Radosław MANTIUK, Anna TOMASZEWSKA, Dawid PAJĄK
POLITECHNIKA SZCZECIŃSKA, INSTYTUT GRAFIKI KOMPUTEROWEJ I SYSTEMÓW MULTIMEDIALNYCH

Wykorzystanie procesorów graficznych do szybkiego przetwarzania obrazów HDR

Dr inż. Radosław MANTIUK

Ukończył studia na Wydziale Techniki Morskiej Politechniki Szczecińskiej w specjalności systemy oprogramowania w 1994 r. oraz w specjalności pomiary i sterowanie w oceanotechnice w 1995 r. W 1999 r., obronił pracę doktorską na Wydziale Informatyki. Obecnie pracuje jako adiunkt w Instytucie Grafiki Komputerowej i Systemów Multimedialnych. Zainteresowania naukowe to przetwarzanie obrazów HDR, analiza percepcyjna oraz systemy graficzne czasu rzeczywistego.

e-mail: rmantiuk@wi.ps.pl

Dr inż. Anna TOMASZEWSKA

Ukończyła studia na Wydziale Informatyki Politechniki Szczecińskiej w 2000 r. W 2003 r. obroniła pracę doktorską na tym samym wydziale. Obecnie pracuje jako adiunkt w Zakładzie Grafiki Komputerowej Instytutu Grafiki Komputerowej i Systemów Multimedialnych. Jej zainteresowania naukowe to przetwarzanie obrazow HDR, analiza percepcyjna oraz systemy graficzne czasu rzeczywistego.

e-mail: atomaszewska@wi.ps.pl

Dawid PAJĄK

Obecnie jest studentem 5 roku Wydziału Informatyki Politechniki Szczecińskiej na specjalizacji grafika komputerowa. Jego zainteresowania naukowe to systemy graficzne czasu rzeczywistego oraz szeroko pojęta optymalizacja.

e-mail: dpajak@wi.ps.pl

Streszczenie

Zdjęcia HDR umożliwiają rejestrację jasności sceny, w zakresie widzianym przez człowieka. W celu poprawnego odwzorowania luminancji, obrazy HDR zapisywane są za pomocą 4-bajtowych liczb zmiennoprzecinkowych. Pomimo dużej wydajności obecne procesory CPU nie są w stanie przetworzyć w sposób interaktywny tak dużej liczby danych. W artykule zaprezentowano architekturę oraz implementację autorskiej biblioteki do przetwarzania i analizy obrazów HDR, wykorzystując procesor graficzny w charakterze jednostki wspomagającej obliczenia. Poprawność działania biblioteki przetestowano na przykładzie algorytmu kompresji tonów obrazów HDR.

Słowa kluczowe: obrazy HDR, GPU, przetwarzanie obrazów, analiza obrazów, grafika komputerowa.

Fast Processing of HDR Images Based on GPU acceleration

Abstract

High Dynamic Range imaging technology allows to capture a full range of luminance visible by a human. To achieve accurate reproduction of a luminance, HDR images are stored based on 4-bytes floating-point representation of pixel. Despite growing efficiency current CPU processors are not able to interactively process so huge amount of data. In the paper we present architecture and implementation of a novel library for processing and analysis of HDR images. The architecture of the HDR library is based on a programmable GPU (Graphics Processor Unit) hardware acceleration. We tested accuracy and efficiently of the library for the implementation of HDR tone compression algorithm.

Keywords: HDR images, GPU, image processing, image analysis, computer graphics.

1. Wprowadzenie

Obrazy HDR (*ang. High Dynamic Range*) rejestrują zakres jasności światła, porównywalny do zakresu widzianego przez człowieka. Dzięki temu na zdjęciach HDR widoczne są detale zarówno w bardzo jasnych, jak i bardzo ciemnych częściach fotografowanej sceny. Cechą takich zdjęć jest brak efektu prześwietlenia czy niedoświetlenia tych obszarów sceny, które widoczne są dla fotografa.

