KONFERENCJA RADIOKOMUNIKACJI
I TELEINFORMATYKI
KRiT 2025

ANALIZA MOŻLIWOŚCI ZRÓWNOLEGLENIA OBLICZEŃ W KODERZE I DEKODERZE JPEG

ANALYSIS OF THE POSSIBILITY OF PARALLELIZING COMPUTATIONS IN JPEG ENCODER AND DECODER

Jakub Stankowski, Adrian Śliwiński, Adam Grzelka, Tomasz Grajek

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

Streszczenie: **Artykuł przedstawia metodę przyspieszenia kodera i dekodera JPEG poprzez zrównoleglenie przetwarzania oraz opcjonalne zastosowanie znaczników restartu. Przeanalizowano potencjalne problemy i możliwe podejścia, zaprezentowano szczegóły implementacji z wykorzystaniem podziału na niezależnie kodowane fragmenty oraz ocenę wydajności, która wykazała znaczące skrócenie czasu kodowania i dekodowania. Przyspieszenie nie miało wpływu na sprawność kompresji. Proponowane podejście pozwala na przyspieszenie kodowania z użyciem współczesnych wielordzeniowych procesorów.**

Abstract: **The paper presents a method for accelerating the JPEG encoder and decoder through parallel processing and the optional use of restart markers. Potential issues and possible approaches were analyzed, implementation details utilizing division into independently encoded segments were presented, and performance evaluation demonstrated significant reductions in encoding and decoding times. The acceleration did not affect compression efficiency. The proposed approach enables faster encoding using modern multicore processors.**

Słowa kluczowe: **kompresja obrazu, JPEG, kodowanie równoległe, znaczniki restartu.**

Keywords: image compression, JPEG, parallel encoding, restart markers.

1. WSTĘP

Technika JPEG [6, 14], pomimo że istnieje na rynku od ponad 30 lat, jest nadal bardzo popularnym kodekiem do kompresji obrazów statycznych. Największą jej zaletą jest ogromna popularność i kompatybilność – praktycznie każde urządzenie i oprogramowanie radzi sobie z tym formatem kompresji [15].

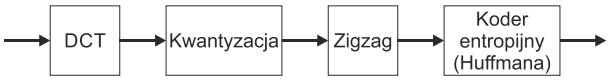
Technika JPEG cechuje się niską złożonością obliczeniową w porównaniu do nowszych technik takich jak HEVC Intra (HEIF) [9].Warto jednak zauważyć, że istnieją urządzenia, które operują na obrazach o rozmiarze >50Mpix, takie jak cyfrowe lustrzanki (DLSR) i nadal stosują technikę JPEG. Kodowanie i dekodowanie tak dużego obrazu z użyciem jednego rdzenia nadal jest czasochłonne.

Technika JPEG powstawała w czasach kiedy nikt nie brał pod uwagę zrównoleglenia obliczeń z podziałem na wiele rdzeni. W literaturze można znaleźć prace związane z problemem zrównoleglenia obliczeń w technice JPEG [2, 11, 12], jednak są to analizy teoretyczne lub próby implementacji nie całkowicie zgodne z normą.

Współczesnym trendem jest wyposażanie komputerów (i urządzeń mobilnych) w wielordzeniowe procesowy posiadające często kilkanaście (lub więcej) jednostek. W związku z tym istotna jest analiza czy możliwe jest wykorzystanie owych wielordzeniowych procesorów do przyspieszenia kodowania i dekodowania.

2. ZRÓWNOLEGLENIE OBLICZEŃ W KODEKU JPEG

Typowy koder JPEG składa się z czterech najistotniejszych etapów przetwarzania: transformacji DCT, kwantyzacji, uporządkowania zygzakowatego i kodera entropijnego (Rys. 1.). Twórcy techniki JPEG przewidzieli możliwość zastosowania dwóch różnych koderów entropijnych – kodera Huffmana i arytmetycznego. W praktyce jednak można przyjąć, że wariant z koderem arytmetycznym nie jest często stosowany – głównie ze względu na istniejące przez wiele lat utrudnienia prawne i licencyjne. Lista patentów dotyczących kodowania arytmetycznego w JPEG jest zawarta w normie [6] (aneks L).

