Adrian Dziembowski
Jakub Stankowski

Katedra Telekomunikacji Multimedialnej i Mikroelektroniki,
Politechnika Poznańska
ul. Polanka 3, 60-965 Poznań
{adziembowski, jstankowski}@multimedia.edu.pl

KKRRiT 2018
Gdańsk, 20-22 czerwca 2018

DOI: 10.15199/59.2018.6.41

SZYBKA SYNTEZA WIDOKÓW WIRTUALNYCH DLA SYSTEMÓW TELEWIZJI SWOBODNEGO PUNKTU WIDZENIA

FAST VIRTUAL VIEW SYNTHESIS FOR FREE-VIEWPOINT TELEVISION SYSTEMS

Streszczenie**: W artykule opisano metodę szybkiej syntezy widoków wirtualnych dla systemów telewizji swobodnego punktu widzenia. Zaprezentowana metoda pozwala na syntezę sekwencji wysokiej rozdzielczości w czasie rzeczywistym, umożliwiając użytkownikowi swobodną nawigację w zarejestrowanej scenie, np. wirtualne przemieszczanie się wokół boiska podczas meczu siatkówki. Opisana metoda została zrealizowana wyłącznie na CPU, a więc może zostać użyta również na komputerach osobistych z tanimi, zintegrowanymi kartami graficznymi.**

Abstract**: In the paper the fast virtual view synthesis method for FTV systems was described. Presented method allows to synthesize high resolution sequences in the real time, allowing the user to freely navigate within acquired scene, e.g. changing viewpoints during the volleyball match. Described method was realized for CPU only, therefore it can be used also on personal computers with cheap, integrated graphic cards.**

Słowa kluczowe**: telewizja swobodnego punktu widzenia, swobodna nawigacja, synteza widoków wirtualnych.**

Keywords**: free-viewpoint television, free navigation, virtual view synthesis.**

1. WSTĘP

Głównym założeniem telewizji swobodnego punktu widzenia (FTV – free-viewpoint television) jest udostępnienie użytkownikowi możliwości płynnej zmiany punktu i kierunku widzenia [5], [15]. W przypadku wyboru punktu widzenia, którego położenie nie pokrywa się z położeniem którejkolwiek z kamer, konieczna jest synteza widoku wirtualnego (rys. 1). W związku z powyższym, opracowanie szybkiego i wydajnego algorytmu syntezy widoków wirtualnych jest kluczowe dla systemów telewizji swobodnego punktu widzenia.

Rys. 1. Widok wirtualny względem widoków rzeczywistych

2. OPIS ALGORYTMU

W celu uzyskania najwyższej możliwej jakości syntezowanych widoków wirtualnych, w zaproponowanym algorytmie użyto hybrydowej metody syntezy, łączącej zalety syntezy w przód (*forward synthesis*) [3] i wstecz (*backward synthesis*) [4], [7]

Synteza wstecz pozwala na przefiltrowanie głębi widoku wirtualnego przed rzutowaniem informacji o kolorze, redukując tym samym wpływ niepoprawnie wyznaczonych, niespójnych map głębi. Wymaga ona jednak dwukrotnego rzutowania punktów – raz z widoków rzeczywistych na widok wirtualny i raz w stronę przeciwną. Synteza w przód wymaga wyłącznie jednego rzutowania każdego punktu, przez co jest operacją szybszą.

W zaproponowanym hybrydowym podejściu punkty rzutowane są wyłącznie raz, a dzięki zapamiętaniu dokładnych pozycji, z których były one rzutowane możliwe jest uzyskanie podobnego efektu, jaki dałaby filtracja mapy głębi widoku wirtualnego.

2.1. Rzutowanie i łączenie głębi

Każdy punkt w dowolnym widoku rzeczywistym może być reprezentowany poprzez wektor m=[x y z 1], gdzie x i y wyrażają pozycję danego punktu w widoku rzeczywistym, a z stanowi odpowiadającą mu wartość głębi. Podczas rzutowania głębi każdy punkt widoku rzeczywistego jest rzutowany do widoku wirtualnego zgodnie ze wzorem:

$$\mathbf{m_V} = \mathbf{m} \cdot \mathbf{P_V} \cdot \mathbf{P_R^{-1}}, \qquad (1)$$