Zarejestrowanie informacji o pełnym zakresie luminancji wymaga zastosowania zmiennoprzecinkowej reprezentacji koloru piksela. W przeciwieństwie do standardowych obrazów LDR (*ang. Low Dynamic Range*), użycie trzech liczb 8-bitowych odpowiadających kanałom R, G i B jest niewystarczające. W celu poprawnego odwzorowania fotografowanego zakresu luminancji wartości kanałów koloru należy zapisać za pomocą 4-bajtowych liczb zmiennoprzecinkowych. Zamiana wartości całkowitych na zmiennoprzecinkowe zwiększa liczbę danych do przetworzenia. Jednocześnie zmienia się charakter obliczeń, np. konieczne jest wykorzystanie jednostek FPU. Pomimo dużej wydajności współczesne procesory CPU nie są w stanie przetwarzać w sposób interaktywny obrazów HDR.

W artykule prezentujemy autorską bibliotekę wspomagającą szybkie przetwarzanie i analizę obrazów HDR. Jej działanie oparte zostało na wykorzystaniu programowalnych procesorów graficznych GPU, dzięki czemu znacząco przyśpieszono obliczenia. W bibliotece uwzględniono szereg mechanizmów optymalizacyjnych, które dostosowują potok przetwarzania do możliwości GPU. Najważniejsze z nich to wykorzystanie operacji wektorowych typu SIMD procesora GPU, kolejkowanie operacji, zrównoleglenie obliczeń oraz adaptacyjne dzielenie zadań pomiędzy jednostki procesora GPU.

Ideę wektorowego przetwarzania danych oraz przegląd artykułów dotyczących przetwarzania obrazów HDR w oparciu o GPU opisano szczegółowo w rozdziale drugim. W kolejnym rozdziale zawarto opis architektury oraz implementację opracowanej przez autorów biblioteki. Wyniki badań zestawiono w rozdziale czwartym, przedstawiając w nim przykładowe wykorzystanie biblioteki do kompresji tonów obrazów HDR. Rozdział piąty zawiera podsumowanie oraz wskazanie kierunków dalszych prac.

2. Wektorowe przetwarzanie danych w GPU

Dostępne obecnie na rynku układy graficzne potrafią nie tylko wspomagać syntezę obrazów, ale również przetwarzać ogromne ilości danych zmiennoprzecinkowych. Najnowsze procesory graficzne [4] zawierają nawet 128 jednostek FPU (SISD), których moc przydzielana jest jednostkom renderującym (*ang. shader unit*) w sposób dynamiczny. Każda jednostka FPU wykonuje dokładnie ten sam program (*ang. fragment shader*) na odpowiadającym jej pikselu obrazu. Stwarza to możliwość uzyskania wydajności rzędu 520 GFLOPS na średniej klasy kartach graficznych.

Procesory GPU, oryginalnie przeznaczone do wspomagania syntezy grafiki trójwymiarowej, są obecnie coraz powszechniej stosowane do przyśpieszania obliczeń również w innych dziedzinach nauki [3]. Szczególną rolę zaczynają odgrywać m.in. w algorytmach analizy i przetwarzania obrazów LDR [1]. Niestety nie opracowano do tej pory uniwersalnej biblioteki wspomagającej sprzętowo przetwarzanie obrazów HDR, dla których wykorzystywane algorytmy różnią się znacząco od tych dedykowanych obrazom LDR. Dotychczasowe prace, prowadzone w celu przy-

śpieszania przetwarzania oraz analizy obrazów HDR z wykorzystaniem GPU, dotyczyły wyłącznie konkretnych algorytmów, takich jak np. kompresja tonów [2]

