen Szenen ist es nicht möglich, die Rendering-Gleichung analytisch zu lösen. Stattdessen kommt beim Path Tracing die Monte-Carlo-Integration zum Einsatz, ein statistisches Verfahren, das die Gleichung mithilfe von Zufallsstichproben approximiert.

### Rendering-Gleichung

Die Rendering-Gleichung, die von Kajiya (1986) eingeführt wurde, beschreibt den Transport von Licht in einer Szene:

* ist die ausgehende Strahldichte (radiance) am Punkt x in Richtung .
* ist die emittierte Strahldichte (radiance) von x, zum Beispiel von einer Lichtquelle.
* ist die bidirektionale Reflexionsverteilungsfunktion (BRDF), die beschreibt, wie das Licht gestreut wird.
* ist die einfallende Strahldichte (radiance) aus der Richtung .
* berücksichtigt den Kosinus des Einfallswinkels und sorgt für eine korrekte Energieerhaltung.
* Das Integral summiert die Beiträge aus allen einfallenden Lichtrichtungen über die Hemisphäre .

Diese Gleichung modelliert, wie Licht mit Oberflächen interagiert. Aber sie direkt für jedes Pixel zu lösen, ist aufgrund ihrer rekursiven Natur unpraktisch. Licht, das an einem Punkt ankommt, kann mehrfach zurückgeworfen worden sein, bevor es die Kamera erreicht.

### Monte-Carlo-Integration beim Path Tracing

Die Monte-Carlo-Integration bietet eine praktische Möglichkeit zur Annäherung an die Rendering-Gleichung, indem eine endliche Anzahl von Stichproben ausgewertet wird, anstatt das vollständige Integral analytisch zu berechnen. Die Monte-Carlo-Approximation für ein Integral der Form

wird gegeben durch:

wobei:

* ist die Anzahl der Proben.
* sind zufällig ausgewählte Punkte in der Domäne .
* ist die Wahrscheinlichkeitsdichtefunktion (PDF), für die Stichprobe .

Durch Anwendung auf die Rendering-Gleichung schätzt der Path-Tracing-Algorithmus die ausgehende Strahldichte, indem er Lichtpfade abtastet und deren Beiträge mittelt:

### Path Tracing Algorithmus

Path Tracing folgt einem stochastischen Ansatz zur Annäherung an die globale Beleuchtung:

1. Strahlenerzeugung: Für jedes Pixel wird ein Primärstrahl von der Kamera in die Szene geworfen.
2. Strahlenschnittpunkt: Der erste Schnittpunkt mit einer Oberfläche wird gefunden.
3. Schattierung und BRDF-Sampling: Die Oberflächen-BRDF bestimmt, wie das Licht reflektiert oder durchgelassen wird.
4. Rekursive Pfadverfolgung: Eine neue Richtung wird auf der Grundlage der BRDF abgetastet, und der Prozess wird fortgesetzt, bis eine Abbruchbedingung erfüllt ist. Zum Beispiel das Erreichen einer Lichtquelle oder maximaler Tiefe.
5. Monte-Carlo-Schätzung: Die akkumulierte Strahldichte entlang des Pfades wird gemittelt, um die Pixelfarbe zu schätzen.

Jeder Pfad ist eine Monte-Carlo-Stichprobe der Rendering-Gleichung. Bei einer ausreichenden Anzahl von Pfaden konvergiert die Annäherung zur richtigen Lösung.

Die Monte-Carlo-Integration ist zwar unvoreingenommen, konvergiert aber langsam und erfordert Tausende von Stichproben pro Pixel, um rauschfreie Ergebnisse zu erzielen. Niedrige Stichprobenzahlen führen zu Monte-Carlo-Rauschen, das als körnige Artefakte sichtbar wird. Dies ist die Hauptmotivation für Rauschunterdrückungstechniken, die darauf abzielen, qualitativ hochwertige Bilder aus verrauschten Renderings mit geringer Abtastung zu rekonstruieren.

### Path Tracing Rauschen

Das durch Path Tracing erzeugte Rauschen hat deutliche Merkmale, die es vom natürlichen Bildrauschen unterscheiden. Es ist das Ergebnis des Monte-Carlo-Samplings, bei dem die Farbe jedes Pixels anhand einer begrenzten Anzahl von zufälligen Lichtpfaden geschätzt wird. Dies führt zu einer hochfrequenten Varianz, die als gesprenkelte oder körnige Muster erscheint, insbesondere in Bereichen mit komplexer Beleuchtung wie indirekter Beleuchtung, Kaustik und weichen Schatten. Im Gegensatz zu natürlichem Rauschen, wie etwa dem Sensorrauschen von Digitalkameras, das in der Regel zufällig, additiv und statistisch gleichmässig über das Bild verteilt ist, ist das Rauschen durch Path Tracing stark strukturiert und szenenabhängig. Seine Intensität und Verteilung variiert mit der Anzahl der Proben pro Pixel, den Oberflächenmaterialien, den Lichtverhältnissen und der geometrischen Komplexität. Natürliches Rauschen ist in der Regel unabhängig vom Bildinhalt, während das Rauschen durch Path Tracing stark mit den visuellen und physikalischen Eigenschaften der Szene korreliert.



Abbildung 1: Verrauschtes Bücherregal mit Blender (Cycles)

## Denoising

Denoising bezeichnet den Prozess der Entfernung von unerwünschtem Rauschen aus digitalen Bildern, ohne dabei relevante Details oder Strukturen zu verfälschen. Insbesondere in der Path-Tracing-basierten Bildsynthese ist Denoising ein essenzieller Bestandteil der Nachbearbeitung, da das visuelle Rauschen bei geringen Samples pro Pixel deutlich sichtbar ist. Ziel ist es, aus verrauschten Zwischenergebnissen möglichst hochwertige, rauschfreie Bilder zu rekonstruieren.

In der Bildverarbeitung existieren verschiedene Ansätze zur Rauschunterdrückung, die sich hinsichtlich Methodik, Komplexität und Anwendungsbereich unterscheiden. Grundsätzlich lassen sich Denoising-Methoden in drei Hauptkategorien einteilen: klassische filterbasierte Verfahren, statistische bzw. modellbasierte Verfahren und lernbasierte Methoden.

