Fakulta informačních technologií VUT v Brně
GMU (Grafické a multimediální procesory)
Tomáš Pelka (xpelka01)
31. prosince 2017

Rychlý bilaterální filtr na GPU

Zadání

- Implementujte standardní metodu výpočtu bilaterálního filtru na GPU.
- Implementujte optimalizace z článku https://people.csail.mit.edu/fredo/PUBLI/Siggraph2002/DurandBilateral.pdf
- Srovnejte rychlost a přesnost standardní a optimalizované varianty výpočtu.

Popis algoritmu

Bilaterální filtr je nelineární filtr, který dokáže potlačit šum při zachování hran.

Pro filtrování jsou důležité 2 základní parametry – blízkost bodů (v souřadném systému) a podobnost barev. Čím jsou body k sobě blíž (pro výpočet se používá Euklidovská vzdálenost – Pythagorova věta), tím větší mají váhu. Čím jsou si body barevně podobnější (opět Euklidovská vzdálenost, ale ve 3D), tím větší mají váhu.

Implementace

Volba knihoven

Pro implementaci jsem zvolil knihovny OpenCL pro paralelizaci na GPU. Dále jsem využil knihovnu OpenCV pro práci s obrázky – vstup, výstup, převod barevného prostoru.

Program byl vytvořen v jazyce C/C++ ve vývojovém prostředí Microsoft Visual Studio. Visual Studio disponuje vlastními nástroji pro sestavení programu, proto jsem nepoužil CMake.

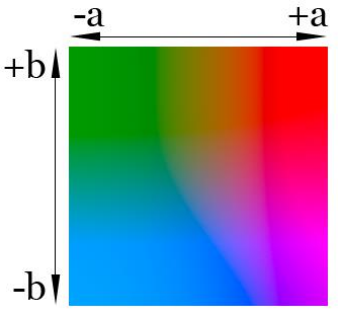
Knihovny OpenCL a OpenCV byly dynamicky linkovány a je proto nutná jejich přítomnost v systému.

Barevný model

Pro určení barevné podobnosti 2 bodů je potřeba se zaměřit na barevný prostor, který tyto body reprezentuje.

Prostor RGB není zcela vhodný, protože každou z barevných složek vnímáme různě. Vhodný je prostor CIE-Lab, kde se pracuje se 3 parametry: světelnost (L), a 2 barevné osy (a, b).

Při použití OpenCV je běžná reprezentace RGB (BGR). Tato knihovna zároveň umožňuje přímý převod do CIE-Lab prostoru.



Obrázek 1: Lab osy a, b

Datové typy OpenCL vs OpenCV

Problémem však byly datové typy, které reprezentují obrázek v paměti. OpenCV má obraz uložený ve formě matice (v poli) volitelných datových typů. Kernel OpenCL však dokáže přijímat jen některé datové typy. Pro reprezentaci barev CIE-Lab pro OpenCL kernel jsem použil float3.

Pro usnadnění převodů mezi různými datovými typy těchto knihoven, jsem vytvořil třídu MyMat.

Fakulta informačních technologií VUT v Brně
GMU (Grafické a multimediální procesory)
Tomáš Pelka (xpelka01)
31. prosince 2017

Bilaterální filtr – standardní

Kernel standardního bilaterálního filtru je v souboru *bilateralFilter_basic.cl*. Zdrojový kód je okomentován, takže zde zmíním jen některé informace.

Rozhodl jsem se algoritmus implementovat tak, že výsledný obrázek bude menší o okrajové (tzv. halo zóny). Velikost okrajů je dána prostorovým parametrem filtru (radius).

Samotné funkce prostorové a barevné blízkosti jsou realizovány Gaussovými funkcemi. Parametry filtru ovlivňují právě tyto funkce.

Ladění

Pro potřeby ladění byl vytvořen testovací kernel v souboru *bilateralFilter_test.cl*. Tento kernel neprovádí filtrování, ale pouze okopíruje vstupní obrázek na výstup, vynechá halo zóny.

