Fast Motion Vector Based Denoiser Implementation

Fachgebiet: Human Interface Technologie
Betreuer: Prof. Roger Cattin
Experte: Dr. Peter Maloca

In den meisten Computern sind heutzutage Multicore-Prozessoren und leistungsstarke Grafikkarten vorhanden. Diese Geräte bieten eine vielfach nicht ausgenutzte Leistung. Mit Hilfe von Parallelisierung und entsprechender Programmierung in z. B. OpenCL lässt sich eine enorme Leistungsverbesserung bemerkbar machen. In dieser Arbeit wurde der «Motion Vector Based Denoiser»-Algorithmus mit Hilfe dieser Technik implementiert und zusätzlich optimiert. Daraus resultierte ein erstaunlich schnelles Programm, welches produktiv eingesetzt werden könnte.

Ausgangslage

Im Rahmen der Masterarbeit von Cyrill Gyger vor einem Jahr wurde ein Algorithmus zum Entrauschen von Optischen Kohärenz Tomographie (OCT) Bildern entwickelt. Beim Implementieren wurde der Fokus vor allem auf die Funktionalität und nicht auf die Performance gelegt. Dadurch entstand ein Algorithmus, welcher extrem gute Ergebnisse in Bezug auf das Entrauschen gab, jedoch mit mehreren Stunden Laufzeit für das Bild eines Auges in der Praxis nicht effizient eingesetzt werden konnte.

Das Ziel dieser Arbeit war somit den Algorithmus so umzuschreiben und zu optimieren, dass er in der Praxis vernünftig eingesetzt werden kann. Dazu wird der Algorithmus so angepasst, dass von der Grafikkarte und modernen Multicore-CPUs ausgiebig Gebrauch gemacht werden kann.

Algorithmus

Bei den OCT-Bildern handelt es sich um Volumen-Bilder. Das bedeutet, dass die Bilder nicht 2 dimensional, sondern 3 dimensional sind. Beim Entrauschen wird nun nicht nur eine Schicht des Bilder einbezogen, sondern auch mehrere Nachbarschichten. Mithilfe einses Block-Matching-Verfahrens wird ein Vektorpfad gebildet, welcher die strukturalen Elemente des Bildes verfolgt. Anhand dieser Vektoren wird schliesslich das Bild entrauscht.
Die aufwendigste Berechnung ist das Erstellen der Vektoren mit Hilfe des Block-Matching.

Optimierungen

Alle Performance relevanten Teile des Algorithmus wurden in OpenCL neu geschrieben. Bei OpenCL handelt es sich um eine Programmiersprache, die benutzt wird um parallele Berechnungen auf der Grafikkarte oder der CPU laufen zu lassen. Der OpenCL-Code wird erst zur Laufzeit des Programmes kompiliert. Dadurch wird das Programm optimal auf das aktuell verwendete Gerät optimiert. Zusätzlich wird auf der CPU von verschiedene Erweiterungen, die das parallele Rechnen zusätzlich beschleunigen (z. B. SSE, AVX), Gebrauch gemacht.

Des Weiteren wurde der Block-Matching-Algorithmus verschnellert, indem er Lookup-Tables benutzt um Zwischenresultate zu speichern. Dadurch wurde die Komplexität der Berechnung stark verkleinert.

Resultate

Die Resultate der Optimierung übertraf die Erwartungen bei weitem. Wie in der Tabelle ersichtlich, konnte bereits für die Ausführung auf der CPU der Faktor 90x heraus geholt werden. Dadurch läuft der Algorithmus in ca. 2.5 Minuten anstatt 3.5 Stunden durch, liefert dennoch das genau gleiche Resultat.

Noch eindrücklicher ist die Performance der Grafikkarte. Hier werden nur noch 22 Sekunden benötigt, was einem Faktor von 570x entspricht.

Fazit

Durch das Optimieren eines Algorithmus auf die Grafikkarte unter Verwendung von OpenCL kann eine extreme Geschwindigkeitssteigerung realisiert werden, wenn sich der Algorithmus dazu eignet.

Es stellte sich heraus, dass OpenCL zudem recht praktisch und angenehm zum Implementieren ist.

Algorithmus	Gerät	Ausführungszeit
Original	2x Intel Xeon X5660 (CPU)	3.5 h
OpenCL	2x Intel Xeon X5660 (CPU)	139 s
	AMD Radeon HD7970 (GPU)	22 s

Vergleich des Orignial-Algorithmus mit dem neuen OpenCL-Algorithmus

Adrian Pauli
+41 79 765 06 33