

ZASTOSOWANIE INSTRUKCJI NEON DO PRZYSPIESZENIA WYZNACZANIA OBIEKTYWNYCH MIAR JAKOŚCI

**APPLICATION OF THE NEON INSTRUCTION SET TO ACCELERATE THE CALCULATION OF OBJECTIVE
QUALITY METRICS**

Jakub Stankowski, Patrycja Kaźmierczak, Adam Grzelka, Tomasz Grajek

Instytut Telekomunikacji Multimedialnej, Politechnika Poznańska, Poznań
{jakub.stankowski, adam.grzelka, tomasz.grajek}@put.poznan.pl

Streszczenie: Artykuł opisuje wykorzystanie zestawu instrukcji SIMD do przyspieszenia obliczeń w oprogramowaniu wyznaczającym obiektywne miary jakości (QMIV). Wykorzystano instrukcje NEON dostępne w procesorach implementujących architekturę ARM64. Opracowane oprogramowanie zostało przebadane na platformie Raspberry Pi 5. Uzyskano średnio 2,75-krotne przyspieszenie obliczeń oraz wzrost sprawności energetycznej.

Abstract: The paper describes the use of a SIMD instruction set to accelerate computations in software that calculates objective quality metrics (QMIV). NEON instructions available in processors implementing the ARM64 architecture were utilized. The developed software was tested on the Raspberry Pi 5 platform. An average 2.75-fold acceleration of computations and an increase in energy efficiency were achieved.

Słowa kluczowe: NEON, obiektywne miary jakości, optymalizacja oprogramowania, SIMD.

Keywords: NEON, objective quality metrics, SIMD, software optimization.

1. WSTĘP

W pracach nad rozwojem technik przetwarzania sygnałów wizyjnych, w tym technik kompresji (np. AVC [9], HEVC [10], VVC [11]), syntezy widoków wirtualnych [16] czy renderowania obrazów z reprezentacji w postaci anizotropowych elipsoid gaussowskich (3D Gaussian splat) [12], niezwykle istotna jest ocena jakości końcowego obrazu. Najbardziej miarodajne wyniki uzyskujemy poprzez subiektywną ocenę jakości [13]. Jednakże ocenianie subiektywne jest procesem niezwykle uciążliwym i czasochłonnym, a przez to kosztownym. Przeprowadzenie oceny subiektywnej wymaga zgromadzenia co najmniej kilkudziesięciu widzów i zrealizowania wielu sesji oglądania ocenianych materiałów. Z tego względu pełna ocena subiektywna przeprowadzana jest tylko w uzasadnionych przypadkach, np. podczas wprowadzania nowej, znormalizowanej techniki kompresji w celu porównania jej z wcześniejszą techniką.

Natomiast inaczej wygląda sytuacja, gdy w ramach prac nad rozwojem danej techniki konieczna jest weryfikacja wpływu wprowadzanej zmiany (ulepszenia,

nowego narzędzia) na sprawność działania. W celu zaoszczędzenia czasu i umożliwienia automatyzacji procesu ewaluacji stosuje się obiektywne miary jakości [13]. Dobrym przykładem jest tutaj rozwój technik kompresji wizji prowadzanych w ramach grupy ekspertów Joint Video Expert Team (JVET), gdzie każda proponowana zmiana wymaga zakodowania kilkunastu sekwencji z użyciem czterech wartości parametru kwantyzacji (QP), zebrania danych o jakości (PSNR) i przepływności, a finalnie wyznaczenia miary Bjøntegaarda [4]. Z tego powodu szybkie wyznaczanie obiektywnych miar jakości jest istotne z punktu widzenia sprawnego przeprowadzania procesu oceny jakości obrazu skompresowanego lub syntezywanego.

Od początku XXI wieku najpopularniejszą platformą sprzętową do prowadzenia obliczeń jest x86 i jej następca x86-64 (AMD64). W związku z tym większość twórców oprogramowania w pierwszej kolejności skupia się na przygotowaniu zoptymalizowanych implementacji właśnie dla platformy x86. Podobnie jest z opracowanym przez autorów tego artykułu oprogramowaniem QMIV (Quality Metrics for Immersive Video) [14][17], które jest używane w ramach prac grupy MPEG (MIV) afiliowanej przy Międzynarodowej Organizacji Normalizacyjnej ISO.