Uproszczony model współpracy pomiędzy GPU i CPU podczas przetwarzania danych wektorowych zawiera: (a) przesyłanie tablicy z danymi z CPU do GPU za pomocą zmiennoprzecinkowej tekstury RGBA (analogia do 128-bitowego wektora liczb zmiennoprzecinkowych z jednostki SSE CPU), (b) załadowanie do GPU, za pomocą CPU, tzw. shader'a, czyli specjalnego programu przetwarzającego dla każdego piksela odpowiednie teksele z tekstury wejściowej oraz zapisującego wynik obliczeń do tekstury wyjściowej. Operacje wykonywane za pomocą shader'a są operacjami wektorowymi, (c) wykonanie obliczeń na GPU, (d) zgranie za pomocą GPU tekstury wyjściowej, wraz z obliczonymi wartościami, z powrotem do pamięci RAM komputera.

Do zapisu fragment shader'a wykorzystano język wyższego poziomu GLSL (*ang. GL Shading Language*), będącego rozwinięciem języka C i przystosowanego do kompilacji na język maszynowy GPU. Pomimo niewątpliwych zalet implementacja algorytmów na GPU związana jest z pewnymi ograniczeniami:

- zapis do bufora, wykorzystywanego jako tekstura wejściowa, jest niedozwolony (ograniczenia kontrolera pamięci GPU),
- brak możliwości akumulacji wyników (zawartość rejestrów po wykonaniu programu dla piksela jest unieważniana),
- stosunkowo duży koszt transferu danych z i do GPU,
- użycie skoków warunkowych w programie shader'a obniża wydajność obliczeń,
- wykorzystanie pełnego potencjału obliczeniowego GPU, wymaga realizacji odpowiednio dużej ilości obliczeń zmiennoprzecinkowych przypadających na każde słowo odczytane z pamięci lokalnej GPU.

Z ostatniego ograniczenia wynika, że największe przyspieszenia na GPU można otrzymać dopiero dla odpowiednio długiego i skompilowanego shader'a. Zależność tę wykorzystano w module Queue prezentowanej biblioteki, gdzie poszczególne operacje są kolejkowane a następnie realizowane za pomocą jednego, złożonego shader'a. Zastosowane podejście intensyfikuje odpowiednio moc GPU.

Pomimo ograniczeń związanych z programowaniem na GPU, zarówno przesyłanie danych do i z GPU, jak i wykonywanie obliczeń wektorowych w GPU jest wielokrotnie szybsze od obliczeń realizowanych standardowo za pomocą CPU. Warto podkreślić, że przyspieszenie obliczeń, uzyskiwane jest nie tylko poprzez realizację operacji w sposób wektorowy na pojedynczej jednostce wykonawczej, ale również poprzez rozbicie obliczeń na wiele jednostek renderujących (najnowsze GPU zawierają nawet do 32 takich jednostek).

3. Architektura biblioteki wspomagającej przetwarzanie obrazów HDR

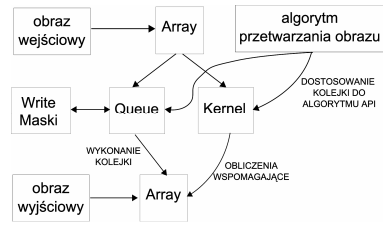
W poniższym rozdziale opisano architekturę oraz implementację biblioteki przyśpieszającej przetwarzanie i analizę obrazów HDR. Przedstawione zostały funkcje biblioteki oraz wyjaśniono mechanizm kolejkowania operacji.

3.1. Funkcje biblioteki

Obraz HDR można utożsamić z dwuwymiarową macierzą danych wektorowych. Każda dana wektorowa (lub inaczej piksel obrazu HDR) definiowana jest za pomocą trzech liczb zmiennoprzecinkowych, np. R, G i B, oznaczających wartości składowe koloru piksela. Zadaniem prezentowanej biblioteki jest przyśpieszanie przetwarzania macierzy (obrazów HDR), z czym związane jest wykonywanie szeregu operacji:

- prostych działań matematycznych (dodawanie, odejmowanie, mnożenie, dzielenie),
- obliczania wartości wybranych funkcji matematycznych (potęgowanie, logarytmowanie itp.),
- operacji akumulujących dane (suma elementów macierzy, obliczanie wartości maksymalnej i minimalnej, obliczanie splotu, powiększanie i pomniejszanie macierzy poprzez zastosowanie różnych filtrów, itp.),
- operacji warunkowych dla elementów macierzy.

