Marek R. OGIELA [1], Tomasz HACHAJ [2]
[1] AGH AKADEMIA GÓRNICZO – HUTNICZA W KRAKOWIE, al. Mickiewicza 30, 30-059 Kraków
[2] UNIWERSYTET PEDAGOGICZNY W KRAKOWIE, KATEDRA INFORMATYKI I METOD KOMPUTEROWYCH, ul. Podchorążych 2, 30-084 Kraków

Klasyfikacja i wizualizacja zobrazowań dpTK w środowisku trójwymiarowym

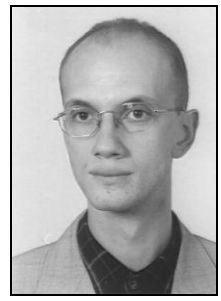
Prof. dr hab. Marek R. OGIELA

Profesor zwyczajny na Akademii Górniczo – Hutniczej w Krakowie. Prowadzi badania nad kognitywnymi systemami informacyjnymi nowej generacji, a także kryptografią i podziałem sekretów. Jest członkiem wielu renomowanych towarzystw naukowych, a także autorem ponad 200 publikacji o zasięgu międzynarodowym.

e-mail: mogiela@agh.edu.pl

Dr inż. Tomasz HACHAJ

Absolwent kierunku Informatyka Politechniki Krakowskiej. Obecnie adiunkt w Katedrze Informatyki i Metod Komputerowych Uniwersytetu Pedagogicznego w Krakowie. Jego zainteresowania naukowe obejmują metody komputerowego przetwarzania i analizy obrazów.



e-mail: tomekhachaj@o2.pl

Streszczenie

W artykule został zaprezentowany wspomagany sprzętowo algorytm wizualizujący trójwymiarowe dyskretne pola skalarne o teoretycznie dowolnych rozmiarach. Zostały również przedstawione oryginalne wyniki badań, w których określono zależność pomiędzy szybkością wizualizacji (fps) a ilością przesłań danych pomiędzy pamięcią RAM komputera i kartą graficzną. Zbadano także w jaki sposób szacowanie gradientu pola skalarnego przed procesem wizualizacji wpływa na przyspieszenie tworzenia grafiki w wypadku dużych objętości. Jeżeli zbiór danych został podzielony na wiele podzbiorów (w przeprowadzany doświadczeniu było ich 512) algorytmy estymujące gradient we wstępnym przetwarzaniu działały wolniej niż liczące go czasie rzeczywistym, ponieważ te drugie redukują ilość danych, które muszą być przesłane do GPU.

Słowa kluczowe: wizualizacja trójwymiarowych pól skalarnych, dynamiczna perfuzja mózgowa, duże zbiory danych, analiza kognitywna.

Classification and visualization of dpCT in three-dimensional environment

Abstract

In this paper a new hardware accelerated algorithm pipeline (Fig. 1) for visualization of three-dimensional scalar fields without limitation on dataset size (Fig. 2) is described. There are presented original results of research on average performance speed (fps) of rendering algorithms as a function of data transitions between RAM and GPU (Tab 1. and Fig. 3). The speed of rendering decreases with number of partitioning and view-aligned slices. It has also been investigated how the pre-rendering gradient estimation influences the visualization process in case of large volumetric datasets. It is shown that in case of large partitioning of volume it is better to transfer less data to GPU memory and to compute the gradient value on the fly. The potential usage of that algorithm as a visualization module for the system performing cognitive analysis of dpCT data [3, 4, 5] is presented. The output data of the diagnostic algorithm is a superimposition onto the volume CT data (Fig. 4). This particular solution gives additional support to medical personnel by supplying them with simultaneous visualization of medical data of different modalities enabling more accurate diagnosis.

Keywords: volume rendering, dynamic brain perfusion, large datasets, cognitive analysis.