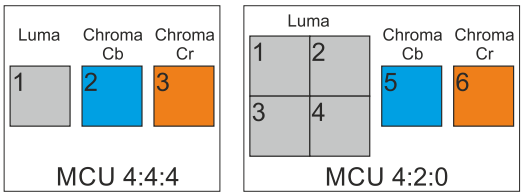


Rys. 1. Schemat blokowy kodera JPEG

W przypadku trzech pierwszych etapów przetwarzania zrównoleglenie obliczeń jest relatywnie łatwe. Zarówno transformacja DCT, kwantowanie, jak i zmiana kolejności współczynników odbywa się w blokach o rozmiarze 8x8. Te bloki mogą być przetwarzane niezależnie od siebie, co stanowi ogromne ułatwienie w przypadku implementacji wielowątkowej. Przykładowym (i stosowanym przez autorów) podejściem jest podzielenie obrazu na kolejne wiersze bloków 8x8 i potraktowanie każdego wiersza jako osobnego zadania obliczeniowego.

Zdecydowanie gorzej wygląda sytuacja związana z próbą zrównoleglenia kodowania Huffmana.

JPEG definiuje strukturę nazywaną „minimalną jednostką kodowania” (Minimum Coded Unit – MCU) [6]. Rozmiar i budowa jednostki MTU zależy od wybranego próbkowania chrominancji. Przykładowo, dla formatu 4:4:4 jednostka MCU ma rozmiar 8x8 i zawiera po jednym bloku luminancji i obu chrominancji. Z kolei dla popularnego formatu 4:2:0 jednostka MCU ma rozmiar 16x16 i zawiera cztery bloki luminancji i po jednym bloku dla obu chrominancji (Rys. 2.).

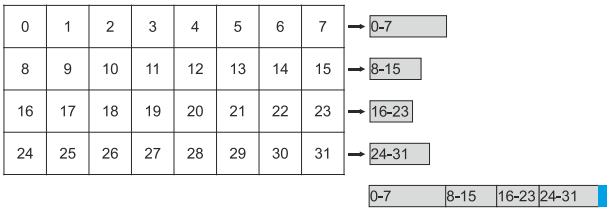


Rys. 2. Typowe jednostki MCU (Minimum Coded Unit) w JPEG dla obrazu kodowanego z próbkowaniem chromy 4:4:4 i 4:2:0

Kodowanie entropijne realizowane jest przez przetwarzanie kolejnych jednostek MCU zaczynając od składowej luminancji. Kolejność przetwarzania bloków została zaznaczona na Rys. 2. Co więcej, współczynniki DC (składowa stała dla każdego bloku) są kodowane predykcyjnie, w ramach danego komponentu. Wysyłana jest różnica pomiędzy obecnie kodowanym współczynnikiem DC a współczynnikiem poprzednio zakodowanego bloku tego samego komponentu. Powoduje to istnienie pewnej zależności pomiędzy kolejnymi blokami.

2.1. Zrównoleglenie kodowania Huffmana w koderze JPEG

Pomimo opisanych wcześniej niedogodności, możliwe jest zrównoleglenie kodowanie entropijnego przez zakodowanie wielu fragmentów obrazu, a następnie scalenie ich w jednym buforze. Ilustracja takiego rozwiązania została przedstawiona na Rys. 3. Podział obrazu na fragmenty powinien odbywać się na granicach MCU. Stosowane przez autorów podejście zakłada podział obrazu na wiersze jednostek MCU, co w przypadku próbkowania 4:2:0 odpowiada wycinkom obrazu o wysokości 16 pikseli.



Rys. 3. Schemat działania równoległego kodera JPEG i łączenia strumieni powstałych w wyniku kodowania poszczególnych fragmentów. Niebieskim kolorem zaznaczono wypełnienie do pełnych bajtów.

