FH Wedel

AI accelerated Data-Structures for efficient Raytracing

Bachelor-Thesis-Exposé

Jonas Sorgenfrei
[Datum]

Im Rahmen dieser Bachelor-Thesis soll geforscht werden in wie weit sich maschinelles Lernen in den Ray-Trace Prozess integrieren lässt und in wie weit sich die Ergebnisse von dem herkömmlichen Render-Ergebnis unterscheidet.
Insbesondere soll die Geschwindigkeit des Raytracing-Vorgangs in Verbindung mit der Genauigkeit des resultierenden synthetischen Bildes analysiert werden.
Im Gegensatz zu bisherigen Ansätzen der Kombination von maschinellem Learning in den Ray-Trace Prozess [1], soll in dieser Arbeit nicht die Anwendung des maschinellen Lernens auf das Ausgabe-Bild untersucht werden sondern dies direkt in dem Geometrie-Teil (Schnitt-Test) des Ray-Tracing-Prozesses Anwendung finden.
Insbesondere soll untersucht werden ob sich ein Netzwerk auf eine Szene trainieren lässt um im folgenden Schattenstrahlen schnell bestimmen zu können.
Dies kann beispielsweise bei animierten Kamerafahrten ein Vorteil ergeben. In diesem Prozess wird generell für ähnliche Punkte die Sichtbarkeit mit Lichtquellen oder diffus reflektierenden bzw. emittierenden Objekt/Flächen-Punkten überprüft. Im naiven Verfahren müsste ein entsprechend Strahl mit allen Geometrien im Raum überprüft werden mit entsprechend hoher Laufzeit. Dieser Vorgang kann durch Volume Hierarchien und Daten-Strukturen wie ein Kd-Tree verbessert werden [2], es verbleibt in der Regel jedoch trotzdem ein großer Teil von Schnitttest, daher befindet sich an dieser Stelle des Raytracing-Prozesses idR ein Bottle-Neck der ggf. durch maschinelles Lernen vermindert werden kann.
Im Hauptteil dieser Arbeit soll ein Netzwerk trainiert werden, das für bestimmte Schnitttests schneller Ergebnisse liefert. Insbesondere stehen im Focus Schattenstrahlen und evtl. Strahlen für das Path Tracing. Letzteres wird beim Monte-Carlo-Raytacing genutzt um mit Hilfe von diffusen Reflexionen von Oberflächen besser die globale Beleuchtung zu simulieren. Durch verschiedene Trainingsansätze soll das Netzwerk im Folgenden optimiert werden.
Hierzu plane ich folgende Ansätze zu evaluieren:

- ein Simplen Raytracing-Datensatz mit einer sehr großen Menge an zufällig vorberechneten Schnitt-Test
- einen vorberechneten Datensatz dessen Verteilung sich jedoch an den Lichtquellen orientiert um insbesondere für die Schattenstrahlen akkurate Ergebnisse liefert
- einen vorberechneten Datensatz dessen Verteilung sich an der Geometrie-Verteilung im Raum orientiert
- einen vorberechneten Datensatz der sich zum einen an den Lichtquellen orientiert aber auch an weiteren stark diffus-reflektierenden Szenenelementen, dazu müssen Material-Eigenschaften ausgewertet werden; dieser Ansatz soll insbesondere für das Monte-Carlo-Raytracing dienen

Für die Ausgaben des Netzwerks sind folgende Ansätze zu untersuchen:

- ein Wahrheitswert der zurückgibt ob es entsprechende Verdeckungen gibt oder nicht
- ein Wahrscheinlichkeitswert der Aufschluss über die Möglichkeit der Verdeckung gibt, so lässt sich die Genauigkeit des Ray-Tracers zur Laufzeit über Parameter einstellen, um in einer bestimmten Wahrscheinlichkeits-Range einen genaueren Test durchzuführen
- der Ansatz in Punkt 2 soll erweitert werden um eine Komponente die der den Wahrscheinlichkeitswert gewichtet je nach Präsens/Priorität der Ausgangsgeometrie in der gesamten Szene, sodass ggf. Hintergrund-Elemente/weit entfernte Elemente eine geringere Gewichtung haben als Geometrien die sich im Vordergrund befinden

In den Anfängen wird das Netzwerk auf 2D Geometrien ausgelegt sein um eine Schnittstelle für den besagten Test zu entwickeln. Sobald diese Schnittstelle Entwickelt ist, soll das Netzwerk auf 3D

Geometrien ausgelegt werden.
Es sollen jeweils zuerst eine simple Szene mit einfachen Geometrien (Triangels, Cube, Sphere, Plane) getestet werden und im späteren Verlauf sollen weitere komplexere Szenen genutzt werden um Aussagekräftige Evaluierungsergebnisse zu erhalten.
Insbesondere soll auf Randbereich (z.B. Schattenkanten & Kanten allgemein) besonders eingegangen werden, da diese typische Fehlerquellen/Problembereiche in der synthetischen Bildproduktion darstellen (Stichwort floating-point Fehler).
Nach erfolgreichen Versuchen im 3D-Raum soll versucht werden das entwickelte Netzwerk in die Pipeline des Physical Based Rendering Systems beschrieben von Pharr et all [2] zu integrieren.