### Klassische Filtermethoden

Zu den einfachsten Ansätzen zählen lineare und nichtlineare Filter, die direkt auf das verrauschte Bild angewendet werden. Bekannte Beispiele sind:

* **Mittelwertfilter**: Glättet das Bild durch Ersetzen jedes Pixels mit dem Durchschnittswert seiner Nachbarn. Dieser Ansatz reduziert Rauschen, führt jedoch oft zu sichtbarem Detailverlust.
* **Medianfilter**: Ersetzt jeden Pixel mit dem Median seiner Umgebung. Besonders effektiv bei impulsartigem ("Salt-and-Pepper") Rauschen.
* **Gaussian-Blur**: Wendet eine gewichtete Glättung an, wobei näherliegende Pixel stärker berücksichtigt werden. Geeignet zur Rauschreduktion mit geringem Detailverlust.

Diese Methoden sind schnell und leicht implementierbar, stossen aber bei komplexeren Rauschmustern, wie sie beim Path Tracing auftreten, schnell an ihre Grenzen.

### Statistische Modelle

Moderne bildverarbeitende Verfahren nutzen statistische Modelle und Annahmen über die Bildstruktur:

* **Wiener-Filter**: Ein lineares Verfahren, das das Rauschen unter Kenntnis der Signal- und Rauschstatistik optimal unterdrückt.
* **BM3D (Block-Matching and 3D Filtering)**: Ein nichtlineares Verfahren, das auf blockweiser Ähnlichkeit basiert. Es gruppiert ähnliche Bildbereiche, transformiert sie gemeinsam in einen Frequenzbereich und filtert dort gezielt Rauschen heraus. BM3D gilt lange Zeit als einer der leistungsstärksten klassischen Algorithmen für Denoising.

Diese Methoden erzielen oft bessere Ergebnisse als einfache Filter, sind jedoch rechenintensiver und schwer auf stark strukturierte oder nicht-stationäre Rauscharten anzupassen.

### Lernbasierte Methoden

Mit dem Aufkommen leistungsfähiger Hardware und grosser Bilddatensätze haben sich neuronale Netze als äusserst effektive Werkzeuge zur Rauschunterdrückung etabliert. Hierbei wird ein Modell auf verrauschten und sauberen Bildpaaren trainiert, um eine Abbildung vom verrauschten zum rauschfreien Bild zu erlernen.

* **Convolutional Neural Networks (CNNs)**: Klassische Deep-Learning-Modelle, die auf lokalen Bildmerkmalen basieren. Sie sind in der Lage, komplexe Rauschmuster zu erkennen und gezielt zu entfernen.
* **Autoencoder**: Netzwerke, die lernen, eine komprimierte Darstellung des Bildes zu erzeugen und daraus die rauschfreie Version zu rekonstruieren.
* **U-Net-Architekturen**: Erweiterte Autoencoder mit Skip Connections, die besonders in der medizinischen Bildverarbeitung und beim Denoising in der Computergrafik verwendet werden.
* **Recurrent und Transformer-basierte Modelle**: Für spezielle Denoising-Aufgaben, insbesondere bei zeitlich korrelierten Bildern (wie Rendersequenzen), können auch rekurrente Strukturen oder Attention-Mechanismen sinnvoll sein.

Lernbasierte Methoden bieten den grossen Vorteil, dass sie an spezifische Rauschtypen angepasst und durch Training optimiert werden können. In der Praxis zeigen sie bei realistischen Renderbildern oft deutlich bessere Resultate als klassische Verfahren – insbesondere bei niedrigem Sample Count.

# Projektmanagement

## Projektziele

Im Rahmen dieser Arbeit wurde das Ziel verfolgt, sich tiefgehend mit den Konzepten des Path Tracing und dem Denoising auseinanderzusetzen. Aufbauend auf theoretischem Wissen sollten geeignete Methoden zur Rauschunterdrückung analysiert, ein Datensatz generiert und ein neuronales Netz zur Bildverbesserung trainiert werden. Die konkreten Ziele der Arbeit waren:

### Theoretische Untersuchung von Path Tracing und Denoising

* Erarbeitung eines fundierten Verständnisses der Path-Tracing-Technik im Kontext der globalen Beleuchtung.
* Analyse der Ursachen für Bildrauschen in Path-Tracing-Renderings.
* Recherche und Bewertung bestehender Denoising-Techniken – insbesondere unter Verwendung neuronaler Netze.

### Datensatzgenerierung

* Untersuchung öffentlich verfügbarer Denoising-Datensätze auf die Eignung für Path-Tracing-Anwendungen.
* Falls erforderlich, Generierung eines eigenen Datensatzes mit Blender unter Verwendung unterschiedlicher Sample-Raten und einer Ground-Truth.

### Entwicklung und Training eines Denoising-Modells

* Entwicklung eines neuronalen Netzes zur Rauschunterdrückung, insbesondere in Form eines Autoencoders oder U-Net-ähnlichen Modells.
* Training des Modells auf den generierten Datensätzen unter Verwendung geeigneter Verlustfunktionen.
* Evaluierung der Ergebnisse auf Basis visueller Qualität und quantitativer Metriken.

### Vergleich mit bestehenden Lösungen

* Vergleich der erzielten Resultate mit bestehenden Denoising-Verfahren, z. B. dem OptiX Denoiser von NVIDIA oder den integrierten Denoisern in Blender.
* Analyse der Stärken und Schwächen des eigenen Ansatzes im Vergleich zu etablierten Methoden.

**Hinweis zur entfernten Zielsetzung:**

Die ursprünglich geplante Implementierung eines eigenen Path Tracers oder die Nutzung des Path Tracers aus dem SLProject4 wurde im Verlauf der Arbeit verworfen, da der Fokus stärker auf der Bildverarbeitung mittels neuronaler Netze lag. Stattdessen wurde auf extern gerenderte Daten mit kontrollierbarem Rauschverhalten (z. B. durch Blender) zurückgegriffen.