Spuštění programu

Program má 4 povinné parametry:

- Cesta ke vstupnímu obrazu.
- Parametr filtru – prostorový (radius).
- Parametr filtru – podobnost barev (barevná blízkost).
- Cesta k výstupnímu souboru.

Dále je možné doplnit 5. parametr *„-b“*. Tímto parametrem se spustí režim *benchmark*, který:

- Do názvu souboru (na konec) přidá dobu běhu ocl_kernel.
- Skončí ihned po dokončení výpočtu (bez tohoto režimu se čeká na stisk klávesy *Enter*).

Testování, benchmark

Hardware

Testování probíhalo na notebooku se 2 grafickými kartami. Pro přepnutí se na výkonnou kartu NVIDIA jsem v kódu připravil vynucení této platformy (nyní zakomentované).

OpenCL správně detekovalo následující hardware:

- CPU: Intel CPU Core i5-3210M @2,5 GHz,
- GPU: Intel HD Graphics 4000,
- GPU: NVIDIA GTX 660M.

Benchmark

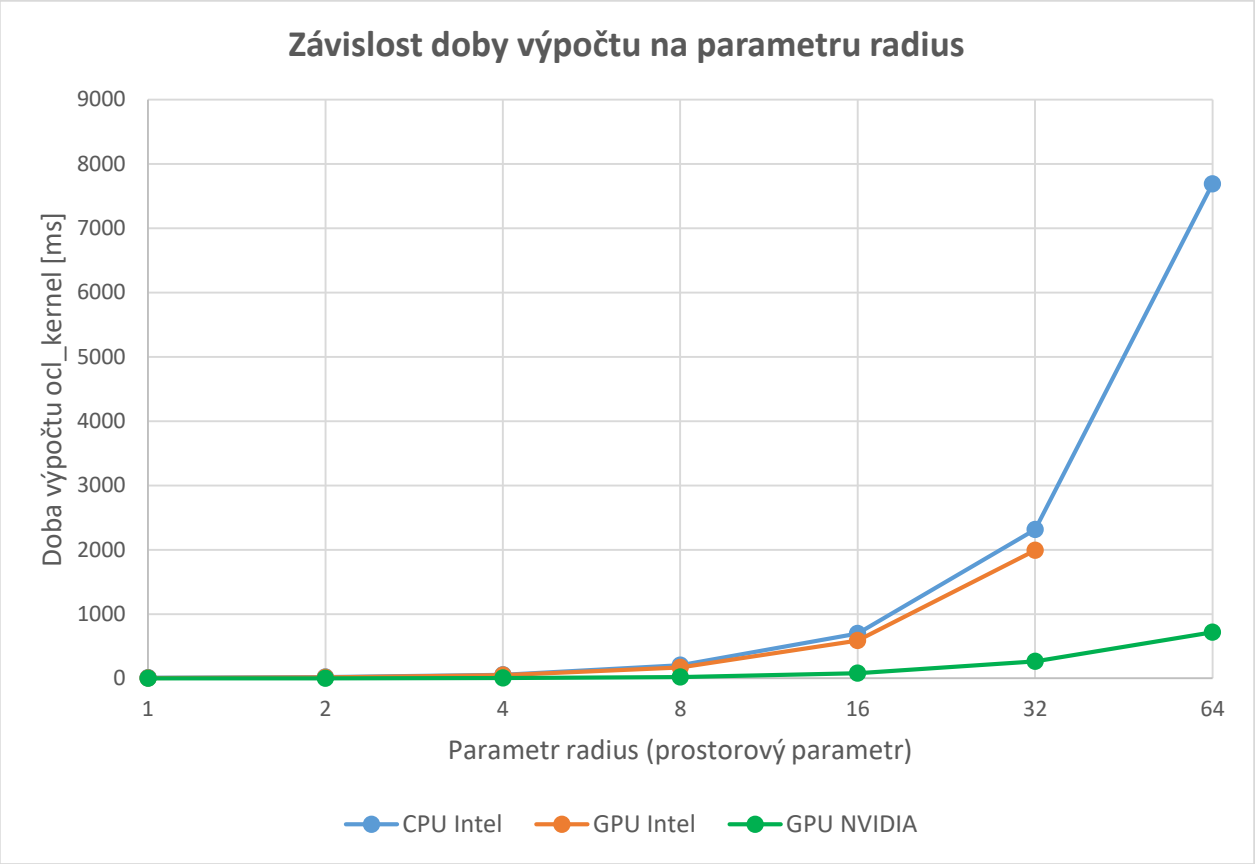
Pro výkonnostní testování jsem použil barevnou fotku *Lenna* v rozlišení 512x512 bodů.