1. Tematyka badań

Współczesna neuroradiologia posługuje się wieloma technikami badawczymi, które wykorzystuje się w codziennej praktyce medycznej. Wśród nich istotną rolę pełni badanie dynamicznej perfuzji mózgowej tomografii komputerowej (dpTK). Przy jego pomocy możliwe jest określenie ilości krwi przepływającej zarówno przez całe mózgowie jak i poszczególne jego tkanki. W wyniku badania perfuzyjnego otrzymywany jest zestaw dwuwymiarowych map perfuzyjnych, które zawierają wartości istotnych z medycznego punktu widzenia parametrów przepływu krwi. Na podstawie licznych przeprowadzonych badań udowodniono, że wartości parametrów map CBF (ang. *cerebral blood flow*) oraz CBV (ang. *cerebral blood volume*) są skorelowane z prognozami dotyczącymi ewolucji zmian niedokrwiennych tkanki mózgowej [1]. Prawidłowa interpretacja map i znajomość prognoz pozwala na podjęcie odpowiednich działań w wypadku podejrzenia u pacjenta zmian chorobowych zakłócających przepływ krwi w mózgu, w tym w szczególności niedokrwiennego udaru mózgu. W ostatnich latach zostało zaproponowanych kilka metod wspomagających lekarza w procesie oceny zmian uwidocznionych na mapach dpTK. Wśród nich można wymienić algorytm Wintermarka [2], oraz system DMD (ang. *Detection Measure and Description*) zaproponowany w [3, 4, 5]. Obydwa wymienione rozwiązania generują prognostyczne mapy ewolucji niedokrwienia, lecz tylko drugie z nich pozwala na automatyczną diagnostykę zmian krwotocznych. Dodatkowo drugie rozwiązanie wyposażone jest w deformowalny atlas mózgu opisujący tkanki znajdujące się w obszarze zaburzonej perfuzji. Walidacja algorytmu DMD przeprowadzona na zbiorze składającym się z 37 przypadków klinicznych wykazała, że w 77% przypadkach opis zdjęcia wygenerowany przez algorytm autorów w pełni zgadzał się z opisem sporządzonym przez lekarza radiologii.

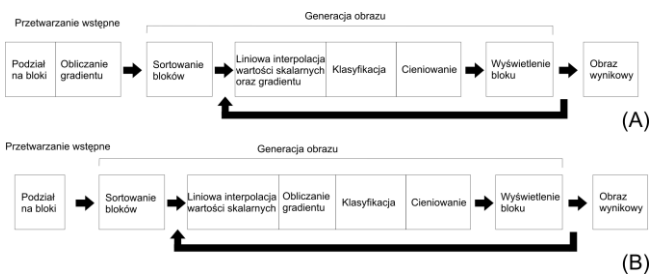
Od współczesnych systemów wspomagających personel medyczny wymaga się nie tylko odpowiedniego poziomu wiarygodności generowanych podpowiedzi i szybkości działania. Równie ważnym czynnikiem jest sposób wizualizacji otrzymanych wyników. Zobrazowania dpTK są dwuwymiarowe, lecz przy pomocy superimpozycji na dane trójwymiarowe (wolumetryczne) można dostarczyć personelowi medycznemu dodatkowych informacji łącząc dane medyczne różnych modalności. Wizualizacja trójwymiarowych danych medycznych (albo szersze zagadnienie – wizualizacja trójwymiarowych pól skalarnych) jest powszechnie znanym zagadnieniem wykorzystywanym w medycynie, fizyce obliczeniowej i innych dyscyplinach nauki [6]. Wśród algorytmów pozwalających na tego typu wizualizację istotną rolę pełnią algorytmy bezpośredniego wyświetlania trójwymiarowych danych skalarnych [7]. Stosowane obecnie algorytmy bezpośrednie oparte są na sprzętowo zaimplementowanych funkcjach udostępnianych przez kartę graficzną z procesorem GPU i dzielą się na dwie grupy: oparte na śledzeniu promienia światła przechodzącego przez badane pole skalarne [8] (ang. *volume ray casting*) oraz na reprezentacji obiektu przez zbiór dwuwymiarowych przekrojów wizualizowanego pola skalarnego [9] (ang. *view-aligned texture-based algorithms*). Jednym z problemów jaki napotyka się przy wizualizacji trójwymiarowych pól skalarnych jest rozmiar zbioru danych, który może przekroczyć rozmiar pamięci powszechnie używanych kart graficznych. Aby zaradzić temu problemowi zaproponowano wiele rozwiązań. Można tu wymienić podział zbioru na mniejsze, mieszczące się w pamięci karty graficznej fragmenty [10, 11], kompresję danych przechowywanych w pamięci [12] czy zastosowanie estymacji wartości wokseli metodą Monte Carlo [13].