Wartości współczynników DC używane do predykcji można pobrać z poprzedzającego bloku (po wcześniejszym wyliczeniu jego położenia).

Niestety sama operacja łączenia zakodowanych fragmentów strumieni jest dość uciążliwa i czasochłonna. Główną przyczyną jest to, że kolejne fragmenty strumieni mają długość, która nie jest liczona w pełnych bajtach, co skutkuje koniecznością przeprowadzenia czasochłonnej manipulacji na pojedynczych bitach przy operacji łączenia. Po operacji łączenia konieczne jest zrealizowanie kolejnego kroku – analizy i unikania emulacji znaczników (markerów). Oba wspomniane kroki muszą być zrealizowane szeregowo przez jeden wątek, a to, zgodnie z prawem Amdahla [4], zmniejsza potencjalne zyski ze zrównoleglenia.

3. ZASTOSOWANIE ZNACZNIKÓW RESTARTU

Wiele technik kompresji zawierały w sobie rozwiązania poprawiające odporność na błędy transmisji. Przykładem jest tu H.263 Annex H („Forward Error Correction for coded video signal”) [7]. Popularnym rozwiązaniem jest również podział obrazu na niezależne fragmenty – plastry (slice) stosowane między innymi w MPEG-4 AVC/H.264 [8], HEVC/H.265 [9] czy VVC H.266 [10].

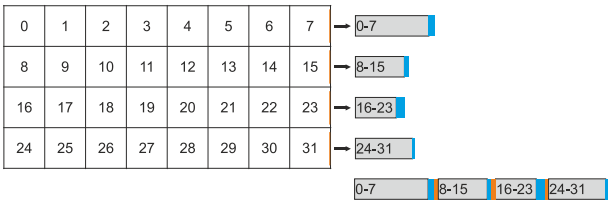
Technika JPEG również przewiduje takie rozwiązanie, a nazywane jest ono znacznikami restartu (Restart Markers – RST). RST zostały przewidziane jako sposób poprawy odporności na błędy transmisji. Pozwalają wznowić dekodowanie od kolejnego znacznika i uniknąć sytuacji gdzie jeden błąd powoduje utratę całego obrazu.

Zgodnie z normą [6] możliwe jest zdefiniowane wartości Restart Interval (RI). W takim przypadku RST jest wstawiany do strumienia co RI jednostek MTU. Co więcej, znaczniki restartu są wyrównane do pełnych bajtów, więc w razie potrzeby są dodatkowo poprzedzane odpowiednią liczbą bitów (jedynek).

Ze względu na analogię do funkcjonalności zdefiniowanej w MPEG-4 AVC/H.264 [8], poszczególne fragmenty obrazu oddzielone RST będą nazywane plastrami. W technice JPEG plastry są kodowane całkowicie niezależnie – RST zrywa predykcję DC i wymusza dodanie wypełnienia do pełnych bajtów do poprzedzającego plastra.

Obie powyższe cechy powodują, że możliwe jest zastosowanie RST niezgodnie z intencją twórców – do zrównoleglenia obliczeń.

Zachowując ten sam sposób podziału jak opisany w podrozdziale 2.1. autorzy przyjęli wartość RI równą liczbie MCU w jednym wierszu (Rys. 4.).

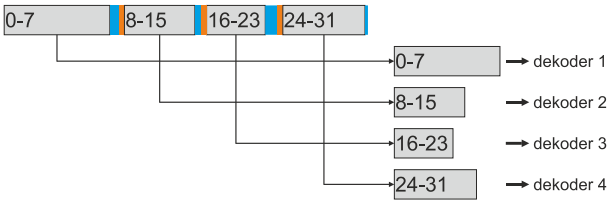


Rys. 4. – Schemat działania równoległego kodera JPEG wykorzystującego znaczniki restartu. Niebieskim kolorem zaznaczono wypełnienie do pełnych bajtów a pomarańczowym – RST.