## Meilensteine

Im Folgenden werden die einzelnen Meilensteine des Projekts detailliert beschrieben. Sie dienen der zeitlichen und inhaltlichen Strukturierung der Arbeit und bilden die Grundlage für das Projektmanagement sowie die Überprüfung des Fortschritts.

### Planung

* Regelmässige Meetings mit dem Betreuer wurden zu Beginn des Projekts vereinbart. Diese dienen dem fortlaufenden Austausch, der Diskussion des Fortschritts sowie der Klärung offener Fragen.
* Die übergeordneten Projektziele sowie die konkreten Arbeitspakete und Meilensteine wurden in Absprache mit dem Betreuer definiert und dokumentiert.
* Ein detaillierter Projektzeitplan wurde erstellt, der die einzelnen Phasen – von der Recherche bis zur Präsentation – umfasst. Dieser Plan dient als Leitfaden für die termingerechte Umsetzung der Projektziele.

### Forschung

* Eine umfassende Literaturrecherche zum Thema Path Tracing sowie zu aktuellen Ansätzen im Bereich der Denoising wurde durchgeführt. Die gewonnenen Erkenntnisse wurden dokumentiert und bilden die theoretische Grundlage für die weitere Arbeit.
* Der im SLProject4 enthaltene Path Tracer wurde analysiert. Dabei wurden bestehende Fehler identifiziert und – soweit möglich – behoben.

### Datensatz

* Bestehende öffentliche Datensätze im Bereich des Bilddenoising wurden recherchiert und auf ihre Eignung für die spezifischen Anforderungen des Projekts hin evaluiert.
* Sollte ein geeigneter Datensatz gefunden worden sein, wurde dieser übernommen und in die Projektinfrastruktur integriert.
* Falls kein passender Datensatz verfügbar war, wurde mithilfe von Blender ein eigener Datensatz generiert. Dabei wurden sowohl verrauschte als auch entrauschte Renderings mithilfe unterschiedlicher Sampling-Raten und der Cycles-Engine erstellt. Der Datensatz wurde strukturiert gespeichert und für das Training der Modelle aufbereitet.

### Modell

* Basierend auf dem erstellten oder ausgewählten Datensatz wurde ein erstes neuronales Netzwerk-Modell implementiert. Dieses wurde auf die Aufgabe des Bilddenoisings trainiert. Das Modell ist in der Lage, verrauschte Bilder zu verarbeiten und eine entrauschte Version als Ausgabe zu liefern.
* Falls unterschiedliche Modellansätze (z. B. Autoencoder, CNN, U-Net) als sinnvoll erachtet wurden, wurden mindestens zwei verschiedene Architekturen entwickelt, trainiert und miteinander verglichen. Die Auswahl der Modelle basierte auf ihrer Leistungsfähigkeit und Komplexität.

### Präsentation und Dokumentation

* Alle Präsentationsmedien – darunter die schriftliche Dokumentation, ein Erklärvideo, ein Poster sowie die Folien für den Vortrag – wurden erstellt, abgestimmt und finalisiert.
* Die abschliessende Präsentation wurde geübt, um einen sicheren und professionellen Auftritt während der Ausstellung zu gewährleisten.
* Zusätzlich wurde der Aufbau für die Abschlussausstellung geplant und vorbereitet, um die Projektergebnisse anschaulich zu präsentieren.

## Projektablauf

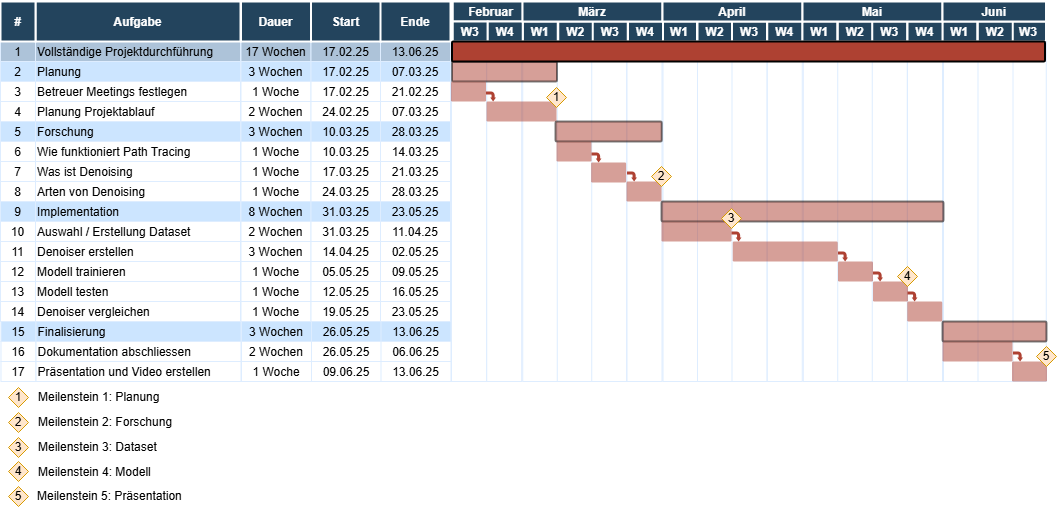


Abbildung 2: Projektablauf Gantt Diagramm

# Implementation

## Erstellung des Datensatzes

Für das Training und die Evaluierung des neuronalen Netzes zur Rauschunterdrückung wurde ein eigener Datensatz erstellt, der synthetisch mit Blender generiert wurde. Ziel war es, realitätsnahe gerenderte Bilder mit definierten Rausch-leveln zu erzeugen und diese mit qualitativ hochwertigen Referenzbildern (Ground Truth) zu paaren. Die Erstellung des Datensatzes erfolgte in mehreren automatisierten und manuellen Schritten.

### Szenenvorbereitung in Blender

Zur Datengenerierung wurden mehrere 3D-Szenen in Blender vorbereitet. Dafür wurden kostenlose öffentliche Szenen von Blender verwendet. In jeder Szene wurden mehrere Kamerapositionen manuell platziert, um verschiedene Blickwinkel und Bildinhalte zu erfassen. Die Kameras wurden so positioniert, dass sie unterschiedliche Bereiche der Szene abdecken, um eine möglichst hohe Varianz im Datensatz zu gewährleisten. Für jede Kamera wurde anschliessend eine Reihe von Bildern aus derselben Szene, aber mit unterschiedlichen Parametern erzeugt.