Oryginalne osiągnięcia zaprezentowane w tym artykule to opis oraz wyniki testów, które przeprowadziliśmy na stworzonym przez nas wspomaganym sprzętowo algorytmie wizualizującym trójwymiarowe dyskretne pole skalarne o teoretycznie dowolnych

rozmiarach. Zaproponowane rozwiązanie jest dedykowane w szczególności do wyświetlania dużych (których rozmiar przekracza pamięć karty graficznej) „bardzo przezroczystych" zbiorów danych (posiadających wiele półprzezroczystych pikseli w funkcji transferu [9]). Typowe metody przyspieszania wizualizacji (takie jak *early ray termination* [10]) dla tego typu trójwymiarowych obiektów nie przynoszą żadnych efektów, dlatego też zdecydowaliśmy się użyć techniki zapewniającej stały czas wyświetlania zarówno dla przezroczystych jak i nie – przezroczystych trójwymiarowych struktur. Przedstawiliśmy również oryginalne wyniki badań, w których określamy zależność pomiędzy szybkością wizualizacji (ilość klatek na sekundę – fps) a ilością przesłań danych pomiędzy pamięcią RAM komputera i kartą graficzną. Sprawdzamy również o ile szacowanie gradientu pola skalarnego przed procesem wizualizacji wpływa na przyspieszenie tworzenia grafiki w wypadku dużych objętości. Do tej pory w literaturze nie napotkaliśmy tego typu wyników badań, dlatego stanowią one istotny wkład nie tylko w obszarze wizualizacji danych medycznych, ale również dla całości problematyki wizualizacji pól skalarnych dużych rozmiarów.

2. Potok przetwarzania danych w algorytmie wizualizacyjnym

Nasz algorytm oparty jest na metodzie mapowania trójwymiarowych tekstur (3D texture mapping) [9], w której przy pomocy wyrysowania zbioru dwuwymiarowych przekrojów wizualizowanego pola skalarnego uzyskuje się wrażenie trójwymiarowości oglądanego obiektu. Opracowaliśmy dwie wersje naszego algorytmu: liczącą estymatę gradientu w kroku przetwarzania wstępnego oraz estymującą gradient w czasie generacji obrazu. Schemat potoku przetwarzania danych zaproponowanych algorytmów widoczny jest na rys. 1. Algorytmy przesyłają dane do pamięci karty graficznej przy użyciu trójwymiarowych tekstur (trójwymiarowe tekstury można traktować jako trójwymiarowe tablice – macierze – danych). W naszym przypadku każda komórka tablicy zawiera gęstość wizualizowanej tkanki (wartość ISO) zmierzonej podczas badania TK oraz w wypadku algorytmu obliczającego gradient w kroku przetwarzania wstępnego (rys. 1A) wartości trzech dyskretnych składowych gradientu liczonych w zadanym punkcie.



Rys. 1. Potok przetwarzania danych w zaproponowanym algorytmie wizualizacyjnym, opis w tekście
Fig. 1. Rendering pipeline in the proposed visualization algorithm, description in the text

2.1. Wstępne przetwarzanie

Przed przystąpieniem do procesu wizualizacji dużego zbioru danych konieczne jest przeprowadzenie pewnych wstępnych obliczeń. Pierwszym krokiem jest podział dyskretnych wartości trójwymiarowego pola skalarnego na mniejsze, mieszczące się w pamięci karty graficznej trójwymiarowe bloki danych. Każdy blok danych (rozpoczynając od początkowej tablicy) dzielony jest na osiem sześcianów o takiej samej objętości (ang. *octree subdivision* – podobnie jak w [11]) tak długo, aż otrzymamy blok danych o odpowiednich rozmiarach. Jeżeli rozmiar danych przekracza dostępną pamięć RAM komputera, możliwe jest przechowywanie niektórych bloków danych na dysku twardym. Takie rozwiązanie jednak znacznie utrudni interaktywną pracę z programem, ponieważ „doczytywanie" poszczególnych bloków danych z dysku twardego może być czasochłonne.

Symulacja oświetlenia znacząco pomaga poprawnie zinterpretować kształt wyświetlanych trójwymiarowych obiektów. W procesie wizualizacji zastosowaliśmy symulację oświetlenia przy pomocy modelu Phonga [14]. Model ten jest sumą trzech składowych: światła otoczenia (ang. *ambient*), rozproszonego (ang. *diffuse*) oraz odbitego zwierciadlanie (ang. *specular*).

$$L = L_{ambient} + L_{diffuse} + L_{specular}, \tag{1}$$

Składniki sumy równania (1) liczone są oddzielnie dla każdego z pikseli. Nasz algorytm wykorzystuje światło otoczenia oraz rozproszone ($L_{specular} = 0$). Składowa światła otoczenia jest zdefiniowany wzorem:

$$L_{ambient} = I_{ambient} \cdot K_{ambient}, \tag{2}$$

gdzie $I_{ambient}$ jest intensywnością światła rozproszonego a $K_{ambient}$ jest współczynnikiem materiału. W naszym przypadku $L_{ambient}$ jest stałą, dzięki czemu będą widoczne nawet fragmenty obiektu, które nie są oświetlone bezpośrednio.