Jednak w ostatnich latach można zauważyć coraz większą popularność komputerów wykorzystujących architekturę ARM, a w szczególności ARM64 (AArch64) [18]. W szczególności są to komputery osobiste z układami SOC Apple M oraz Qualcomm Snapdragon X. Procesory implementujące architekturę ARM64 są również coraz powszechniej wykorzystywane w centrach obliczeniowych. Przykładami takich konstrukcji są linie produktów Ampere Altra, NVIDIA Grace, Amazon Graviton.

Powyższa obserwacja stała się motywacją do podjęcia prac celem dostosowania oprogramowania QMIV do architektury ARM64.

2. Oprogramowanie QMIV

Quality Metrics for Immersive Video (QMIV) to otwarto-źródłowe oprogramowanie do wyznaczania obiektywnych miar jakości, udostępniane na licencji BSD

[14][17]. Jego głównym zastosowaniem jest ocena jakości obrazów syntezowanych (MPEG MIV) a także wyników renderowania obrazów z reprezentacji w postaci anizotropowych elipsoid gaussowskich (3D Gaussian splats) [12]. W związku z powyższym QMIV zawiera implementację wielu obiektywnych miar jakości, zarówno klasycznych, jak i dedykowanych do oceny jakości wizji immersyjnej.

Miary wyznaczone w oprogramowaniu QMIV można podzielić na dwie grupy: miary oparte na pomiarze błędu średniokwadratowego i miary oparte na podobieństwie strukturalnym [6]. W przypadku miar opartych na pomiarze błędu średniokwadratowego są to: MSE (Mean Squared Error), PSNR (Peak Signal-to-Noise Ratio), WS-PSNR (Weighted Spherical PSNR). Do miar opartych na podobieństwie strukturalnym zaliczyć można SSIM (Structural Similarity Index Metric) i MS-SSIM (Multi-Scale - SSIM). Co więcej, QMIV zawiera miary zaprojektowane do oceny jakości widoków zszyntezowanych (renderowanych), które uwzględniają zniekształcenia występujące w tego typu treściach - IV-PSNR (Immersive Video - PSNR), IV-SSIM (Immersive Video - SSIM) oraz IV-MSSSIM (Immersive Video - MSSSIM).

Dotychczasowa wersja QMIV (v2.0) zawiera oprogramowanie napisane w taki sposób, żeby maksymalnie wykorzystać możliwości oferowane przez procesory z rodziny x86. W szczególności QMIV zawiera mechanizm wykrywania dostępnych rozszerzeń Single Instruction, Multiple Data (SIMD) [5] i dynamicznie wybiera procedury, które są zoptymalizowane pod kątem wykorzystania poszczególnych rozszerzeń.

Opisana powyżej funkcjonalność nie jest jednak dostępna dla procesorów zgodnych z ARM64. Rosnące grono użytkowników takich platform, a w szczególności ich uwagi i obserwacje, skłoniły autorów niniejszej publikacji do zajęcia się tym problemem.

3. Zestaw instrukcji NEON

Instrukcje operujące na wielu elementach danych (SIMD – Single Instruction, Multiple Data) [5] są znanym sposobem na efektywną i wydajną implementację operacji na obrazach. W przypadku architektury x86 programista może skorzystać z jednej z trzech grup rozszerzeń – SSE, AVX i AVX512 oferujących kolejno 128, 256 i 512 rejestrów SIMD oraz dedykowane zestawy instrukcji.

W przypadku ARM64 dostępne są dwa rozszerzenia: NEON i SVE (Scalable Vector Extension) [1]. NEON oferuje 32 rejestry o długości 128 bitów i zestaw dedykowanych instrukcji operujących na praktycznie wszystkich popularnych typach danych (8, 16, 32 i 64-bitowych liczbach całkowitych z/bez znaku oraz zmiennoprzecinkowych binary16, binary32 i binary64 [8]). Instrukcje NEON są dostępne w każdym procesorze zgodnym z ARM64. Natomiast SVE to nowszy zestaw instrukcji. Jednakże na rynku dostępnych jest niewiele procesorów zdolnych do ich wykonywania. Dodatkowo SVE nie definiuje stałej długości rejestru SIMD, a pozostawia projektantowi swobodny wybór

w zakresie 128–2048 bitów, co bardzo komplikuje potencjalną implementację.