### Automatisierte Bildgenerierung per Python-Skript

Ein zentrales Element der Datensatz-Generierung ist ein eigens entwickeltes Python-Skript, das Blender im sogenannten Headless-Modus (d. h. ohne Benutzeroberfläche) startet. Dies ermöglicht eine vollautomatisierte Verarbeitung und eignet sich besonders für das Rendering auf Servern oder in CI-Umgebungen ohne grafische Oberfläche.

Das Skript verarbeitet jede vorbereitete Szene sequenziell und führt folgende Schritte aus:

* Öffnen der .blend-Datei.
* Initialisierung der Blender-Cycles-Renderengine.
* Iteration über alle definierten Kameras.
* Für jede Kamera werden Bilder mit unterschiedlichen Samples per Pixel (SPP) gerendert: **2, 5,** 10, 25, 50, 100, 200, 500. Diese Bilder enthalten das gewünschte Path Tracing Rauschen und dienen als Trainingsinput.
* Für jede Kamera-Perspektive wird zusätzlich ein Ground-Truth-Bild generiert. Dieses verwendet eine hohe Sampleanzahl in Kombination mit dem OptiX-Denoiser von Blender, um ein visuell möglichst rauschfreies Referenzbild zu erhalten.

Die Bilder werden in einer Auflösung von 1024 × 1024 Pixeln gespeichert. Alle Ausgaben werden systematisch benannt, um eine eindeutige Zuordnung zwischen verrauschten und rauschfreien Bildern zu ermöglichen.

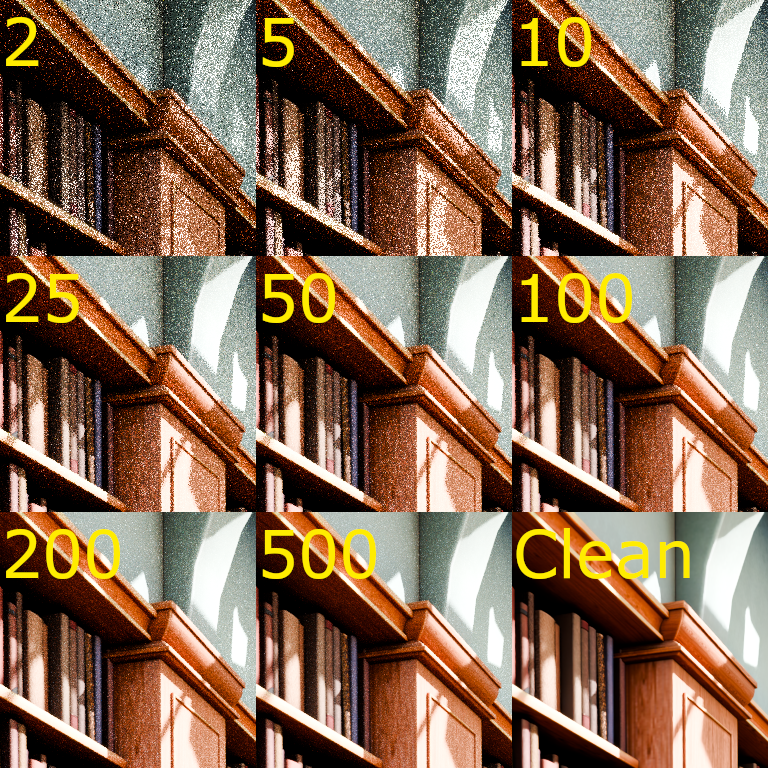


Abbildung 3: Visualisierung der verschiedenen Rauschstufen eines Patches

Die obige Abbildung zeigt ein Beispiel eines Bildpatches aus dem Datensatz. Es sind insgesamt neun Varianten desselben Ausschnitts dargestellt – acht mit unterschiedlichem Rauschgrad, erzeugt durch verschiedene Sample-Werte (Samples per Pixel), sowie ein entrauschtes Referenzbild (Ground Truth). Die Sample-Anzahl ist jeweils oben links im Bild angegeben; das Referenzbild ist mit "Clean" gekennzeichnet. Die zunehmende Bildqualität mit steigender Sample-Zahl ist deutlich erkennbar.

### Bildausschnitte

Da neuronale Netze, insbesondere Autoencoder, typischerweise auf kleinere Eingabedimensionen trainiert werden, wurden die 1024 × 1024 grossen gerenderte Bilder in kleinere Bildausschnitte (Patches) unterteilt. Ein separates Python-Skript übernimmt diese Aufgabe:

* Jedes Bild wird in 16 nicht überlappende Patches der Grösse 256 × 256 Pixel unterteilt.
* Dabei werden sowohl die verrauschten Bilder als auch die entsprechenden Ground-Truth-Bilder verarbeitet.
* Die resultierenden Patches werden mit einem konsistenten Namensschema abgespeichert, das folgende Informationen enthält:
  + Szenenname
  + Kameranummer
  + Sampleanzahl (für verrauschte Bilder)
  + Patch-Index

Beispielhafte Dateinamen:

* Scene-name\_cam\_0\_noisy\_10\_patch\_5.png
* Scene-name\_cam\_0\_clean\_patch\_5.png

Diese Struktur ermöglicht eine effiziente Zuordnung der Trainingsdaten während des Ladevorgangs in das neuronale Netz.

### Organisation und Verwendung des Datensatzes

Der vollständige Datensatz besteht aus 13056 Patch-Paaren (verrauschtes Eingabebild + Ground Truth). Die Daten wurden in drei Teilmengen aufgeteilt:

* Training – 9856 Patch-Paare
* Validierung – 1792 Patch-Paare
* Test – 1408 Patch-Paare

Die Verzeichnisse sind entsprechend gegliedert, um eine saubere Trennung für die Modellentwicklung zu gewährleisten. In welche Teilmenge ein Bild kommt wird schon bei der Blender Szene entschieden. Diese sind ebenfalls in die Verzeichnisse «train», «val» und «test» unterteilt. Ist eine Blender Szene im Testverzeichnis kommen die generierten Bilder automatisch in den Testdatensatz.