Der im Buch „Physical Based Rendering: From Theory to Implementation“ (Pharr et. all) [6] beschrieben Raytracer soll als Basis für das Projekt dienen. Dieser wurde bereits als Grundlage für das open source Projekt LuxCoreRenderer [3] genutzt, bei dem es sich um ein Physical Based-Render handelt, der u.a. eine Integration für Blender bietet. Des Weiteren wurde die generelle Systemarchitektur für den von Pixar entwickelten RenderMan Renderer verwendet.
In ähnlicher komplexen Weise arbeiten weitere Namenhafte in der Industrie verwendete Render-Engins wie Arnold, V-Ray, Clarisse oder Octane.
Zusätzlich werden auf der Website zum Buch verschiedene komplexe Szenen zum Download angeboten mit denen die Effizienz & Genauigkeit des entwickelten Systems zu evaluieren ist.
Dieser Schritt soll dazu dienen die Anwendbarkeit des zu entwickelnden Netzwerkes für State-Of-The-Art Render-Systemen zu überprüfen.

Um schließlich eine Aussage über die Einsatzfähigkeit in der Filmproduktion zu machen, soll die Entwicklung mithilfe der „Moana Island Scene“ getestet werden. Dabei handelt es sich um eine reale Produktionsszene der Walt Disney Animation Studios (WDAS). Es ist eine Szene aus dem 2016 erschienenen Animationsfilm Moana. In dieser befinden sich Meshes mit mehr als 90 Millionen Quads und Triangles. Sowie ca. 28 Millionen Instanzen für Blätter, Gestein und Felsen. [4]
Zusätzlich zu dieser Szene soll ebenfalls darauf eingegangen werden, in wie weit sich das Netzt ggf. aus animierte Szenenobjekte eignen könnte bzw. in welcher Form das Netz für diese verändert werden müsste. Eine Implementierung letzteres ist zu diesem Zeitpunkt nicht vorgesehen, da dies eine komplett veränderte Herangehensweise erfordert.

Der Entwicklungs-Anteil an einer Demo-Software/Benchmark dieser Bachelor-Thesis soll ein Netzwerk sein, welches auf der Tensorflow-API basiert [6]. (Alternativ soll bei Schwierigkeiten auf PyTorch umgestiegen werden). Die Entwicklung der Schnittstellenwird in C++ stattfinden einige Elemente die sich auf Tensorflow beziehen werden u.U. in python erstellt.
Bei erfolgreichen und vielversprechenden Test soll basierend auf dem bereits genannten PBRT Render-System das entwickelte Netzwerk in dieses Integriert werden. Die Entwicklung der Thesis bezogenen-Erweiterungen wird in einem Git-Repository [5] auf github.com stattfinden.
Alle Evaluierungen sollen dabei so durchgeführt werden, dass die entsprechende Szene einmal mit dem entwickelten Netzt gerendert wird und einmal wie herkömmlich. Danach werden die Zeitlichen Unterschiede im Vergleich zu den Pixeln/Bildfehlern untersucht.

Verweise

[1] R. A. C. CHAKRAVARTY, S. K. ANTON , S. CHRISTOPH, S. MARCO , L. AARON, N. DEREK und A. TIMO, „Interactive Reconstruction of Monte Carlo Image Sequences using aRecurrent Denoising Autoencoder," [Online]. Available: https://research.nvidia.com/sites/default/files/publications/dnn_denoise_author.pdf. [Zugriff am April 2019].

[2] P. Matt, J. Wenzel und H. Greg, Physically Based Rendering: From Theory to Implementation - Third Edition, Morgan Kaufmann; 3. edition (21 Nov 2016), 2016.

[3] LuxCoreRender project, „LuxCoreRenderer," [Online]. Available: https://luxcorerender.org/. [Zugriff am April 2019].

[4] Walt Disney Animation Studios, „Moana Island Scene," Walt Disney Animation Studios, [Online]. Available: https://www.technology.disneyanimation.com/islandscene. [Zugriff am April 2019].

[5] J. Sorgenfrei, „GIT AI accelerated PBRT," April 2019. [Online]. Available: https://github.com/jonassorgenfrei/AIAPBRT-BachelorProject. [Zugriff am April 2019].

[6] Google Brain Team, „TensorFlow Framework," Google, [Online]. Available: https://www.tensorflow.org/. [Zugriff am April 2019].