4. Implementacja kluczowych funkcji z użyciem NEON

Głównym celem opisanych prac jest implementacja kluczowych (z punktu widzenia czasu obliczeń) procedur z wykorzystaniem instrukcji z rodziny NEON i ocena uzyskanego (potencjalnie) przyśpieszenia wyznaczania metryk jakości.

Wszystkie kluczowe z punktu widzenia wydajności funkcje zostały zaimplementowane z użyciem instrukcji z zestawu NEON. W oprogramowaniu wykorzystano „Arm C Language Extensions” (ACLE) [3] oraz wbudowane funkcje kompilatora (intrinsic functions) – są to funkcje, które przekładają się na konkretną instrukcję procesora.

W związku z tym, że każdy rdzeń zgodny z ARM64-A musi implementować instrukcje NEON, nie było potrzeby weryfikacji ich obecności na etapie uruchamiania aplikacji. Funkcje zoptymalizowane z użyciem NEON są automatycznie wybierane na etapie kompilacji, jeżeli docelową architekturą jest ARM64.

Kod źródłowy zawierający wspomniane optymalizacje jest dostępny jako QMIV v3.0 (<https://github.com/jstankowski/QMIV/releases/tag/v3.0>)

5. Weryfikacja eksperymentalna

Weryfikację eksperymentalną przeprowadzono z wykorzystaniem platformy z procesorem ARM – Raspberry Pi 5 [15]. Wspomniany komputer jednopłytkowy zbudowany został na bazie układu Broadcom BCM2712 zawierającego cztery rdzenie ARM Cortex-A76.

Pomiaru czasu dokonano przez odczyt rejestru CNTVCT_EL0 (Counter-timer Virtual Count) udostępniającego wartość monotonicznego zegara taktowanego częstotliwością 54 MHz.

Dodatkowo dokonano pomiaru zużycia energii poprzez odczyt wartości udostępnianych przez wbudowany układ PMIC (DA9091), a w szczególności wartości napięcia zasilającego rdzenie Cortex-A76 oraz natężenia pobieranego przez nie prądu.

5.1. Pomiar wydajności poszczególnych funkcji

Pierwszym z eksperymentów mających na celu weryfikację wydajności zoptymalizowanej implementacji było przeprowadzenie syntetycznych testów dla poszczególnych procedur. W tym celu użyto pseudolosowych danych testowych (przeważnie obrazów o rozmiarze 2048×2048 pikseli), a czas wykonywania operacji przez każdą z funkcji zebrano dla 32 przebiegów i uśredniono. Czas obliczeń został przeliczony na wydajność w milionach pikseli na sekundę (Mpix/s).

Wyniki dla ogólnych operacji, takich jak: konwersja przestrzeni barw, konwersja typu danych, pod/nad próbkowanie, konwersja ułożenia danych (AoS/SoA) itp., przedstawione zostały w Tabeli 1. Uzyskane

przyspieszenie waha się w zakresie od 1,09× do 4,95×, przy średnim przyspieszeniu rzędu 2,56×.

Tab. 1. Wyniki uzyskane dla ogólnych operacji na pikselach.

Operacja	Wydajność dla danej implementacji [Mpix/s]		Przyspieszenie
	Skalarna	NEON	
RGB→YCbCr	124	180	1,45×
YCbCr→RGB	130	188	1,44×
uint8→uint16	885	3723	4,21×
uint16→uint8	1128	2620	2,32×
UpsampleHV	486	528	1,09×
DownsampleHV	438	873	1,99×
CvtUpsampleHV	498	559	1,12×
CvtDownsampleHV	418	1375	3,29×
UpsampleH	894	1181	1,32×
DownsampleH	622	1578	2,54×
CvtUpsampleH	745	1273	1,71×
CvtDownsampleH	566	2062	3,64×
SoA3→AoS4	379	400	1,05×
AoS4→SoA3	133	204	1,53×
CheckIfInRange	1132	8034	7,09×
CountNonZero	1182	5850	4,95×
CompareEqual	1115	3117	2,80×
Średnia			2,56×

W Tabeli 2 podsumowano wyniki uzyskane dla operacji bezpośrednio powiązanych z wyznaczaniem miar jakości. W tym przypadku uzyskano średnie przyspieszenie rzędu 3,02×. Dla funkcji SSD (Sum of Squared Differences), która jest składnikiem miar MSE, PSNR i WSPSNR, uzyskano istotne przyspieszenie rzędu 5,67×.