## Erstellung des Modells

Zur Entrauschung der gerenderten Bilder wurde ein Autoencoder-Modell im Stil eines U-Net entwickelt. Dabei handelt es sich um ein neuronales Netzwerk, das speziell darauf ausgelegt ist, Bildinformationen zu verarbeiten und gleichzeitig unerwünschtes Rauschen zu entfernen. Die Struktur des Netzwerks ist symmetrisch aufgebaut: Ein Teil des Netzwerks (der Encoder) verkleinert schrittweise die Bildinformationen, um sie in einer kompakten Form zusammenzufassen. Der andere Teil (der Decoder) vergrössert diese Darstellung wieder, sodass am Ende ein rauschfreies Bild entsteht.

### Modell Architektur

Der Encoder besteht aus mehreren Verarbeitungsschritten, bei denen das Bild immer weiter verkleinert wird, während gleichzeitig die wichtigsten Merkmale extrahiert werden. Das Ziel ist es, eine kompakte Darstellung zu erzeugen, die die wesentlichen Bildinhalte enthält. Diese Verdichtung endet in einem sogenannten "Bottleneck" (engl. für Flaschenhals), einem zentralen Abschnitt des Netzwerks, der nur eine stark reduzierte Version des Bildes enthält. Hier entscheidet sich, wie gut das Modell in der Lage ist, die Bildstruktur zu verstehen und später zu rekonstruieren.

Anschliessend beginnt der Decoder, der das Bild Schritt für Schritt wieder in die ursprüngliche Auflösung bringt. Dabei werden nicht nur die verdichteten Informationen aus dem Bottleneck verwendet, sondern auch sogenannte Skip-Connections. Diese verbinden frühere Verarbeitungsschritte des Encoders direkt mit passenden Stellen im Decoder. So kann das Modell auf Details zurückgreifen, die sonst beim Verkleinern des Bildes, verloren gehen würden. Man kann sich Skip-Connections wie Abkürzungen vorstellen, die es dem Netzwerk ermöglichen, wichtige Bildinformationen schneller und verlustfreier wiederzuverwenden.

### Optimierung der Hyperparameter

Um das Modell möglichst effektiv zu trainieren, wurde die Optimierungsbibliothek Optuna eingesetzt. Sie hilft dabei, wichtige Einstellungswerte des Trainings automatisch zu testen und zu verbessern. Dazu gehören unter anderem die sogenannte Batch-Grösse (wie viele Bilder gleichzeitig verarbeitet werden), die Lernrate (wie schnell das Modell lernt), die Anzahl der Kanäle in den Verarbeitungsschritten (also wie viele unterschiedliche Merkmale betrachtet werden) sowie Einstellungen für den verwendeten Optimierungsalgorithmus.

Bei der automatisierten Optimierung mit Optuna wurden gezielt verschiedene sogenannte Hyperparameter getestet. Diese bestimmen grundlegende Eigenschaften des Trainingsprozesses und der Modellstruktur. Die wichtigsten dabei waren:

* **Batch-Grösse** (batch\_size): Gibt an, wie viele Bilder gleichzeitig verarbeitet werden. Getestet wurden Werte von 32 und 64.
* **Lernrate** (lr): Bestimmt, wie stark das Modell seine Gewichte bei jedem Lernschritt anpasst. Werte wurden im Bereich von 0.00001 bis 0.01 ausprobiert (in logarithmischen Abstufungen).
* **Anzahl der Basis-Kanäle** (base\_channels): Gibt die Breite der ersten Verarbeitungsschicht an und beeinflusst damit direkt die Tiefe und Kapazität des Modells. Es wurden 64 und 128 Kanäle getestet.
* **Beta1 und Beta2** (beta1, beta2): Diese beiden Parameter gehören zum verwendeten Adam-Optimierer und regeln, wie stark frühere Gradientenverläufe berücksichtigt werden. Die getesteten Bereiche lagen bei:
  + Beta1: zwischen 0.8 und 0.99
  + Beta2: zwischen 0.9 und 0.999
* **Epsilon** (eps): Eine kleine Zahl, die zur Vermeidung von Division durch Null dient. Getestet wurden Werte zwischen 1e-9 und 1e-6.
* **Gewichtsverfall** (weight\_decay): Eine Form der Regularisierung, die sehr grosse Gewichtswerte vermeidet und damit Overfitting entgegenwirkt. Der Bereich lag zwischen 1e-6 und 1e-2.

Jede Kombination dieser Werte wurde von Optuna bewertet, indem sie mit einem Trainingslauf getestet wurde. So konnte die beste Zusammenstellung gefunden werden, ohne dass man alle denkbaren Kombinationen manuell ausprobieren musste. Dieses Verfahren sparte nicht nur viel Zeit, sondern führte auch zu besseren Ergebnissen, als es mit festen Standardwerten möglich gewesen wäre.

Folgende Werte wurden von Optuna als beste Hyperparameter gefunden.

{

"batch\_size": 32,

"lr": 0.00010135211376349008,

"base\_channels": 64,

"beta1": 0.9175359182964491,

"beta2": 0.9757206708695416,

"eps": 3.422801253803442e-09,

"weight\_decay": 9.692445546369089e-05

}

### Schutz vor Overfitting

Ein wichtiges Ziel beim Training eines neuronalen Netzwerks ist es, dass es nicht nur die Trainingsbilder auswendig lernt, sondern auch auf neue, unbekannte Bilder gute Ergebnisse liefert. Dieses Problem nennt man Overfitting (engl. für Überanpassung). Um dem entgegenzuwirken, wurden mehrere Strategien eingesetzt:

* **Validierungsdaten**: Ein Teil der Bilddaten wurde nicht zum Lernen verwendet, sondern nur zur Überprüfung, wie gut das Modell auf unbekannten Bildern funktioniert.
* **Frühzeitiges Abbrechen**: Wenn sich die Qualität auf den Validierungsdaten über längere Zeit nicht verbessert hat, wurde das Training automatisch gestoppt.
* **Regulierung der Gewichte**: Das Netzwerk wurde leicht dazu "bestraft", wenn es sehr extreme Werte für seine internen Verbindungen verwendet hat. So bleibt es stabiler und verallgemeinert besser.

