FH Wedel

AI accelerated Data-Structures for efficient Raytracing

Bachelor-Thesis-Exposé

Jonas Sorgenfrei
[Datum]

In der Film- und Video-Produktion werden Renderengines genutzt, um die 3D-Szenen, die in einem 3D Editor/Compositor (z.B. Houdini, Maya, Cinema4d, Clarisse) erstellt wurden, als ein Pixelbild zu rendern/berechnen.
Zu Bekannten, derzeit in der Industrie genutzten Renderengines gehören u.a. Arnold, Octane, Indigio, V-Ray, RenderMan und Redshift [1].
In diesem Teil der Produktions-Pipeline wird das Problem der (physikalisch annähernd korrekten) Lichtberechnung einbezogen, welches je nach gewünschter Genauigkeit, die Berechnungsdauer des einzelnen Ausgabe-Bildes enorm erhöht. Dadurch entsteht insbesondere an dieser Stelle ein Haupt-Bottle-Neck in der kreativen Grafikgestaltung.
In der aktuellen Entwicklung wurden bereits viele Techniken entwickelt, die diesen Prozess durch effiziente Datenstrukturen, effiziente Ausnutzung der Hardware, durch Hardwarenahe Befehlsinstruktionen oder Nutzung der GPU-Rechenpower beschleunigen konnten.
Zum Beispiel werden die Szenenstrukturen angepasst, es wird (bis zu einem bestimmten Anteil) parallel gearbeitet (Multithreading, GPU-Rendering) und es werden effiziente Approximationen für Lichtmodelle verwendet.
Insbesondere in der aktuellen Entwicklung wird das Thema „künstlicher Intelligenz(AI)/Maschinen-Learning“ in viele Bereiche der Technik integriert, da diese viele Prozesse, insbesondere mit einer großen Menge an Input-Daten, beschleunigen kann.
Insbesondere auf der Bildebene gab es bereits sehr viele positive Ergebnisse in diesem Bereich.
Z.B. Objekt-Detektion, Szenen-Interpretation und Denoising.
Im Bereich des Renderings werden derzeit Ansätze des Maschinen Learning genutzt, zur Filterung von partiellen Render-Ergebnissen durchzuführen, um so zwischenpixel zu berechnen [2].

In der Echtzeitgrafik kam mit der Einführung der neuen Nvidia Grafikkarten Serie RTX die Turing-Technik in den Spiele-Consumer-Markt und es entstand die Möglichkeit Raytracing auch in der Echtzeitgrafik zu nutzen. [3]
Mithilfe von Compute Shadern ist es bereits möglich in Grafik-APIs wie z.B. OpenGL, Vulkan und DirectX Raytracing zu nutzen. [4]

Auch die bereits genannten Renderer nutzen inzwischen die Power der parallel GPU Berechnung für den Render-Prozess. Zum Beispiel basiert der Renderer Redshift komplett auf GPU Berechnungen und im März 2019 hat der Solide Angle eine Beta-Version eines GPU basierten Arnold Renderers (in Autodesk Maya integriert) herausgebracht. [5]

Wie aus diesen genannten Beispielen ersichtlich wird, ist die Entwicklung von effizienten Renderern und Raytracern (/Pathtracern) in der aktuellen Forschung des CG-Bereiches im Arbeit, um diese zu Beschleunigen und/oder die Ergebnisse zu verbessern.

In Renderern für die Filmindustrie wird generell ein Ansatz genutzt der sich PBR nennt. Hierbei handelt es sich um Physical Based Rendering. Es werden Algorithmen und Lichtapproximationsmodell verwendet die versuchen das Ergebnis möglichst nahe an der realen physikalischen Welt darzustellen. Das große Ziel ist dabei ein Bild zu generieren, welches sich nicht von einer Photographie unterscheiden lässt. Entsprechend müssen die verwendeten Algorithmen und Modelle (Lichtmodelle ..) so gewählt sein, dass diese der Realität möglichst nahe kommen.
Simplifiziert kann dies in folgende Kategorien eingeteilt werden:

- ➔ Kameras und Optik
- ➔ Lichtquellen und Indirekte Beleuchtung
- ➔ Strahl-Objekt Schnittstellen sowie Strahlausbreitung (Energie)
- ➔ Materialien und Oberflächen-Streuung