Zastosowanie podejścia ze znacznikami restartu umożliwia zrównoleglenie większej liczby operacji. Możliwe jest niezależne kodowanie poszczególnych plastrów oraz analiza i unikanie emulacji znaczników. Natomiast samo łączenie strumieni jest prostą i szybką operacją kopiowania bloków pamięci.

Oczywiście opisane powyżej podejście ma pewne wady. Największą z nich jest zmniejszenie sprawności kompresji. Owa utrata sprawności ma dwie przyczyny. Pierwszą jest zerwanie predykcji DC na granicy plastrów, skutkujące większą liczbą bitów zużytych na zakodowanie tego elementu składni. Drugim jest konieczność wysłania samych znaczników (i ewentualnie wyrównania do pełnych bajtów) co przekłada się na dodatkowy wzrost rozmiaru strumienia (bądź pliku).

Z drugiej strony nie można wspomnieć o jeszcze jednej zalecie użycie RST. Podział obrazu na plastry umożliwia jego równoległe dekodowanie. W takiej sytuacji dekoder może w pierwszej kolejności sparsować strumień/plik i wyszukać w nim wszystkie znaczniki. Następnie można wydzielić poszczególne plastry i każdy z nich dekodować niezależnie (Rys. 5.).

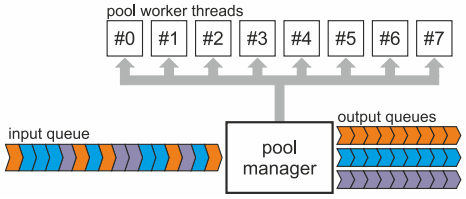


Rys. 5. Schemat działania zrównoleglonego dekodera przetwarzającego strumień ze znacznikami restartu

4. IMPLEMENTACJA

W ramach weryfikacji przedstawionych wcześniej koncepcji przygotowano implementację wielowątkowego kodeka JPEG. Kodek powstał jako rozwinięcie opracowanego przez autorów oprogramowania JOptEnc (https://github.com/jstankowski/joptenc).

Opracowane oprogramowanie pozwala na równoległe kodowanie zarówno w wariancie z użyciem znaczników restartu, jak i bez ich stosowania. Dodatkowo przygotowano dekoder, który pozwala na równoległe dekodowanie obrazów zawierających znaczniki restartu.



Rys. 6. Idea wzorca projektowego „pula wątków” (thread pool)

Oba moduły (koder i dekoder) wykorzystują własną implementację puli wątków (thread pool) w celu minimalizacji narzutu związanego z przekazywaniem zadań do dodatkowych wątków (Rys. 6.). Dokładny opis tej techniki można znaleźć w [13].

5. WERYFIKACJA EKSPERYMENTALNA

5.1. Metodologia

Eksperyment został przeprowadzony na bazie testowych sekwencji Netflix [1]. Zawiera ona 9 zróżnicowanych sekwencji w formacie FullHD. Przygotowane w ramach tej pracy oprogramowanie kodera zostało uruchomione dla trzech punktów pracy, związanych z siłą stosowanej kwantyzacji, odpowiednio dla $Q_L \in \{20, 25, 30, 35\}$, $Q_M \in \{50, 55, 60, 65\}$, $Q_H \in \{80, 85, 90, 95\}$. Warto zauważyć, że w technice JPEG parametr Q oznacza „jakość”. Wyższa wartość Q wiąże się ze słabszą kwantyzacją i większym strumieniem.

Test był przeprowadzany na komputerach z procesorem AMD Ryzen R9-7950X, który posiada 16 rdzeni. Każde kodowanie było realizowane niezależnie, czyli jeden eksperyment był realizowany w danym momencie. Pomiar czasu był wykonany z wykorzystaniem odczytu licznika TSC procesora, czyli instrukcji RDTSCP [5]. Eksperyment zrealizowano w trzech trybach pracy:

- Tryb A – tryb szeregowy, traktowany jako odniesienie do wyliczenia przyśpieszenia uzyskanego w wyniku zrównoleglenia.
- Tryb B – wykorzystujący zrównoleglenia, ale bez użycia znaczników restartu.
- Tryb C – wariant wielowątkowy, dodatkowo wykorzystujący znaczniki restartu.