Współczynnik światła rozproszonego wyraża się wzorem

$$L_{diffuse} = \begin{cases} I_{diffuse} \cdot K_{diffuse} \cdot \cos(\alpha), & |\alpha| < 90^{\circ} \\ 0 \quad w \quad pozostaych \quad przypadkach \end{cases}, \tag{3}$$

gdzie $I_{diffuse}$ jest intensywnością źródła światła, $K_{diffuse}$ współczynnikiem materiału a α jest kątem pomiędzy wektorem zdefiniowanym przez punktowe źródło światła i oświetlany punkt płaszczyzny, a wektorem normalnym płaszczyzny.

Podstawową informacją wymaganą przy symulacji oświetlenia powyższym modelem jest znajomość wektorów normalnych powierzchni. W przypadku wizualizacji opartej na dyskretnych wartościach pola skalarnego wektory normalne szacowane są poprzez wyliczenie wektora gradientu dla każdego dyskretnego punktu:

$$\nabla f(x, y, z) = \left(\frac{df}{dx}, \frac{df}{dy}, \frac{df}{dz} \right), \tag{4}$$

Do oszacowania gradientu użyliśmy ilorazu różnicowego centralnego.

2.2. Generacja obrazu

W tym etapie następuje „wyrysowanie" w odpowiedniej kolejności bloków danych. Pierwszym krokiem jest posortowanie bloków od najdalszego względem obserwatora do najbliższego. Następnie algorytm dokonuje liniowej interpolacji wartości ISO oraz (w wypadku algorytmu A) wartości gradientu. Dokonuje tego przy pomocy wartości przechowywanych w trójwymiarowej teksturze, która została przekazana do pamięci GPU. Procesor graficzny wylicza szukaną wartość we wskazanym punkcie wykorzystując sprzętową implementację interpolacji trójliniowej, co znacznie przyspiesza obliczenia. W wypadku algorytmu B na tym etapie szacowana jest również dyskretna wartość gradientu w każdym punkcie należącym do bloku (dokonywane jest to podobnie jak w algorytmie A przy pomocy ilorazu różnicowego centralnego na podstawie wartości ISO). Kolejnym krokiem jest klasyfikacja wyświetlanych wokseli (w wypadku danych TK - tkanek). Klasyfikacja polega na zastąpieniu wartości ISO pola skalarnego przez wartości koloru modelu RGBA. Wartości RGBA przyporządkowane do odpowiednich wartości ISO zdefiniowane są przez funkcję transferu, która przechowywana jest w postaci tablicy LUT (ang. *lookup table*).

Ponieważ znane są wartości RGBA woksela oraz dyskretna wartość gradientu obliczana jest teraz oświetlenie przy użyciu opisanego wcześniej modelu Phong'a (na rys. 1 krok „cieniowa-

nie"). Ostatnim etapem generacji obrazu jest wyświetlenie „wyrysowanego" bloku. W naszym algorytmie użyliśmy mapowania trójwymiarowych tekstur. Dwuwymiarowe przekroje ustawiane są tak, aby prosta zdefiniowana przez punkt, w którym znajduje się obserwator oraz kierunek wektora obserwacji była prostopadła do wszystkich przekrojów (ang. *view aligned slices*). Algorytm ten pozwala na sterowanie szybkością procesu wizualizacji pola skalarnego poprzez zmniejszenie liczby wyświetlanych przekrojów (a co za tym idzie zmniejszenie jakości otrzymanego obrazu). Ta technika pozwala, jeśli zachodzi taka potrzeba, na wyświetlenie nawet bardzo dużych obrazów w czasie rzeczywistym (fps > 30). Wykorzystuje się to na przykład podczas interakcji w aplikacji (rotacja / translacja obiektu względem obserwatora). Nasz algorytm wykorzystuje standardowe dla wybranej przez nas techniki rysowania równanie, określające wartość piksela na wynikowym obrazie dwuwymiarowym:

$$C = \alpha_{src} \cdot C_{src} + (1 - \alpha_{src}) C_{dst}, \tag{5}$$

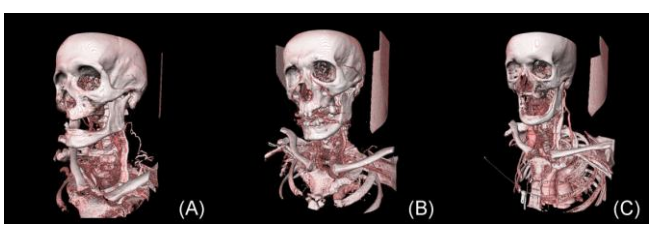
gdzie C jest kolorem wynikowym, C_{dst} jest wartością koloru rozważanego woksela, a C_{src} i α_{src} są wartościami koloru i współczynnika przeciwnego do przezroczystości kolejnego leżącego wzdłuż prostej obserwacji woksela (iteracja przebiega od najdalszego do najbliższego woksela względem obserwatora).

Po zakończeniu generacji obrazu otrzymujemy dwuwymiarowy obraz wynikowy trójwymiarowego pola skalarnego.

3. Wyniki

Algorytmy zaprezentowane w poprzednim rozdziale zostały przez nas zaimplementowane w technologii .NET C#, przy użyciu API graficznego XNA Framework 2.0 (Direct X 9.0) a program GPU został napisany w HLSL. Program został przetestowany na komputerze wyposażonym w procesor Intel Core 2 Duo CPU 3.00 GHz, 3.25 GB pamięci RAM, kartę graficzną Nvidia GeForce 9600 GT i 32 – bitowy system operacyjny Windows 7. Ponieważ stworzone przez nas rozwiązanie zostało w całości napisane w kodzie zarządzanym jest w pełni przenośne pomiędzy komputerami wyposażonymi w system operacyjny wspierający .NET framework oraz kartę graficzną z programowalnym potokiem przetwarzania.

Użyty przez nas zbiór testowy składał się z trzech trójwymiarowych zobrazowań tomografii komputerowej o wymiarach 512x512x425, 512x512x433 i 512x512x497 wokseli (rys. 2).



Rys. 2. Wizualizacja trzech zobrazowań TK wchodzących w skład zbioru testowego wykonana przez nasz algorytm. Rozmiary zobrazowań to: (A) 512x512x425, (B) 512x512x433, (C) 512x512x497 wokseli

Fig. 2. Visualization of three computer tomography images from the test set performed by our algorithm. Their dimensionalities are: (A) 512x512x425, (B) 512x512x433 and (C) 512x512x497 voxels

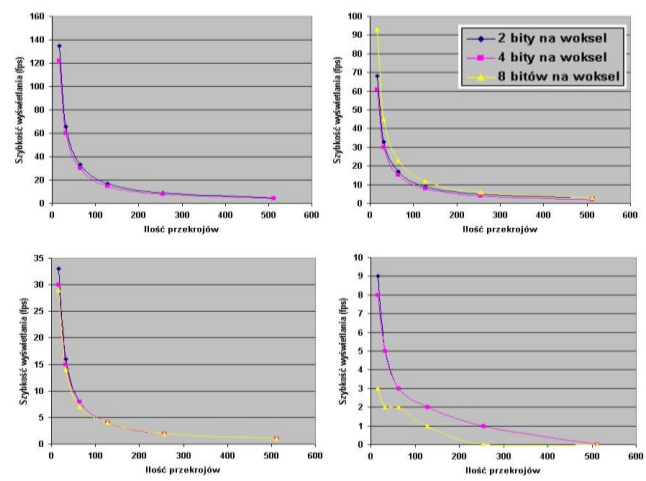
Użyliśmy funkcji transferu, dla której wokesele reprezentujące gęste tkanki (kości) są nieprzezroczyste, natomiast woksele reprezentujące naczynia krwionośne mają wysoki współczynnik przezroczystości. Pozostałe woksele są całkowicie przezroczyste.