## Modell Training

Das Training des Denoising-Autoencoders erfolgte auf einer leistungsstarken Serverplattform mit einer NVIDIA A100 Grafikkarte, die speziell für rechenintensive Aufgaben wie das Training neuronaler Netzwerke ausgelegt ist. Als Softwaregrundlage wurde PyTorch in der Version 2.6.0 verwendet, das auf CUDA 12.6 basiert, um die GPU-Beschleunigung optimal zu nutzen.

Während des Trainings wurden alle relevanten Messwerte und Fortschritte mit TensorBoard (Version 2.19.0) protokolliert. Dies ermöglichte eine übersichtliche Visualisierung von Kennzahlen wie Verlustwerten und Lernraten während des Trainings und erleichterte so die Überwachung und Optimierung des Modells.

Ein Bild, das Text, Diagramm, Reihe, Screenshot enthält.

KI-generierte Inhalte können fehlerhaft sein.

Abbildung 4: Verlauf des Training- und Validierungsverlusts

Die Abbildung zeigt den Verlauf des Trainings- und Validierungsverlusts über die Trainingsschritte hinweg. Beide Kurven basieren auf Werten, die während des Trainingsprozesses mithilfe von TensorBoard aufgezeichnet wurden. Ein niedrigerer Verlustwert deutet auf eine bessere Modellleistung hin, da der Unterschied zwischen dem vorhergesagten und dem tatsächlichen Bild geringer ist. Idealerweise nähern sich beide Kurven mit fortschreitendem Training einem stabilen, niedrigen Wert an. Obwohl der Trainingsverlust am Ende sich weiter verbessert, wurde das Training wegen dem stagnierten Validierungsverlusts gestoppt.

Für Leserinnen und Leser, die das Modell selbst trainieren oder anpassen möchten, sind im zugehörigen GitHub-Repository detaillierte Anleitungen zur Installation und Einrichtung der benötigten Softwarekomponenten im README-Dokument enthalten.

## Grafische Anwendung

Im Rahmen dieser Arbeit wurde eine grafische Benutzeroberfläche (GUI) entwickelt, die es ermöglicht, beliebig grosse verrauschte Bilder einzulesen, automatisch zu entrauschen und die Resultate direkt vergleichend anzuzeigen. Hierbei wird das Bild in passende Teilstücke (Patches) zerlegt, welche einzeln durch das trainierte neuronale Modell zum Denoising verarbeitet werden. Anschliessend werden die einzelnen Teile wieder zusammengesetzt, um das vollständige entrauschte Bild zu erhalten. Diese Vorgehensweise ermöglicht die Verarbeitung von Bildern beliebiger Grösse, ohne die Beschränkungen der Netzwerkarchitektur bezüglich der Eingabegrösse.

Die GUI zeigt jeweils entweder das originale verrauschte Bild oder das entrauschte Bild im Vollbildmodus an. Mit beliebigen Tastendrücken kann zwischen diesen beiden Darstellungen umgeschaltet werden. Zusätzlich stehen Zoom- und Verschiebefunktionen zur Verfügung, um Details an beliebiger Bildstelle genau zu betrachten.

Um die visuelle Qualitätsverbesserung objektiv zu messen, wird neben der subjektiven Betrachtung die BRISQUE-Metrik eingesetzt. BRISQUE (Blind/Referenceless Image Spatial Quality Evaluator) ist ein referenzfreies Bildqualitätsmass, das ohne Vergleich zum Originalbild die wahrgenommene Bildqualität quantifiziert. Durch die Berechnung von BRISQUE-Werten für das verrauschte und das entrauschte Bild kann die Verbesserung der Bildqualität numerisch erfasst und dem Nutzer angezeigt werden. Ein niedrigerer BRISQUE-Wert entspricht dabei einer besseren Bildqualität.

Darüber hinaus bietet die Anwendung die Möglichkeit, das entrauschte Bild auf Knopfdruck in einem Datei-Explorer zu speichern.

## Test und Validierung

Zur objektiven Bewertung der Leistungsfähigkeit des entwickelten Entrauschungsmodells wurde ein automatisiertes Testskript implementiert, welches das trainierte Modell auf dem gesamten Testdatensatz ausführt und Qualitätsmetriken berechnet. Das Skript lädt dazu alle Testbilder mit ihrem zugehörigen sauberen Referenzbild, führt eine Inferenz mit dem ONNX-Modell durch und wertet die Ergebnisse aus.

Dabei werden zwei wichtige Metriken berechnet:

* L1-Loss (Mean Absolute Error): Diese Metrik misst die durchschnittliche absolute Differenz zwischen den Pixelwerten des verrauschten bzw. entrauschten Bildes und des sauberen Referenzbildes. Ein niedrigerer Wert entspricht einer höheren Genauigkeit und besseren Entrauschung.
* SSIM (Structural Similarity Index Measure): SSIM quantifiziert die wahrgenommene strukturelle Ähnlichkeit zwischen zwei Bildern, wobei Werte näher bei 1 eine hohe Bildähnlichkeit anzeigen. Diese Metrik bewertet das visuelle Erscheinungsbild der Bilder besser als reine Pixelabstände.

Das Skript berechnet beide Metriken sowohl vor als auch nach der Entrauschung für jedes Bild im Testdatensatz. Zudem werden die Ergebnisse tabellarisch in einer CSV-Datei gespeichert, um eine spätere Analyse zu erleichtern.

Ein Bild, das Text, Screenshot, Diagramm, Schrift enthält.