Najważniejszą cechą biblioteki jest bardzo szybkie wykonywanie operacji przy zachowaniu poprawności i precyzji uzyskiwanych wyników. Głównym mechanizmem przyśpieszającym jest kolejkowanie operacji, mające na celu jak najbardziej efektywne wykorzystanie możliwości obliczeniowych procesorów GPU. Nie bez znaczenia jest również właściwe dostosowanie biblioteki konkretnie do potrzeb przetwarzania i analizy obrazów HDR, bazujące na czytelnym i efektywnym API (*ang. Application Programmer Interface*).



Rys. 1. Architektura biblioteki do analizy i przetwarzania obrazów HDR
Fig. 1. Architecture of library for processing and analysis of HDR images

3.2. Architektura biblioteki

Architektura opracowanej biblioteki zawiera dwie struktury danych: tablicę *Array* i strukturę *WriteMask* oraz dwa moduły funkcjonalne: *Quene* oraz moduł *Kernel* (rysunek 1). Dane wejściowe (obraz HDR) przechowywane są w strukturze *Array*. Kolejne etapy algorytmu przetwarzania obrazów zapisywane są w postaci sekwencji operacji (kolejki) i przekazywane do modułu *Queue.* Moduł ten odpowiedzialny jest za ich efektywne pogrupowanie oraz wykonanie (patrz rozdział 3.3). Otrzymany wynik przetwarzania umieszczany jest w wyjściowej strukturze *Array*.

Struktura *WriteMask* jest macierzą bitową wykorzystywaną w operacjach warunkowych. Jej wymiary pokrywają się rozmiarem wyjściowej struktury *Array*. W zależności od wartości ustawionych w tablicy *WriteMask*, możliwe jest zablokowanie wykonywania operacji dla wybranej danej ze struktury wejściowej (dla danego piksela). Dodatkowo zapis do struktury wyjściowej powoduje zmianę stanu odpowiedniego elementu struktury *WriteMask*.

Drugi moduł, *Kernel,* odpowiedzialny jest za wykonywanie operacji akumulacyjnych. Umożliwia również bezpośrednie wykonywanie poszczególnych operacji bez konieczności ich wcześniejszego kolejkowania. *Kernel* dodatkowo pełni rolę fabryki dla klas *Array* i *Queue*.

3.3. Mechanizm kolejkowania operacji

Kolejkowanie prostych operacji ma na celu takie zgrupowanie poszczególnych etapów algorytmu przetwarzania obrazu, aby były one wykonane w sposób jak najbardziej efektywny. Pojęcie efektywności związane jest tu z dużą szybkością przetwarzania. W wielu przypadkach poprawność wyniku nie zależy od kolejności wykonywanych operacji, odpowiednia kolejność ma za to zasadniczy wpływ na szybkość wykonywania obliczeń. Jest to szczególnie istotne w przypadku przetwarzania danych na GPU, gdzie ich transfer z i do pamięci karty graficznej jest stosunkowo wolny. Powoduje to, że wykonanie prostej operacji na GPU, przesłanie wyniku do CPU, a następnie realizacja kolejnej operacji na GPU jest bardzo nieefektywne, ponieważ wymaga ponownego przesyłania danych z i do pamięci karty.

Mechanizm kolejkowania operacji umożliwia pełne wykorzystanie architektury wektorowej i wielowątkowej procesora GPU.

3.4. Implementacja biblioteki

Biblioteka zaimplementowana została w języku C++, do jej kompilacji użyto kompilatora GCC 4.1.1. Programy shader'ów napisane zostały w języku GLSL. W momencie uruchomienia programu lista operacji z kolejki (*Queue*) zamieniana jest na program shader'a, przesyłana do GPU i wykonywana.