Dla wszystkich pól skalarnych wchodzących w skład zbioru testowego wyznaczyliśmy średnią szybkość generowania obrazów wynikowych (fps) przez algorytmy przy zmiennej liczbie podziałów na bloki oraz zmiennej liczbie przekrojów użytych w procesie wizualizacji. Rozmiar wynikowego dwuwymiarowego obrazu został ustalony na 1024x768 pikseli. Wyniki zaprezentowano w tab. 1 oraz na rys. 3. Sprawdziliśmy również ile przekrojów w algorytmie rysowania jest potrzebnych, aby uniknąć artefaktów w procesie wizualizacji (np. nieciągłości powierzchni obiektów). Algorytm B został przetestowany w dwóch wersjach: B1, w której wartości ISO zapisane w teksturze 3D przechowywane są w dwóch bajtach danych oraz B2, w której oprócz dwubajtowej wartości ISO do każdego piksele przypisane są jeszcze dwa dodatkowe bajty. Te dodatkowe bajty nie są w tej chwili wykorzystywane, ale mogą zostać użyte do zapamiętania dodatkowych informacji o wokselu (np. danych wymaganych przez bardziej złożony proces klasyfikacji tkanek TK wykorzystujący wiele funkcji transferu, parametrów oświetlanego materiału itp.). Implementacja algorytmu A wykorzystuje trójwymiarową teksturę, w której dane woksela pamiętane są w ośmiu bajtach: 2 przeznaczone są na wartość ISO i po 2 na każdą z trzech składowych dyskretnej wartości wektora gradientu.

Tab. 1. Średnia szybkość generowania obrazów wynikowych (fps) algorytmów dla różnych podziałów na bloki oraz liczby przekrojów użytych w procesie wizualizacji

Tab. 1. Average performance speed (fps) of rendering algorithms as a function of size of octree and view aligned slices count

Podział na bloki (liczba bloków)	Liczba przekrojów	Rozmiar danych: 512x512x425			Rozmiar danych: 512x512x443			Rozmiar danych: 512x512x497		
		Alg. A	Alg. B1	Alg. B2	Alg. A	Alg. B1	Alg. B2	Alg. A	Alg. B1	Alg. B2
1x1x1 (1)	512		4	4		5	4		4	
	256		8	7		9	8		8	
	128		15	14		17	15		16	
	64		29	27		33	30		31	
	32		59	54		66	60		61	
	16		118	110		135	122		125	
2x2x2 (8)	512	3	2	2	3	3	2		2	2
	256	6	4	4	6	5	4		4	4
	128	12	8	7	12	9	8		8	7
	64	23	15	14	23	17	15		15	14
	32	46	29	27	45	33	30		31	27
	16	95	59	55	93	68	61		63	56
4x4x4 (64)	512	1	1	1	1	1	1	1	1	1
	256	2	2	2	2	2	2	2	2	2
	128	4	4	4	4	4	4	3	4	4
	64	8	7	7	7	8	8	5	8	7
	32	16	15	13	14	16	15	11	15	14
	16	32	29	27	29	33	30	22	31	28
8x8x8 (512)	512	<1	<1	<1	<1	<1	<1	<1	<1	<1
	256	<1	1	1	<1	1	1	<1	1	1
	128	1	2	2	1	2	2	<1	2	2
	64	2	3	3	2	3	3	1	3	3
	32	2	5	5	2	5	5	2	5	5
	16	3	8	8	3	9	8	3	8	8

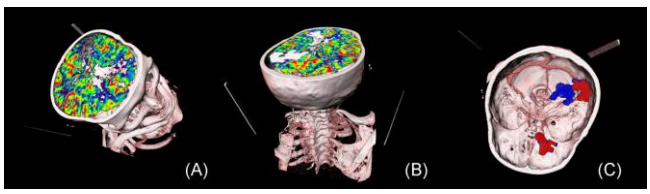


Rys. 3. Średnia szybkość generowania obrazów wynikowych (fps) algorytmów dla różnych podziałów na bloki oraz liczby przekrojów użytych w procesie wizualizacji (wyniki z Tab. 1 dla danych TK o rozmiarze 512x512x443). Dla pozostałych przypadków kształt wykresu nie różni się znacząco. Liczba bloków poczynając od lewego górnego rogu: 1x1x1 (1), 2x2x2 (8), 4x4x4 (64) i 8x8x8 (512)

Fig. 3. Average performance speed (fps) of rendering algorithms on volumetric CT data as a function of size of octree and view aligned slices count (the results from Tab. 1 for data with dimensionality 512x512x443). For other volumes the shape of the plot does not differ significantly. The number of blocks from the top-left corner is: 1x1x1 (1), 2x2x2 (8), 4x4x4 (64) i 8x8x8 (512)

Dla niektórych podziałów nie wygenerowaliśmy danych wynikowych algorytmów A i B2. Wynika to z naszej implementacji, która podczas budowy tekstury trójwymiarowej alokuje więcej pamięci niż może zostać zarezerwowane dla procesu 32-bitowego systemu Windows (2 GB). Ten techniczny problem zostanie wyeliminowany kiedy program zostanie uruchomiony w systemie 64-bitowym.