gdzie $\mathbf{m_V} = [x_v\, y_v\, z_v\, 1]$ jest wektorem reprezentującym pozycję punktu w widoku wirtualnym (x_v, y_v) i jego głębię (z_v), czyli odległość pomiędzy pozycją punktu w przestrzeni trójwymiarowej i płaszczyzną przetwornika kamery wirtualnej. $\mathbf{P_R}$ i $\mathbf{P_V}$ są macierzami projekcji, odpowiednio dla widoku rzeczywistego i wirtualnego. Każda z macierzy projekcji łączy parametry wewnętrzne i zewnętrzne kamery: $\mathbf{P} = \mathbf{K} \cdot \mathbf{E}$, gdzie macierz parametrów wewnętrznych **K** zawiera długość ogniskowej danej kamery i jej punkt centralny, a macierz parametrów zewnętrznych **E** składa się z macierzy rotacji i wektora translacji, odpowiadając za przesunięcie i obrót kamery względem globalnego układu współrzędnych [8].

Wynikiem rzutowania głębi są dwie mapy głębi – jedna dla punktów rzutowanych z lewego widoku rzeczywistego i druga dla punktów rzutowanych z widoku prawego. W kolejnym kroku obie mapy są łączone w celu stworzenia jednej spójnej mapy głębi dla widoku wirtualnego. W tym kroku dla każdego punktu wybierana jest mniejsza wartość głębi (a więc punkt znajdujący się bliżej kamery).

2.2. Rzutowanie i mieszanie koloru

W przypadku typowej syntezy wstecz, kolor wszystkich punktów widoku wirtualnego rzutowany jest na podstawie wyznaczonej we wcześniejszym kroku mapy głębi, a operacja ta dokonywana jest przez zastosowanie analogicznego wzoru jak w (1), przy czym w odwrotnym kierunku – a więc na podstawie pozycji i głębi punktu w widoku wirtualnym ustalana jest pozycja odpowiadających mu punktów w obu widokach rzeczywistych [7].

Proces rzutowania punktów wymaga wielu operacji, a więc w algorytmie czasu rzeczywistego zdecydowano się zrealizować metodę hybrydową, gdzie operacja rzutowania dokonywana jest poprzez odczyt tablicy stworzonej podczas rzutowania głębi.

Dowolny punkt widoku wirtualnego mógł być widoczny przez dwie lub jedną kamerę rzeczywistą lub też niewidoczny w ogóle. W przypadku, gdy był on widoczny w jednej kamerze, kolor tego punktu zostanie wprost skopiowany z widoku rzeczywistego. Kiedy dany punkt był widziany przez obie rzeczywiste kamery, sprawdzana jest głębia rzutowana z obu kamer. Jeżeli głębia ta była podobna, kolor odpowiadającego punktu z obu widoków rzeczywistych jest mieszany, jeżeli nie – wybierany jest kolor tego punktu, który miał mniejszą głębię (a więc odpowiada obiektowi znajdującemu się bliżej kamery). Dzięki takiemu podejściu, nie zachodzi sytuacja, gdy kolor obiektu znajdującego się w centrum sceny mieszany jest z kolorem tła.

2.3. Filtracja widoku wirtualnego

Celem filtracji widoku wirtualnego jest wyeliminowanie dwóch efektów: nieciągłości powodowanych skończoną rozdzielczością widoków i map głębi, a także małych artefaktów będących efektem błędnie wyznaczonych map głębi.

W celu usunięcia nieciągłości w widoku wirtualnym, dla każdego punktu sprawdzane jest, czy różnica w kolorze tegoż punktu i punktu po lewej jego stronie jest wyższa, niż różnica w kolorze pomiędzy lewym i prawym sąsiadem. Jeżeli tak, przyjmuje się, iż kolor analizowanego punktu został rzutowany w sposób błędny, a następnie jest zastępowany przez kolor prawego sąsiada. W celu usunięcia zarówno pionowych, jak i poziomych nieciągłości, analogiczna operacja wykonywana jest również w kierunku pionowym.

Błędy mapy głębi mogą powodować pojawianie się małych, nieistniejących obiektów, zarówno w tle, jak i bliżej kamer. Podczas nawigacji użytkownika w scenie, obiekty bliższe przemieszczają się szybciej, niż pozostałe obiekty w scenie. Tym samym, subiektywna jakość syntezowanych widoków jest mniejsza.