4. Wyniki badań

Działanie biblioteki przetestowane zostało na przykładzie implementacji algorytmu fotograficznego operatora tonów (wersja lokalna operatora). Szczegółowy opis algorytmu można znaleźć w pracach Reinhard'a et al. [5,6]. Fotograficzny operator tonów jest reprezentatywnym algorytmem testowym do przetwarzania i analizy obrazów HDR. Realizacja tego algorytmu wymaga wykonania szeregu podstawowych operacji na macierzach wejściowych (obrazach HDR), takich jak dodawanie, mnożenie, dzielenie, potęgowanie czy obliczanie wartości logarytmu. Realizacja fotograficznego operatora tonu wymaga również wykonania operacji akumulacji podczas wyznaczania wartości średniej logarytmicznej oraz splotu. Implementacja analizy otoczenia pikseli bazuje na wykorzystaniu struktury *WriteMask*. Szczegółowy opis implementacji algorytmu fotograficznego operatora tonów, opartej na prezentowanej bibliotece, przedstawiono w [7].

Celem przeprowadzonych testów było wykazanie przydatności biblioteki do zadań przetwarzania oraz analizy obrazów HDR. Podczas badań oceniono poprawność uzyskanych rezultatów w stosunku do softwarowej implementacji wzorcowej, pochodzącej z pakietu *pfstools* [8] oraz przedstawiono wyniki przyśpieszenia obliczeń, uzyskane dzięki wykorzystaniu opracowanej biblioteki.

Testy przeprowadzone zostały na pięciu wybranych zdjęciach HDR *(BurnedRedwood, TuolumneTree, BoyScoutTrail16, BristolBridge* oraz *Memorial*) o różnych rozdzielczościach. Obliczenia wykonano na komputerze z procesorem AMD Athlon X2 3800+ (zegar 2.0 GHz) wyposażonym zamiennie w dwie karty graficzne nVidia Geforce 7600 GT (256 MB RAM, magistrala 128-bitowa, zegar 650 MHz, zegar pamięci 1400 MHz) oraz nVidia Geforce 8800 GTS (640 MB RAM, magistrala 320-bitowa, zegar 500 MHz, zegar pamięci 1600 MHz, zegar shaderów 1200 MHz).

Tab. 1. Wyniki testów wydajnościowych dla lokalnego fotograficznego operatora tonów
Tab. 1. Results of effeciency tests for the photographics local tone operator

rozmiar obrazu HDR	Czas wykonania algorytmu oraz uzyskane przyśpieszenie na CPU i GPU				
	CPU *pfs*	GPU nVidia 7600GT		GPU nVidia 8800GTS	
	t [*ms*]	*t* [*ms*]	*p*	*t* [*ms*]	*p*
512x768	430	27	15.9x	17	25.2x
1000x1504	1563	106	14.7x	33	47.3x
2048x1536	5912	221	26.7	82	72.1x
2272x1704	7782	259	30.0x	98	79.4x
2000x3008	8226	376	21.8x	144	57.1x

Poprawność uzyskanych rezultatów oceniona za pomocą algorytmu VDP *(ang. Visual Difference Predictor*) [9]. Obrazy wyjściowe uzyskane w wyniku kompresji obrazów HDR za pomocą fotograficznego operatora tonów były takie same dla wersji programowej (wzorcowej) i sprzętowej (z zastosowaniem biblioteki GPU). Algorytm VDP nie wykazał percepcyjnych różnic pomiędzy obrazami.