U bilaterálního filtru byly kombinatoricky měněny oba vstupní parametry. Pro spuštění benchmarku je nachystána dávka v *„Sources/benchmark.bat“.* Výsledná zjištění jsou následující:

- Parametr „podobnost barev“ nemá na výkon prakticky žádný vliv.
- Parametr „radius“ má na výkon významný vliv.
- Při dvojnásobném zvětšení radiusu dojde cca ke trojnásobnému prodloužení doby výpočtu.

- GPU Intel dokázalo počítat do radiusu 32 (včetně), zatímco ostatní hardware zvládl i 64.

Naměřené hodnoty jsou přiloženy v souboru Benchmark.xlsx. Zde prezentuji na grafu závislost doby výpočtu na parametru radius:



Graf 1: Závislost doby výpočtu na parametru radius

Výsledné obrázky z benchmarku je možné stáhnout z mého webového uložiště (cca 60 MB): http://www.stud.fit.vutbr.cz/~xpelka01/gmu_benchmark.zip.

Porovnání výsledných obrazů

Porovnal jsem taky výsledné obrázky mezi jednotlivým hardware. Vyfiltrované obrázky jsou prakticky shodné. Minimální (neznatelné) odchylky vznikly pravděpodobně při operacích s plovoucí řádovou čárkou.

Závěr

Na závěr musím poznamenat, že týmový kolega (Karol Troška) neměl na řešení projektu velký přínos. Konkrétně se podílel na části kostry programu (main.cpp) a na neúspěšných pokusech s CUDou. Na mně (Tomáš Pelka) zůstaly všechny ostatní úkoly, přičemž jsem nakonec implementoval projekt v OpenCL kvůli znalostem z počítačových cvičení.

Fakulta informačních technologií VUT v Brně
GMU (Grafické a multimediální procesory)
Tomáš Pelka (xpelka01)
31. prosince 2017

Podařilo se mi implementovat standardní metodu bilaterálního filtrování na GPU. Bohužel, nezvládl jsem implementovat optimalizovanou metodu dle článku. Přesto jsem provedl důkladný benchmark alespoň základní metody.

Dovoluji si Vám navrhnout následující rozdělení bodů: 25 % Karol Troška, 75 % Tomáš Pelka. Svou aktivitu a podíl na řešení projektu mohu doložit verzovacím systémem Git.

Děkuji za pochopení a omlouvám se za komplikaci, Tomáš Pelka.

Zdroje informací

https://people.csail.mit.edu/fredo/PUBLI/Siggraph2002/DurandBilateral.pdf

https://en.wikipedia.org/wiki/Bilateral_filter

http://xidexia.github.io/Bilateral-Filtering/

https://github.com/OpenCL/

http://people.csail.mit.edu/sparis/bf_course/slides/03_definition_bf.pdf

https://stackoverflow.com/questions/5000665/bilateral-filtering-with-color

https://en.wikipedia.org/wiki/Lab_color_space

https://cs.wikipedia.org/wiki/Lab