W celu usunięcia takich artefaktów zaproponowano prostą, acz efektywną metodę. Dla każdego punktu widoku wirtualnego sprawdzane jest, czy punkt jest bliżej wirtualnej kamery, niż punkty po jego lewej i prawej. Jeżeli tak, kolor tego punktu zastępowany jest przez kolor prawego sąsiada. Oczywiście, tak jak w przypadku usuwania nieciągłości, analogiczna operacja przeprowadzana jest również w kierunku pionowym.

2.4. Uzupełnianie widoku wirtualnego

Na tym etapie zsyntezowany i przefiltrowany widok wirtualny wciąż zawiera obszary, dla których nie rzutowano żadnej informacji (tzw. obszary odsłonięte, niewidoczne w żadnym widoku rzeczywistym). Obszary te muszą zostać uzupełnione [1], [2], [10], [13].

W opisywanym algorytmie zaproponowano szybkie, oparte na głębi, uzupełnianie tych obszarów. Podczas uzupełniania kolor każdego z nieprzerzutowanych punktów jest kopiowany z jednego z czterech punktów: najbliższego rzutowanego punktu z lewej, prawej, z góry i z dołu. W celu wyboru kierunku kopiowania, porównywana jest głębia wszystkich sąsiadów, a także odległość pomiędzy każdym z sąsiadów i analizowanym punktem. Im mniejsza odległość i większa wartość głębi, tym wyższy priorytet ma kopiowanie z danego kierunku.

3. IMPLEMENTACJA

3.1. Implementacja algorytmu syntezy

Pomimo zastosowanych uproszczeń w zakresie filtracji i uzupełniania widoku wirtualnego, zaprezentowana metoda nie działałaby w czasie rzeczywistym dla sekwencji wysokiej rozdzielczości.

Rozważmy operację rzutowania punktów (1). Nawet jeżeli macierze $\mathbf{P_V}$ i $\mathbf{P_R^{-1}}$ zostałyby wymnożone jedynie raz dla całego widoku rzeczywistego, mnożenie $\mathbf{m} \cdot \mathbf{P_V} \cdot \mathbf{P_R^{-1}}$ wymagałoby 16 operacji mnożenia i 12 operacji dodawania dla każdego punktu w obu widokach rzeczywistych. Dla sekwencji o FullHD, a więc o rozdzielczości 1920x1080, dałoby to łącznie ponad 66 miliony mnożeń i prawie 50 milionów dodawań. Nawet przy założeniu, że zarówno operacja mnożenia, jak i dodawania wykonywana jest tylko w jednym takcie zegara i procesorze o taktowaniu 4 GHz, samo rzutowanie punktów dla jednej ramki sekwencji wielowidokowej zajęłoby 30 ms. W przypadku typowej częstotliwości wynoszącej 25 ramek na sekundę przetwarzanie czasu rzeczywistego nie może zająć więcej, niż 40 ms, a więc na wszystkie pozostałe operacje, tj. rzutowanie koloru, filtrację i uzupełnianie zostałoby jedynie 10 ms.

W celu zmniejszenia liczby operacji, mnożenie (1) zostało podzielone na niezależne części, np. mnożenie drugiej kolumny macierzy $\mathbf{P_V}$ i $\mathbf{P_R^{-1}}$ i pionowej pozycji punktu (y) wykonywane jest jedynie raz dla całego wiersza widoku wirtualnego. Również mnożenie dla każdej kolumny obrazu wirtualnego wykonywane jest raz, a następnie tablicowane.

Innym sposobem zmniejszającym czas obliczeń jest zmniejszenie liczby operacji dzielenia. Operacje tego typu są najbardziej czasochłonnymi spośród wszystkich

prostych operacji matematycznych, więc w prezentowanym algorytmie ich liczba została znacząco zmniejszona, a większość z nich została albo zastąpiona operacjami mnożenia, albo wykonywana jest wyłącznie raz dla całego widoku. Po optymalizacji, w całym opisywanym procesie syntezy widoków wirtualnych tylko jedna operacja dzielenia wykonywana jest dla każdego punktu obrazu.