KI-generierte Inhalte können fehlerhaft sein.Parallel dazu erzeugt das Skript einen Plot, der die L1- und SSIM-Werte für alle Testbilder visualisiert. Dieser Plot zeigt anschaulich die Verbesserung durch das Modell: Die roten und orangen Kurven repräsentieren die Werte vor dem Denoising (verrauschtes Bild), während die grünen und blauen Kurven die Werte nach dem Denoising darstellen. Typischerweise ist zu beobachten, dass der L1-Loss nach der Entrauschung deutlich sinkt und der SSIM-Wert steigt, was auf eine erfolgreich verbesserte Bildqualität hinweist. Die ersten 256 Bilder stammen aus der Szene «Cornell Box». Bei dieser scheint Cycles ein Problem zu haben, was zu unsauberen Ground-Truths führte. Deswegen fluktuieren die Werte im Graphen stark, obwohl das Ergebnis zufriedenstellend ist.  
Zusätzlich werden für jedes Testbild Vergleichsbilder gespeichert, die das verrauschte, das entrauschte sowie das saubere Bild nebeneinander darstellen. Diese Bilder ermöglichen eine direkte visuelle Kontrolle der Entrauschungsqualität.

Abbildung 5: Qualitätsmetriken auf dem Testdatensatz

# Resultate

Average L1 Loss (Noisy): 0.064722

Average L1 Loss (Denoised): 0.019576

Average SSIM (Noisy): 0.405321

Average SSIM (Denoised): 0.900885

Schlussfolgerungen/Fazit

Duis autem vel eum iriure dolor in hendrerit in vulputate velit esse molestie consequat, vel illum dolore eu feugiat nulla facilisis at vero eros et accumsan et iusto odio dignissim qui blandit praesent luptatum zzril delenit augue duis dolore te feugait nulla facilisi. Lorem ipsum dolor sit amet, consectetuer adipiscing elit, sed diam nonummy nibh euismod tincidunt ut laoreet dolore magna aliquam erat volutpat.

Ut wisi enim ad minim veniam, quis nostrud exerci tation ullamcorper suscipit lobortis nisl ut aliquip ex ea commodo consequat. Duis autem vel eum iriure dolor in hendrerit in vulputate velit.

# Abbildungsverzeichnis

[Abbildung 1: Verrauschtes Bücherregal mit Blender (Cycles) 8](#_Toc199952508)

[Abbildung 2: Projektablauf Gantt Diagramm 12](#_Toc199952509)

[Abbildung 3: Visualisierung der verschiedenen Rauschstufen eines Patches 14](#_Toc199952510)

[Abbildung 4: Verlauf des Training- und Validierungsverlusts 17](#_Toc199952511)

# Tabellenverzeichnis

Es konnten keine Einträge für ein Abbildungsverzeichnis gefunden werden.

# Glossar

**Denoising**

Englisch für Entrauschung oder Rauschunterdrückung. In dieser Arbeit ist besonders der Prozess gemeint, Rauschen aus Bildern zu entfernen.

**Overfitting**

Englisch für Überanpassung. Problem bei Neuronalen Netzen, die lediglich den Trainings-Datensatz auswendig lernen.

**Cowoll**

Et ut aut isti repuditis qui ium

# Literaturverzeichnis

**Literatureintrag**

*Autorname, Autorvorname, Buchtitel, Verlag, Ort, Ausgabe, Jahr* 7

**Literatureintrag**

*Autorname, Autorvorname, Buchtitel, Verlag, Ort, Ausgabe, Jahr* 9

**Literatureintrag**

*Autorname, Autorvorname, Buchtitel, Verlag, Ort, Ausgabe, Jahr* 11

# Anhang

## Meeting Protokoll

### Kick-off Meeting (24.02.2025 09:00)

**Standort**: Microsoft Teams Online Meeting

**Teilnehmer**:

* Prof. Marcus Hudritsch (Betreuer)
* Pascal Cornu (Studierender)

**Meeting Protokoll:**

* Prof. Marcus Hudritsch gibt Rahmenbedingungen, so wie den Zugewiesenen Experten (Dr. Harald Studer) durch.
* Zukünftige Meetings für den Abgleich werden alle zwei Wochen für das gesamte Projekt festgesetzt.
* Pascal Cornu zeigte seine bisherige Recherche in das Thema Path Tracing.
* Die nächsten Schritte, insbesondere die Planung und Recherche wurden diskutiert: Das Projekt «smallpt» soll selbst ausgeführt werden und mit dem Path Tracer von SLProject4 verglichen werden. Dadurch sollen die Artefakte vom SLProject4 Path Tracer untersucht werden. Zusätzlich soll ein einfacher Autoencoder selbst implementiert werden.

### Abgleich 1 (13.03.2025 11:00)

**Standort**: Microsoft Teams Online Meeting

**Teilnehmer**:

* Prof. Marcus Hudritsch (Betreuer)
* Pascal Cornu (Studierender)

**Meeting Protokoll:**

* Pascal Cornu zeigte seine weitere Recherche in die Themen Path Tracing und Denoising, so wie die das erlangte Wissen über die Unterschiede vom «smallpt» und dem SLProject4 Path Tracer. Die so genannten Fireflies, die Artefakte die im SLProject, nicht aber im «smallpt» vorhanden sind, wurden erklärt und ein bisschen eingedämmt.
* Der simple Autoencoder, der auf dem MNIST Datensatz basiert, wurde vorgestellt.
* Prof. Marcus Hudritsch gibt Feedback über die Projektplanung und teilt seine Kenntnisse über Denoising.
* Die nächsten Schritte werden geplant: Für den Denoising Autoencoder braucht es ein Datensatz. Öffentliche Datensätze sollen gesucht und analysiert werden.

### Abgleich 2 (24.03.2025 09:00)

**Standort**: Microsoft Teams Online Meeting

**Teilnehmer**:

* Prof. Marcus Hudritsch (Betreuer)
* Pascal Cornu (Studierender)

**Meeting Protokoll:**

* Pascal Cornu präsentiert zwei gefundene Datensätze, die für das Modell verwendet werden können.
* Prof. Hudritsch findet die Qualität der Datensätze nicht ausreichend. Ein eigener Datensatz soll synthetisch erstellt werden. Beispielsweise mit Blender in kostenlosen öffentlichen Szenen ein Bild generieren lassen nach 5 Sekunden rendern, 30 Sekunden, 1 Minute und so weiter.
* Für den Expertentreff soll eine PowerPoint Präsentation vorbereitet werden.