W obu zrównoleglonych wariantach zastosowano podział obrazu na wiersze jednostek MCU, co w przypadku obrazów FullHD z próbkowaniem chromy 4:2:0 oznacza podział na 68 jednostek pokrywających obszar 1920x16 pikseli.

Warto zaznaczyć, że wersja wielowątkowa kodera i dekodera JPEG działa identycznie jak wariant jednowątkowy – w tym sensie, że zakodowany strumień i zdekodowany obraz są binarnie identyczne. W związku z tym samo zrównoleglenie nie wpływa na jakość zakodowanego obrazu, rozmiar strumienia czy sprawność kompresji.

5.2. Wpływ stosowania znaczników restartu

Pierwszym krokiem weryfikacji przedstawionej koncepcji jest określenie, jak samo użycie znaczników restartu wpływa na sprawność kompresji. Jak wynika z Tabeli 1, wpływ ten jest zauważalny, ale niezbyt istotny. W najgorszym przypadku – dla bardzo niskiej jakości obrazu – wiąże się z 2% wzrostem rozmiaru skompresowanego pliku. Jednak w bardziej realistycznym przypadku – dla wysokiej jakości – wzrost rozmiaru obrazu to tylko 0,6%. Wpływ użycia RST na czas kodowania jest minimalny i wynosi ~31µs przy czasie kodowania jednego obrazu wahającym się pomiędzy 5 a 10 ms na ramkę.

Tab. 1. Wpływ użycia RST na rozmiar zakodowanego obrazu dla RST wstawianego co jeden wiersz MCU

Punkt pracy	Średni wzrost rozmiaru jednego obrazu [B]	Średni względny wzrost rozmiaru jednego obrazu [%]	Średni wzrost czasu kodowania jednego obrazu [µs]
Q_L	179,13	1,950	
Q_M	183,34	1,252	31,24
Q_H	192,31	0,594	

5.3. Przyśpieszenie uzyskane przez zrównoleglenie kodeka

Wyniki zawierające ocenę wydajności implementacji wielowątkowej przedstawiono w Tabeli 2. Zauważalne jest, że przyspieszenie jest większe dla wyższego zakresu jakości. Jest to spodziewane zachowanie, gdyż złożoność obliczeniowa kodera i dekodera entropijnego jest bezpośrednio skorelowana z liczbą bitów na obraz. Można też zauważyć, że zastosowanie znaczników restartu pozwala na znaczne zwiększenie zysków wynikających z przetwarzania wielowątkowego.

Co istotne, zastosowanie znaczników restartu pozwala również na przyspieszenie procesu dekodowania, co nie jest możliwe w przypadku obrazów zakodowanych bez użycia RST.

Tab. 2. Przyśpieszenie kodowania i dekodowania uzyskane dla implementacji wielowątkowej

Punkt pracy	Przyśpieszenie kodera (tryb B)	Przyśpieszenie kodera + RST (tryb C)	Przyśpieszenie kodera RST (tryb C)
Q_L	3,90×	4,87×	2,94×
Q_M	3,78×	4,93×	3,03×
Q_H	3,42×	5,24×	3,42×

W Tabeli 3. zebrano wyniki dla zrównoleglonego kodera JPEG wykorzystującego technikę RDOQ, która jest zaimplementowana w JOptEnc. Warto zauważyć, że użycie RDOQ większa sprawność kompresji o około 10%, ale kosztem 10, do 20-krotnego wydłużenia czasu kodowania [3]. W tym przypadku użycie wielowątkowej implementacji pozwala na częściowe skompensowanie wzrostu złożoności obliczeniowej spowodowanego przez użycie RDOQ.