Jak napisano w punkcie drugim, w celu oszczędzenia czasu nie zaimplementowano typowej syntezy wstecz. W opisywanym algorytmie zamiast powtórnego rzutowania, dla wszystkich punktów w widoku wirtualnym zapisywana jest informacja o odpowiadających im punktach w obu widokach rzeczywistych. Poprawa jakości widoku wirtualnego realizowana jest poprzez operacje dokonywane na pozycjach odpowiadających punktów, a samo rzutowanie koloru realizowane jest poprzez odczyt pozycji punktu odniesienia z tabeli.

Inną prostą modyfikacją zapewniającą szybsze wykonywanie algorytmu jest mieszanie koloru dla tych punktów widoku wirtualnego, które zostały rzutowane z obu widoków rzeczywistych i w obu tych widokach miały podobną głębię. Operacja ta jest wykonywana wyłącznie za pomocą przesunięć bitowych.

W ramach przetwarzania końcowego po rzutowaniu informacji o kolorze wykonywane są dwie operacje – filtracja i uzupełnianie. Każda z nich wymaga jedynie dwóch dodatkowych pętli po wszystkich punktach widoku wirtualnego. W przypadku filtracji widoku, w pierwszej pętli widok jest filtrowany w kierunku poziomym (w celu usunięcia pionowych nieciągłości obrazu), w drugiej zaś w kierunku pionowym, dla usunięcia pozostałych, poziomych dziur.

Zastosowana operacja uzupełniania wymaga trzech pętli. Pierwsze dwie w celu analizy obrazu, gdzie dla każdego niezsytnezowanego punktu wyznaczana jest pozycja najbliższego zsyntezowanego punktu w 4 głównych kierunkach. W pętli trzeciej dokonywane jest samo uzupełnianie obrazu. Jednakowoż, analiza widoku w celu znalezienia najbliższego lewego i górnego zsyntezowanego punktu wykonywana jest w tym samym czasie, co rzutowanie informacji o kolorze, a więc całe przetwarzanie końcowe wymaga jedynie czterech dodatkowych pętli po całym widoku wirtualnym.

3.2. Implementacja z użyciem instrukcji SIMD

W wyniku analizy przedstawionego powyżej algorytmu, oceniono że pomimo znacznej redukcji złożoności obliczeniowej, etap rzutowania głębi nadal stanowi najbardziej skomplikowany obliczeniowo krok algorytmu.

W związku z powyższym, zostały podjęte kolejne starania w celu dalszego przyśpieszenia procesu rzutowania głębi. Głównym składnikiem złożoności obliczeniowej procesu rzutowania głębi jest wyliczanie docelowej pozycji rzutowanego punku. Obliczenia te dokonywane są na liczbach zmiennoprzecinkowych (pojedynczej precyzji – typ *float*) i pomimo optymalizacji zaproponowanej w punkcie 3.1 wymagają wykonania znacznej liczby operacji mnożenia i dodawania. Niekorzystny wpływ na czas wyliczania docelowej pozycji ma fakt, że operacje na liczbach zmiennoprzecinkowych są powolne (dla współczesnych procesorów: dodawanie – 3-4 taktów, mnożenie 4-5 taktów, w zależności od implementacji).

Jedną z możliwości przyśpieszenia obliczeń jest zastosowanie instrukcji operujących na wektorach danych (SIMD) i implementacja algorytmu w taki sposób, aby wykorzystać możliwość równoległego obliczania wielu wartości. Oczywiście, nie każdy algorytm może zostać zrównoleglony z wykorzystaniem instrukcji SIMD, ale proces rzutowania głębi jest bardzo regularny (a obliczenia wykonywane identycznie dla każdego punktu) więc stanowi dobrą bazę do implementacji z użyciem instrukcji SIMD.

Implementacja wykorzystująca instrukcje wektorowe została opracowana w taki sposób, aby możliwie maksymalnie wykorzystać możliwości współczesnych procesorów z rodziny x86. W tym celu autorzy implementacji posłużyli się instrukcjami z zestawów AVX [11] i FMA [14] będących rozszerzeniami architektury x86. Instrukcje z zestawów AVX2 i FMA operują rejestrach o szerokości 256 bitów i pozwalają na jednoczesne wykonanie wybranej operacji na 8 wartościach typu *float*. Co więcej instrukcje FMA (Fused multiply–add) [9] pozwalają na wykonanie operacji „pomnóż-i-dodaj” w czasie zdecydowanie krótszym niż niezależne operacje mnożenia i dodawania. Połączona para operacji mnożenia i dodawania występuje bardzo często w procesie rzutowania głębi.