Zaimplementowana biblioteka w znaczącym stopniu przyśpiesza przetwarzanie obrazów HDR. W najlepszym przypadku uzyskano ponad 79-krotne przyśpieszenie przy zachowaniu oryginalnej jakości obliczeń. Uzyskane przyśpieszenie rośnie wraz ze wzrostem rozdzielczości zdjęcia. Wynika to z coraz efektywniejszego wykorzystywania możliwości zrównoleglenia obliczeń na GPU. Dla nowszego procesora nVidia 8800GTS uzyskano prawie 3-krotnie lepsze wyniki niż dla poprzedniej generacji GPU nVidia 7600GT. Przyczynił się do tego nie tylko wzrost wydajności sprzętowej procesora, ale również zmiany w architekturze, które w coraz większym stopniu predysponują procesory graficzne do wykonywania obliczeń wektorowych zarezerwowanych do tej pory dla CPU.

5. Podsumowanie i kierunki dalszych prac

W artykule zaprezentowano architekturę oraz implementację autorskiej biblioteki przeznaczonej do przetwarzania i analizy obrazów HDR. Biblioteka wykorzystuje procesor graficzny GPU w charakterze jednostki wspomagającej obliczenia. Poprawność działania biblioteki przetestowano na przykładzie algorytmu kompresji tonów obrazów HDR, uzyskując ponad 79-krotne przyśpieszenie obliczeń w porównaniu z softwarową implementacją wzorcową [8]. Zaimplementowana biblioteka stanowi uniwersalne narzędzie, które może zostać wykorzystane do większości operacji związanych z przetwarzaniem i analizą obrazów HDR.

W najbliższej przyszłości planowane jest dostosowanie biblioteki do nowej generacji kart graficznych firmy nVidia z serii 8. Skuteczność praktycznego zastosowania biblioteki wymaga ciągłego dostosowywania jej do nowych wersji sprzętu. Planowane jest również wzbogacanie biblioteki w nowe funkcje, które pozwolą na realizację algorytmów z zakresu percepcyjnej analizy obrazów HDR.

Praca naukowa finansowana ze środków na naukę w latach VIII 2006 - VII 2008 jako projekt badawczy.

6. Literatura

[1] J. L.T. Cornwall, O. Beckmann, P. H.J. Kelly. Accelerating a C++ Image Processing Library with a GPU, POHLL 2006: Workshop on Performance Optimization for High-Level Languages and Libraries (colocated with IPDPS06, Rhodes), 2006.
[2] G. Krawczyk, K. Myszkowski, H.-P. Seidel. Perceptual effects in real-time tone mapping, Spring Conference on Computer Graphics 2005, ACM, 2005.
[3] J. D. Owens, D. Luebke, N. Govindaraju, M. Harris, J. Kreuger, A. E. Lefohn, T. Purcell. A survey of general-purpose computation on graphics hardware. Eurographics 2005, State of the Art Reports, str. 21–51, September 2005.
[4] nVidia Technologies, http://www.nvidia.com/page/geforce8.html
[5] E. Reinhard, M. Stark, P. Shirley, J. Ferwerda. Photographic Tone Reproduction for Digital Images, ACM Trans. on Graph. t.21, n.3, str. 267-276, 2002.
[6] E. Reinhard, G. Ward, Greg, S. Pattanaik, P. Debevec, High Dynamic Range Imaging. Data Acquisition, Manipulation, and Display, wydawnictwo Morgan Kaufmann, 2005.
[7] D. Pająk. General-Purpose Computation Using Graphics Hardware for Fast HDR Image Processing. Proc. of 11-th Central European Seminar on Computer Graphics, 2007.
[8] R. Mantiuk, G. Krawczyk, R. Mantiuk. High dynamic range imaging pipeline: Perception-motivated representation of visual content. Proc. of Human Vision and Electronic Imaging XII, IS&T/SPIE's Annual Symposium on Electronic Imaging, SPIE Proceedings Series, 2007.
[9] R. Mantiuk, K. Myszkowski, H. P. Seidel. Visible difference predicator for high dynamic range images. Proceedings of IEEE International Conference on Systems, Man and Cybernetics, str. 2763–2769, 2004.

Artykuł recenzowany