Tab. 2. Wyniki uzyskane dla operacji bezpośrednio powiązanych z wyznaczaniem miar jakości.

Operacja	Wydajność dla danej implementacji [Mpix/s]		Przyspieszenie
	Skalarna	NEON	
Pomiar zniekształcenia			
SD	1058,99	3197,01	3,02×
SAD	910,92	3163,52	3,47×
SSD	569,42	3226,88	5,67×
Kompensacja przesunięcia odpowiadających sobie pikseli (SCP)			
DistAsymmetric	4,25	5,37	1,26×
GenShftComp	4,30	5,62	1,31×
Podobieństwo strukturalne (Structural Similarity)			
Avg8x8	4,31	15,68	3,64×
Avg11x11	2,33	3,99	1,71×
Avg16x16	1,13	4,63	4,10×
Średnia			3,02×

5.2. Pomiar czas wyznaczania miar jakości

Kolejnym krokiem weryfikacji był pomiar czasu potrzebnego na wyznaczenie poszczególnych miar jakości. W tym przypadku wykorzystano bazę 10 sekwencji testowych o rozdzielczości FullHD (1920×1080), zaś czasy zebrane dla każdej ramki

uśredniono. Wyniki przedstawiono w Tabeli 3. Przyspieszenie uzyskane dla poszczególnych miar jakości waha się w zakresie od 1,48× (IV-PSNR) do 3,83× (WS-PSNR), przy średniej wynoszącej 2,75×.

Tab. 3. Wyniki uzyskane dla poszczególnych miar jakości.

Miara	Wydajność dla danej implementacji [ms/ramkę]		Przyspieszenie
	Skalarna	NEON	
MSE	11,26	3,40	3,31×
PSNR	11,24	2,98	3,77×
WS-PSNR	11,23	2,94	3,82×
IV-PSNR	480,04	325,27	1,48×
SSIM	89,05	28,00	3,18×
MS-SSIM	126,15	41,52	3,04×
IV-SSIM	672,08	317,17	2,12×
IV-MS-SSIM	746,32	344,54	2,17×
Średnia			2,75×

5.3. Analiza uzyskanych wyników

Jak pokazano w poprzednich punktach, uzyskane przyspieszenie nie jest szczególnie wysokie (przy 128-bitowych rejestrach, w których przetwarza się 16-bitowe dane, można by oczekiwać nawet 8-krotnego przyspieszenia). Zdaniem autorów, taki wynik może być spowodowany czynnikami związanymi z architekturą rdzenia Cortex-A76 i faktem, że jego docelowym zastosowaniem były urządzenia przenośne. Cortex-A76 ma tylko dwie jednostki wykonawcze SIMD, z tego jedna może realizować tylko „proste operacje” (np. operacje logiczne, dodawanie, odejmowanie) [2]. Dodatkowo wiele operacji trwa dłużej niż dla części skalarnej – przykładowo wynik dodawania w części skalarnej dostępny jest po jednym takcie, zaś dodawanie SIMD zajmuje dwa takty zegara [2]. Co więcej, układ BCM2712 zawiera pojedynczy kontroler pamięci LPDDR4X-4267 o szerokości szyny równej 32 bity. Ogranicza to szybkość transmisji danych z/do pamięci do ~17,0 GB/s, co też jest sporym ograniczeniem dla mniej skomplikowanych obliczeniowo operacji.

5.4. Sprawność energetyczna

Oprócz wpływu implementacji SIMD na wydajność przeprowadzono również ocenę wpływu na sprawność energetyczną. W tym celu, w czasie obliczeń regularnie pobierano z układu PMIC informacje o poborze energii przez rdzenie Cortex-A76. W przypadku kodu skalarnego, rdzenie procesora pobierały średnio 4,83 W. W czasie wykonywania kodu wykorzystującego instrukcje NEON pobór mocy wynosił średnio 5,28 W. Oznacza to, że 2,75-krotny wzrost wydajności okupiony został 1,09-krotnym wzrostem poboru mocy. Wyraźnie widać więc, że wykorzystanie operacji SIMD zauważalnie podnosi sprawność energetyczną obliczeń.

6. Podsumowanie

W niniejszym artykule przedstawiono oprogramowanie QMIV (v3.0). Zostało ono zoptymalizowane pod kątem architektury ARM64 z wykorzystaniem zestawu instrukcji SIMD NEON.