Wykorzystanie instrukcji SIMD pozwoliło na istotne przyśpieszenie kroku rzutowania głębi i skrócenie czasu potrzebnego na syntezę widoku wirtualnego.

4. WYNIKI EKSPERYMENTALNE

W celu oceny przedstawionej powyżej implementacji dokonano szeregu eksperymentów mających na celu ocenę złożoności obliczeniowej i oszacowanie maksymalnej osiągalnej liczby zsyntezowanych klatek na sekundę.

Przygotowano 3 implementacje algorytmu syntezy:

- implementacja odniesienia (R), gdzie proces rzutowania głębi został zaimplementowany zgodnie z opisem z punktu 2.1,
- implementacja zoptymalizowana (O), opisana w punkcie 3.1,
- implementacja wykorzystująca instrukcje wektorowe (V) oraz wspomniane wcześniej optymalizacje, opisana w punkcie 3.2.

Eksperymenty prowadzone były z użyciem sekwencji „Poznan_Volleyball” [6], cechującej się rozdzielczością FullHD przy 25 ramkach na sekundę (1080p25). Treścią wybranej sekwencji jest mecz siatkówki, a więc zaprezentowane wyniki są reprezentatywne dla swobodnej nawigacji dla wydarzeń sportowych.

Eksperymenty prowadzone były według dwóch scenariuszy: pierwszy zakładał że rozdzielczość obrazu syntezowanego jest równa rozdzielczości sekwencji testowej

(FullHD), drugi scenariusz zakładał że redukcję rozdzielczości obrazu syntezowanego do ¼ (rozdzielczość qHD – 960x540). Obliczenia prowadzone były na typowym komputerze wyposażonym w procesor z rdzeniem „Skylake" pracujący z zegarem 4.0 GHz. Pomiar czasu poszczególnych etapów dokonywany był z rozdzielczością <1μs [12].

Wyniki eksperymentu zostały zebrane w tabeli 1. Dla każdego ze wspominanych scenariuszy podano uśrednione czasy obliczeń poszczególnych etapów syntezy (DP – projekcja głębi, DM – łączenia głębi, VP – projekcja widoku wirtualnego, F – filtrowanie widoku wirtualnego, IP – uzupełnianie widoku wirtualnego) oraz całkowity czas syntezy widoku wirtualnego.

Tab. 1. Czas obliczeń związanych z procesem syntezy widoku wirtualnego

Implementacja	Czas syntezy (FullHD→FullHD) [ms]					
	DP	***DM***	***VP***	***F***	***IP***	***Całkowity***
R	59,95	1,03	8,24	12,81	4,18	**86,20**
O	29,86	1,04	8,26	12,86	4,17	**56,19**
V	19,74	1,05	8,22	12,50	4,2	**45,71**

Implementacja	Czas syntezy (FullHD→qHD) [ms]					
	DP	***DM***	***VP***	***F***	***IP***	***Całkowity***
R	59,73	0,25	2,11	2,77	0,83	**65,69**
O	25,02	0,23	2,20	2,78	0,84	**31,07**
V	13,56	0,22	2,21	2,69	0,82	**19,50**

Oznaczenia etapów syntezy:
DP – projekcja głębi, **DM** – łączenie głębi,
VP – projekcja widoku wirtualnego,
F – filtrowanie widoku wirtualnego,
IP – uzupełnianie widoku wirtualnego

Opisane w rozdziale 3 rozwiązania pozwoliły na znaczne skrócenie czasu rzutowania głębi, co w połączeniu z zoptymalizowaną implementacją pozostałych etapów algorytmu, pozwoliło na uzyskanie syntezy widoku wirtualnego w bardzo krótkim czasie.

W tabeli 2. Przedstawiona została analiza wydajności badanego algorytmu syntezy. W przypadku implementacji w której wszystkie etapy algorytmy wykonywane są jako jeden niepodzielny proces możliwe jest osiągnięcie wydajności ~22 obrazów/sekundę dla rozdzielczości wyjściowej FullHD i ~51 obrazów/sekundę dla rozdzielczości wyjściowej qHD.