Implementacja naszego algorytmu wizualizacyjnego pozwala również na superpozycję dwuwymiarowych map dpTK oraz map prognostycznych wygenerowanych przez system DMD na trójwymiarowe dane TK. Przykładowa wizualizacja tego typu przedstawiona jest na rys. 4.



Rys. 4. Przykładowa wizualizacja wykonana przy pomocy naszego rozwiązania. (A) Superimpozycja mapy CBF na zobrazowanie TK. (B) Superimpozycja mapy CBF na TK z zaznaczonymi wykrytymi przez algorytm DMD obszarami asymetrii perfuzji (potencjalny obszar chorobowy zaznaczony jest białym kolorem). (C) Nałożenie mapy prognostycznej zmian niedokrwiennych mózgu na zobrazowanie TK. Niebieski region oznacza tkanki zagrożone, czerwony – tkanki objęte udarem

Fig. 4. The example visualization performed with proposed framework. (A) Superimposition of CBF onto volume CT data. (B) Superimposition of CBF map with marked perfusion abnormalities (white regions) detected by DMD algorithm onto volume CT data. (C) Superimposition of prognostic map with marked perfusion abnormalities onto volume CT data. Blue regions mark tissues at risk, red – infarcted tissues

4. Dyskusja i wnioski

W artykule zaprezentowaliśmy wspomagany sprzętowo algorytm wizualizujący trójwymiarowe dyskretne pola skalarne o teoretycznie dowolnych rozmiarach. Nasz algorytm pozwala na wyświetlenie trójwymiarowych danych medycznych typowych rozmiarów z szybkością umożliwiającą wygodną interakcję z użytkownikiem na niedrogim i powszechnie dostępnym „konsumenckim" sprzęcie. Na podstawie danych z tab. 1 oraz rys. 3 możemy spostrzec, że szybkość wizualizacji silnie maleje wraz ze wzrostem liczby przekrojów (kształt wykresu zbliżony jest do funkcji ekspotencjalnej). Szybkość wyświetlania danych zależy również od rozmiaru dyskretnego zbioru wokseli – im zbiór jest większy, tym wolniej działa cały proces. Bardzo interesująca jest zależność pomiędzy liczbą bloków, na który podzielono wyświetlany zbiór a szybkością wyświetlania (fps) jeżeli porównamy ją pomiędzy poszczególnymi algorytmami. Wraz ze wzrostem liczby bloków maleje fps. Wynika to faktu, że poszczególne bloki składowe przenoszone są z pamięci RAM do pamięci karty graficznej, co zajmuje stosunkowo dużo czasu. Im większa porcja danych przenoszona jest jednorazowo, tym dłużej trwa cały proces. Z tego powodu implementacja algorytm B1 w większości przypadków działa szybciej niż algorytmu B2 (ponieważ bloki w B2 są dwukrotnie większe niż w B1). Przy niewielkiej ilości bloków algorytm A, który ma wyznaczone wartości gradientu w kroku wstępnego przetwarzania, działa szybciej niż algorytmy z grupy B, które obliczają gradient dynamicznie za każdym razem przy generowaniu grafiki. Im więcej bloków wykorzystujemy, tym mniejsza jest różnica szybkości pomiędzy A i B, aż w przypadku 512 bloków algorytmy z grupy B zaczynają działać szybciej. Dzieje się tak, ponieważ algorytm A przenosi większe ilości danych niż algorytmy B1 i B2 (odpowiednio cztery i dwa razy więcej) i czas potrzebny na dynamiczne liczenie gradientu staje się niższy, niż czas potrzebny na alokacje tej ilości pamięci przez GPU. Można więc wysunąć wniosek, że dla dużych objętości, które wymagają znaczącej liczby bloków można pozwolić sobie na poważną oszczędność pamięci przy wizualizacji nie tracąc przy tym na szybkości działania całego rozwiązania. Nasze badania wykazały również, że dla zbioru danych o rozmiarze około 512^3 w celu wyeliminowania artefaktów wizualizacji dla podziału 1x1x1 wymaganych jest co najmniej 512 przekrojów, dla 2x2x2: 256, dla 4x4x4: 128 i dla 8x8x8: 64.