Im Rahmen dieser Bachelorthesis soll auf verschiedene (möglichst AI-basierte) Techniken eingegangen werden, mit denen sich der Render-Prozess effizienter gestalten lässt. Insbesondere soll auf die Möglichkeiten der Szenen-Datenstrukturierung und mögliche Optimierungen durch AI/Maschinen-learning eingegangen werden.
Als konkretes Beispiel kann sich folgender Ansatz eignen:
Beim Ray-Tracing werden Schnitttests mit Strahlen und der Szenengeometrie durchgeführt.
In einem naiven brute-force Ansatz wird jedes Objekt für jeden Strahl getestet ob es ein Schnittpunkt gibt. Dieser Ansatz ist jedoch sehr ineffizient (insbesondere bei sehr Komplexen Geometrien) und bei vielen komplexen Objekten in einer komplexen Szene (was generell der in der Filmproduktion der Fall ist) würde dieser Ansatz zu enormen Renderzeiten führen.
Mit Hilfe von Datenstrukturen wie Bounding-Volumen und Hierarchien kann dieser Prozess beschleunigt werden, sodass sich die Laufzeit auf eine logarithmischen Zeit minimieren lässt, diese Strukturen sind in Kapitel 3 und 4 des Buches Physical Based Rendering von Pharr et. all beschrieben [6].
Konkrete bezogen auf die Bounding-Volumen gilt, dass diese am effizientesten sind, wenn bei einer (starken) Simplifizierung der geometrischen Struktur, diese dazu führen, dass ein Großteil der Schnitttests falsifiziert werden, sodass die komplexe Geometrie nicht genauer geprüft werden muss. Da beim Ray-Tracing Reflektion und Refraktionen beachtet werden, kann die Szene nicht einfach gecullt werden wie es bei einem Rasterizer der Fall ist.
Der Forschungsansätze wäre, die Szenen-Hierarchie sowie die Bounding-Volumen der Geometrien mithilfe von Maschinen-learning so zu wählen, das diese effizient (automatisch) anhand der vorgegebenen Szene gewählt und konstruiert werden.
Ein Beispiel dafür, wäre die Geometrie eines Baumes. Für diesen wäre eine Kugel generell nicht das beste Bounding-Volume. In einer Szene in der dieser sich jedoch weit entfernt von der restlichen Geometrie und außerhalb des Kamera-Frustums befindet ist die Wahrscheinlichkeit, dass dieser von einem Strahl getroffen wird eher unwahrscheinlich, sodass eine hochauflösende konvexe Hülle um dieses Objekt nicht erforderlich ist, da der Raumwinkel ausgehend von der restlichen Geometrie zu dem Objekt hin je nach Abstand abnimmt. Dadurch würde auch die Schnitt-Wahrscheinlichkeit eines ausgesendeten Strahles von der Sphere um die Zentrumsgeometrie der Szene ja nach Abstand des besagten Baumes bzw. dessen Bounding-Volumens abnehmen und es kann eine gröbere Approximation gewählt werden (wie z.B. die besagte Kugel).
Weitere Faktoren wie Anzahl der Reflektionsflächen (bei einfachem Raytracing; bei Pathtracing sind dies ja generell alle), Reflektionswinkel-Anteil-Funktion um die sichtbaren Geometrien, Lichtenergien, Kamera-Einstellungen, Verdeckung etc. sind weitere Parameter die mit in diese Entscheidung eine Rolle spielen.

Ein AI-Netzwerk kann trainiert werden anhand dieser Parameter Bounding-Volumen sowie Szenen-Unterteilung/Hierarchien zu erzeugen, die den finalen Ray-Tracing Schritt vereinfachen.
Ein AI-Netzwerk hat den Vorteil, dass die große Anzahl der Parameter nicht serielle verarbeitet werden muss für diese Struktur-Erstellung.
In einer Erweiterung der beschriebenen Idee könnten ebenfalls Parameter wie Objekt-Deformationen, Simulations- und Animations-Eigenschaften sowie generell die Zeitkomponente beachtet werden.

Der im Buch Physical Based Rendering (Pharr et. all) [6] beschrieben Raytracer soll als Basis für das Projekt dienen. Dieser diente bereits als Basis für das open source Projekt LuxRender, bei dem es sich um ein PB-Render handelt, der u.a. eine Integration für Blender bietet. Des Weiteren wurde die generelle Systemarchitektur für den von Pixar entwickelten RenderMan Renderer verwendet.

Um einen Vergleich zu der Technik in Bezug auf Laufzeit, Effizienz und generell Praktikabilität zu bekommen, sollen mehrere Verschiedene Szene den normalen CPU PBRT Render mit Multithreading durchlaufen sowie die entsprechende AI-basierte Konfiguration.

Sollte sich diese als effektiv herausstellen, soll ebenfalls die Kompatibilität auf GPU-Ebene getestet und verglichen werden. Für diesen Zweck soll die CPU Variante des PBRT auf eine GPU Variante mithilfe des NVIDIA Cuda SDKs umgeschrieben werden.

Um schließlich eine Aussage über die Einsatzfähigkeit in der Filmproduktion zu machen, soll schließlich die Entwicklung mithilfe der „Moana Island Scene“ getest werden. Dabei handelt es sich um eine reale Produktionsszene von Walt Disney Animation Studios (WDAS) von dem 2016 erschienenen Animationsfilm Moana. In dieser befinden sich Meshes mit mehr als 90 Millionen Quads und Triangles. Sowie ca. 28 Millionen Instanzen für Blatter, Gestein und Felsen. [7]

Als letzten Punkt soll dann noch auf die mögliche Kompatibilität auf Echtzeit Renderer basierend auf der von NVIDIA genutzten Turing-Architektur eingegangen werden um dies ggf. mit DirectX oder Vulkan fuer Echtzeit Raytracing zu nutzen.

Der Entwicklungs-Anteil an einer Demo-Software/Benchmark dieser Bachelor-Thesis soll basierend auf dem bereits genannten PBRT Renderer sein und die Entwicklung der Thesis bezogenen-Erweiterungen wird in einem Git-Repository [8] auf github.com stattfinden.

Verweise

[1] M. v. Übel, „All3DP 25 Best 3D Rendering Software Tools in 2019,“ April 2019. [Online]. Available: https://all3dp.com/1/best-3d-rendering-software/.

[2] 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].

[3] Nvidia, „Nvidia Geforce RTX,“ [Online]. Available: https://www.nvidia.com/de-de/geforce/20-series/rtx/. [Zugriff am April 2019].

[4] NVIDIA, „Introduction to DirectX RayTracing,“ 2018. [Online]. Available: http://intro-to-dxr.cwyman.org/. [Zugriff am April 2019].

[5] Solide Angle Arnold, „Introducing Arnold 5.3 with Arnold GPU in public beta,“ Marz 2019. [Online]. Available: https://www.arnoldrenderer.com/news/press-release-arnold-5-3-gpu/. [Zugriff am April 2019].

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

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

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