Przeprowadzone eksperymenty na platformie Raspberry Pi 5 wykazały, że zastosowanie instrukcji SIMD pozwala na uzyskanie średnio 2,75-krotnego przyspieszenia procesu wyznaczania obiektywnych miar jakości obrazu. Natomiast w przypadku pojedynczych, niskopoziomowych operacji uzyskano nawet 7-krotne przyspieszenie prędkości obliczeń.

Ważną obserwacją z przeprowadzonych badań jest również poprawa efektywności energetycznej. Blisko trzykrotny wzrost wydajności okupiony został zaledwie 9-procentowym wzrostem poboru mocy. Wraz ze wzrostem popularności platform wykorzystujących architekturę ARM64 jest to kluczowy argument dla stosowania optymalizacji niskopoziomowych.

PODZIĘKOWANIA

Praca finansowana ze środków przyznanych przez Ministerstwo Nauki i Szkolnictwa Wyższego.

LITERATURA

- [1] Arm Limited. 2021. "Arm® Architecture Reference Manual for A-profile architecture".
- [2] Arm Limited. 2021. "Arm Cortex-A76 Software Optimization Guide". Version 10.0, PJDOC-466751330-7215.
<https://developer.arm.com/documentation/PJDOC-466751330-7215/r4p1/>
- [3] Arm Limited. 2024. "Arm C Language Extensions (ACLE)". <https://github.com/ARM-software/acle>.
- [4] Bjøntegaard G. 2001. „Calculation of average PSNR differences between RD curves”, *ITU-T SG16/Q6*, dokument VCEG-M33.
- [5] Demikhovsky, E. 2013. "Intel® AVX-512 Architecture. Comprehensive vector extension for HPC and enterprise", *LLVM Developers' Meeting*.
- [6] Dziembowski A., Mieloch D., Stankowski J., Grzelka A. 2022. „IV-PSNR—The Objective Quality Metric for Immersive Video Applications”. *IEEE Transactions on Circuits and Systems for Video Technology*. 32 (11): 7575-7591.
- [7] Dziembowski A., Nowak W., Stankowski J. 2024. „IV-SSIM—The Structural Similarity Metric for Immersive Video”. *Applied Sciences*. 14 (7090).
- [8] IEEE Computer Society .2019. „IEEE Standard for Floating-Point Arithmetic”. IEEE STD 754-2019.
- [9] ISO/IEC. 2004. „Information technology — Coding of audio-visual objects — Part 10: Advanced Video Coding”. *ISO/IEC IS 14496-10, także jako ITU-T Rec. H.264*.
- [10] ISO/IEC. 2013. „Information technology — High efficiency coding and media delivery in heterogeneous environments — Part 2: High Efficiency Video Coding”. *ISO/IEC IS 23008-2, także jako ITU-T Rec. H.265*.
- [11] ISO/IEC. 2024. „Information technology — Coded representation of immersive media — Part 3: Versatile Video Coding”. *ISO/IEC IS 23090-3, także jako ITU-T Rec. H.266*.
- [12] Kerbl B., Kopanas G., Leimkühler T., Drettakis G. 2023. „3D Gaussian Splatting for Real-Time Radiance Field Rendering”. *ACM Transactions on Graphics*. 42 (4).
- [13] Ohm J. -R., Sullivan G. J., Schwarz H., Tan T.K., Wiegand T. 2012. „Comparison of the Coding Efficiency of Video Coding Standards—Including High Efficiency Video Coding (HEVC)”. *IEEE Transactions on Circuits and Systems for Video Technology*. 22 (12): 1669-1684.
- [14] QMIV (Quality Metrics for Immersive Video) software. <https://github.com/jstankowski/QMIV>.
- [15] Raspberry Pi Ltd. 2023. "Raspberry Pi 5 Product Brief".
<https://datasheets.raspberrypi.com/rpi5/raspberry-pi-5-product-brief.pdf>.
- [16] Senoh T., Tetsutani N., Yasuda H. 2018. „Depth estimation and view synthesis for immersive media”. *International Conference on 3D Immersion (IC3D)*.
- [17] Stankowski J., Dziembowski A. 2025. „IV-PSNR: Software for immersive video objective quality evaluation”. *SoftwareX*. 33 (102487).
- [18] Whaley D. 2021. „64-bit computing for mobile”. *ARM*.