Zaproponowany przez nas algorytm został z powodzeniem wykorzystany w module wizualizacyjnym systemu DMD. Dzięki superimpozycji dwuwymiarowych danych dpTK na trójwymiarowe zobrazowanie TK nasze rozwiązanie dostarcza dodatkowego wsparcia personelowi medycznemu pozwalając na wirtualne zestawienie obrazowych danych medycznych różnych modalności. Nasz algorytm wizualizacyjny może zostać użyty również do innej modalności trójwymiarowych danych medycznych jak i dla dowolnego zdyskretyzowanego trójwymiarowego pola skalarnego.

Zaproponowane w pracy rozwiązania będą w dalszym ciągu rozwijane. W wypadku algorytmu wizualizacyjnego planujemy opracowanie efektywnego algorytmu pozwalającego na ominięcie w procesie wizualizacji tych bloków, które zawierają jedynie przezroczyste woksele (ang. *empty space skipping*). Drugim kierunkiem badań jest ściślejsze powiązanie elementu wizualizacyjnego z algorytmem DMD poprzez dodanie możliwości odwzorowywania w przestrzeni trójwymiarowej dwuwymiarowych obszarów chorobowych wykrytych na mapach dpTK oraz na mapach prognostycznych. Do tego celu planujemy zastosować wielowymiarowe funkcje transferu budowane dynamicznie na etapie wstępnego przetwarzania danych.

Praca naukowa finansowana ze środków budżetowych na naukę w latach 2010-2012 jako projekt badawczy MNiSW numer N N516 511939.

5. Literatura

[1] Hoeffner E.G., et al.: Cerebral Perfusion CT: Technique and Clinical Applications, Radiology. Jun;231(3) (2004) 632-44.
[2] Wintermark M., et al.: Comparison of Admission Perfusion Computed Tomography and Qualitative Diffusion- and Perfusion-Weighted Magnetic Resonance Imaging in Acute Stroke Patients, Stroke.;33, (2002) 2025-2031.
[3] Hachaj T., Ogiela M. R.: CAD System for Automatic Analysis of CT Perfusion Maps, Opto-Electronics Review, 19(1), pp. 95-103, (2011), DOI: 10.2478/s11772-010-0071-2.
[4] Ogiela M. R., Hachaj T.: Komputerowe wspomaganie detekcji zmian chorobowych w badaniach perfuzji mózgowej, PAK 2010, 5, s. 453-456.
[5] Hachaj T., Ogiela M. R.: A System for Detecting and Describing Pathological Changes using Dynamic Perfusion Computer Tomography Brain Maps, Computers in Biology and Medicine 41 (2011), pp. 402-410.
[6] Guthe S., Strasser W.: Advanced Techniques for High-Quality Multi-resolution Volume Rendering, Computers & Graphics, Volume 28, Issue 1, February 2004, Pages 51-58.
[7] Nelson M.: Optical Models for Direct Volume Rendering. IEEE Transactions on Visualization and Computer Graphics, 1(2):99-108, June 1995.
[8] Kruger J., Westermann R.: Acceleration Techniques for GPU-based Volume Rendering, Proceedings of the 14th IEEE Visualization 2003 (VIS'03), 2003.
[9] Engel K., et all.: Real-Time Volume Graphics, CRC Press, 2006.
[10] Lee T. H., et all.: Fast Perspective Volume Ray Casting Method Using GPU-based Acceleration Techniques for Translucency Rendering in 3D Endoluminal CT Colonography, Comput Biol Med. 2009 Aug;39(8):657-66.
[11] Xie K., Yang J., Zhu Y. M.: Real-Time Visualization of Large Volume Datasets on Standard PC Hardware, Comput Methods Programs Biomed. 2008 May;90(2):117-23. Epub 2008 Feb 19.
[12] Yu H., et all.: Fast Rendering of Foveated Volumes in Wavelet-Based Representation, The Visual Computer, ISSN: 01782789, Vol: 21, Date: September 2005, Pages: 735-744.
[13] Li X., Yang J., Xie K., Zhu Y. M.: High-Quality Rendering with Depth Cueing of Volumetric Data Using Monte Carlo Integration, Comput Biol Med. 2006 Sep;36(9):1014-25. Epub 2005 Aug 29.
[14] Phong B.T.: Illumination for Computer Generated Pictures, Communications of the ACM, 18(6):311-317, June (1975).

otrzymano / received: 03.09.2011
przyjęto do druku / accepted: 03.10.2011 *artykuł recenzowany*