### Expertentreff (07.04.2025 09:00)

**Standort**: Biel, Rolex Gebäude, Zimmer N553

**Teilnehmer**:

* Prof. Marcus Hudritsch (Betreuer)
* Dr. Harald Studer (Experte)
* Pascal Cornu (Studierender)
* Nick Denzler, Kilian Wampfler, Tim Schär (Zuschauer)

**Meeting Protokoll:**

* Pascal Cornu stellt in einer 10 Minuten PowerPoint Präsentation folgende Themen vor:
  + Projektziele und Projektablauf
  + Erkenntnisse der Recherche über Path Tracing und Denoising
  + Arbeit am SLProject Denoiser
  + Generierung des Datensatzes mit Blender
* Dr. Harald Studer stellt mehrere Fragen über das Projekt, insbesondere über vergleichbare Arbeiten und die Einschränkung der Projektziele.
* Prof. Marcus Hudritsch stellt mehrere Fragen über den bisher erstellten Datensatz.

### Abgleich 3 (22.04.2025 09:00)

**Standort**: Microsoft Teams Online Meeting

**Teilnehmer**:

* Prof. Marcus Hudritsch (Betreuer)
* Pascal Cornu (Studierender)

**Meeting Protokoll:**

* Pascal Cornu präsentiert seine Erweiterung des Datensatzes mit folgenden Funktionen:
  + In den Blender Szenen können manuell weitere Kameras platziert werden, die mit dem Skript automatisch eingelesen werden, um die Grösse des Datensatzes zu erweitern.
  + Unterteilung der 1024 x 1024 Pixel Bilder in 16 nicht überlappende 256 x 256 Pixel Bilder. (Die verlangte Grösse des Modells)
* Pascal Cornu präsentiert einen simplen Autoencoder, der den erstellten Datensatz verwendet und beliebige 256 x 256 Pixel Bilder entrauschen kann. Die Qualität des Outputs ist jedoch noch nicht gut.
* Prof. Marcus Hudritsch gibt Feedback über den erstellten Datensatz. Er ist skeptisch über die Menge der Bilder, 10000 Bilder reichen womöglich nicht aus. Zusätzlich erläutert er, das bisher jedes Projekt mit einem eigenen Datensatz gescheitert ist.
* Die nächsten Schritte werden geplant: Die Architektur des Modells soll verbessert werden. Generell soll mehr Arbeit in das Modell gesteckt werden.

### Abgleich 4 (05.05.2025 09:00)

**Standort**: Microsoft Teams Online Meeting

**Teilnehmer**:

* Prof. Marcus Hudritsch (Betreuer)
* Pascal Cornu (Studierender)

**Meeting Protokoll:**

* Pascal Cornu präsentiert seine Fortschritte beim Modell:
  + Tiefere Modell Architektur mit «skip connections» und «double convolutions».
  + Optuna Hyperparameter Training wurde implementiert, aber noch nicht ausgeführt.
* Prof. Marcus Hudritsch gibt Feedback über den fehlenden Schutz gegen Overfitting.
* Die nächsten Schritte werden geplant: Zugriff bei der Management Platform for Deep Learning (MLMP) soll angefragt werden. Das Training des Modells soll auf dieser Platform stattfinden. Zuvor soll aber noch der Schutz gegen Overfitting erweitert werden.

### Abgleich 5 (19.05.2025 09:00)

**Standort**: Microsoft Teams Online Meeting

**Teilnehmer**:

* Prof. Marcus Hudritsch (Betreuer)
* Pascal Cornu (Studierender)

**Meeting Protokoll:**

* Pascal Cornu präsentiert das trainierte Modell und die Verbesserungen am Modell gegen Overfitting. Zusätzlich wurde der Datensatz erweitert und aufgeteilt in Trainings-, Test- und Validationsdatensatz.
* Prof. Marcus Hudritsch ist grundsätzlich zufrieden mit dem Modell. Er weist darauf hin, die Bachelorthesis eine Woche früher abzugeben, damit er Zeit hat alles anzuschauen vor der Verteidigung. Weiter soll das Modell entweder ins SLProject eingebaut werden, oder mit Python und OpenCV eine grafische Anwendung erstellt werden.

### Abgleich 6 (02.06.2025 09:00)

**Standort**: Microsoft Teams Online Meeting

**Teilnehmer**:

* Prof. Marcus Hudritsch (Betreuer)
* Pascal Cornu (Studierender)

**Meeting Protokoll:**

* Pascal Cornu präsentiert die grafische Anwendung, die es erlaubt ein beliebig grosses Bild zu entrauschen. Zusätzlich wurde das Poster und das Book erstellt.
* Prof. Marcus Hudritsch gibt Feedback über das Book. Es muss noch erwähnt werden warum Path Tracing so wichtig ist. Zudem ist der Umfang eher knapp.
* Prof. Marcus Hudritsch weist auf einige Dinge hin, die in der Dokumentation nicht fehlen dürfen. Weiter soll ein Vergleich mit dem Optix Denoiser in Blender gemacht werden.

# Selbständigkeitserklärung

Ich bestätige mit meiner Unterschrift, dass ich meine vorliegende Bachelor-Thesis selbständig durchgeführt habe. Alle Informationsquellen (Fachliteratur, Besprechungen mit Fachleuten, usw.) und anderen Hilfsmittel, die wesentlich zu meiner Arbeit beigetragen haben, sind in meinem Arbeitsbericht im Anhang vollständig aufgeführt. Sämtliche Inhalte, die nicht von mir stammen, sind mit dem genauen Hinweis auf ihre Quelle gekennzeichnet.

Ich bestätige weiterhin, dass ich bei der Erstellung dieser Studienarbeit durchgehend steuernd gearbeitet habe und von einer KI erzeugte Inhalte nicht unreflektiert übernommen habe

Pascal Cornu

Name, Vorname ………………………………………………

07.06.2025

Datum ………………………………………………



Unterschrift ………………………………………………