Tab. 3. Przyśpieszenie kodera używającego RDOQ uzyskane dla implementacji wielowątkowej

Punkt pracy	Przyśpieszenie kodera (tryb B)	Przyśpieszenie kodera + RST (tryb C)
Q_L	6,72×	6,80×
Q_M	7,56×	7,74×
Q_H	9,02×	9,31×

Jak pokazano w powyższych tabelach, możliwe jest znaczne przyspieszenie działania kodeka JPEG poprzez podział obliczeń na wiele wątków i wykorzystanie możliwości współczesnych procesorów wielordzeniowych. W skrajnym przypadku uzyskana prawie dziesięciokrotne przyśpieszenie obliczeń.

6. Podsumowanie

W artykule przedstawiono koncepcję zrównoleglenia obliczeń w kodeku JPEG. Przeanalizowano dwa podejścia do zrównoleglenia – z wykorzystaniem znaczników restartu oraz bez ich użycia. W celu weryfikacji opracowanej koncepcji przygotowano wielowątkową implementację kodeka JPEG i przeprowadzono eksperymenty mające na celu ocenę potencjalnego przyśpieszenia kodowania i dekodowania.

Wyniki eksperymentalne wskazują na to, że nawet technika kompresji, która nie powstawała z myślą o ułatwieniu zrównoleglenia obliczeń pozostawia twórcom kodeka pewne możliwości w kwestii zrównoleglenia. W opisywanym przypadku uzyskano ponad pięciokrotne przyśpieszenie obliczeń dla prostego kodera JPEG oraz prawie dziesięciokrotne dla kodera wykorzystującego technikę RDOQ. Pokazano również potencjał zrównoleglenia dekodera JPEG.

LITERATURA

[1] Bampis C. G., Li Z., Moorthy A. K., Katsavounidis I., Aaron A. Bovik A.C. 2016. „LIVE Netflix Video Quality of Experience Database„. Online: http://live.ece.utexas.edu/research/LIVE_NFLXStudy/index.html.

[2] Castells-Rufas D., Joven J., Carrabina J. 2010. „Scalability of a Parallel JPEG Encoder on Shared Memory Architectures”. *39th International Conference on Parallel Processing*. 502-507.

[3] Grajek T., Stankowski J. 2022. „Rate-Distortion Optimized Quantization in Motion JPEG”. *30th International Conference in Central Europe on Computer Graphics, Visualization and Computer Vision (WSCG)*.

[4] Gustafson, J.L. 2011. Amdahl’s Law. In: Padua, D. (eds) *Encyclopedia of Parallel Computing*. Springer.

[5] Intel. 2015. „Intel 64 and IA-32 Achitectures Software Deceloper's Manual„. 2B : 4-302.

[6] ISO/ITU. 1992. „Information Technology — Digital Compression and Coding of Continuous-Tone Still Images — Requirements and Guidelines”, ISO 10918-1, także jako ITU T.81.

[7] ITU-T Rec. H.263:2005. „Video coding for low bit rate communication”. International Telecommunication Union.

[8] ITU-T Rec. H.264:2021. „Advanced video coding for generic audiovisual services„. International Telecommunication Union.

[9] ITU-T Rec. H.265:2023. „High efficiency video coding„. International Telecommunication Union.

[10] ITU-T Rec. H.266:2023. „Versatile video coding„. International Telecommunication Union.

[11] Monnes P., Furht B. 1994. „Parallel JPEG algorithms for still image compression” *SOUTHEASTCON '94*. 375-379.

[12] Shin H., Lee J. 2024. „Hardware Multi-Threaded System for High-Performance JPEG Decoding”. Journal of Signal Processing Systems. 96 : 67–79.

[13] Stankowski J., Grzelka A., Mieloch D., Wegner K. 2018. „Processing Pipeline for Real-Time Remote Delivery of Virtual View in FTV Systems”. *2018 International Conference on Signals and Electronic Systems (ICSES)*. 118-123

[14] Wallace G.K. 1992. „The JPEG still picture compression standard”, *IEEE Transactions on Consumer Electronics*, 38 (1) : xviii – xxxiv.

[15] W3Techs. 2025. „Usage statistics of JPEG for websites”, https://w3techs.com/technologies/details/im-jpeg.