Tab. 2. Analiza wydajności implementacji

Scenariusz	implementacja bez podziału		implementacja potokowa	
	czas przetwarzania [ms]	***wydajność [FPS]***	***czas przetwarzania [ms]***	***wydajność [FPS]***
FullHD→FullHD	45,71	21,9	45,71	50,7
FullHD→qHD	19,50	51,3	19,50	73,7

W przypadku implementacji potokowej, całkowity czas przetwarzania pozostaje niezmieniony, ale wydajność (rozumiana jako liczba zsyntezowanych obrazów na sekundę) jest ograniczona przez najwolniejszy etap potoku, którym nadal jest krok rzutowania głębi. W takim przypadku możliwe jest osiągnięcie wydajności ~50 i ~74 obrazów/sekundę, odpowiednio dla rozdzielczości wyjściowej FullHD i qHD.

5. PODSUMOWANIE

W pracy przedstawiono algorytm szybkiej syntezy widoków wirtualnych. Algorytm został zaimplementowany i przetestowany. W ramach prowadzonych eksperymentów udowodniono że nawet dla skomplikowanej sekwencji w rozdzielczości FullHD, opracowany algorytm pozwala na syntezę widoków wirtualnych w czasie rzeczywistym.

PODZIĘKOWANIA

Praca finansowana ze środków przyznanych przez Ministerstwo Nauki i Szkolnictwa Wyższego na działalność statutową polegającą na prowadzeniu badań naukowych lub prac rozwojowych oraz zadań z nimi związanych, służących rozwojowi młodych naukowców oraz uczestników studiów doktoranckich.

LITERATURA

[1] Barnes C., et. al., 2009, "Patch-Match: a randomized correspondence algorithm for structural image editing", ACM Transactions on Graphics - TOG, tom 28, nr 3.
[2] Bertalmio M., G. Sapiro, V. Caselles, 2000, "Image inpainting", SIGGRAPH 2000, Nowy Orlean, USA.
[3] Do L., G. Bravo, S. Zinger, P.H.N. de With, 2011, "Real-time free-viewpoint DIBR on GPUs for large base-line multi-view 3DTV videos", VCIP 2011, Tainan, Tajwan.
[4] Domański M., M. Gotfryd, K. Wegner, 2009, "View synthesis for multiview video transmission", IPCV, Las Vegas.
[5] Domański M., et al., 2014, "Telewizja swobodnego punktu widzenia - nowa usługa czy futurystyczna wizja?", Przegląd Telekomunikacyjny, nr 8-9, s. 734-737.
[6] Domański M., et al., 2018, "Free-viewpoint television demonstration for sports events", ISO/IEC JTC1/SC29/ WG11 MPEG2018, M41994, Gwangju, Korea.
[7] Du-Hsiu L., H. Hsueh-Ming, L. Yu-Lun, 2013, "Virtual view synthesis using backward depth warping algorithm", Picture Coding Symposium, PCS 2013, San Jose, USA.
[8] Heyden A., M. Pollefeys, 2005, "Multiple view geometry", w: "Emerging Topics in Comp. Vis.", Prentice Hall, 63-75.
[9] IEEE Standard, 2008, „IEEE Standard for Floating-Point Arithmetic", *IEEE Std 754-2008* ,1-70.
[10] Komodakis N., G. Tziritas, 2007, "Image completion using efficient belief propagation via priority scheduling and dynamic pruning", IEEE Tr. Im. Proc., tom 16, s. 2649-2661.
[11] Lomont C., 2011, „Introduction to Intel® Advanced Vector Extensions", Intel White Paper.
[12] Microsoft Developer Network Library, 2018, „Acquiring high-resolution time stamps", https://msdn.microsoft.com/en-us/library/windows/desktop/dn553408.
[13] Oh K.J., S. Yea, Y.S. Ho, 2009, "Hole filling method using depth based inpainting for view synthesis in free viewpoint television and 3-D video", PCS 2009, Chicago, USA.
[14] Quinnell E., E. E. Swartzlander, C. Lemonds, 2007, "Floating-Point Fused Multiply-Add Architectures," 41 Conf. on Sign., Syst. and Comp., Pacific Grove, pp. 331-337.
[15] Tanimoto M., et. al., 2012, "FTV for 3-D spatial communication", Proc. IEEE, tom 100, nr 4, s. 905-917.