AUTOMATYKA • 2008 • Tom 12 • Zeszyt 1

Grzegorz Ulacha*

Wykorzystanie adaptacyjnego kodera arytmetycznego z przełączaniem kontekstów do bezstratnej kompresji obrazów**

1. Wstęp

Można wyróżnić dwie metody kompresji obrazów: stratną oraz bezstratną i właśnie temu drugiemu typowi poświęcono niniejszy artykuł. W przypadku stosowania nowoczesnych metod kompresji wykorzystuje się zwykle dwa etapy: modelowanie danych, a następnie kompresję jedną z wydajnych metod entropijnych, wśród których najefektywniejsze to kodowanie arytmetyczne i kodowanie Huffmana [19]. Głównym celem wcześniejszych badań [23] było opracowanie efektywnego algorytmu modelowania obrazów umożliwiającego uzyskanie wysokiej efektywności kompresji. Etapem badań, który zostanie tu szerzej zaprezentowany, jest analiza efektywności adaptacyjnego kodera arytmetycznego z przełączaniem kontekstów. Został on wykorzystany celem uzyskania wysokiego stopnia kompresji, przy zachowaniu akceptowalnej złożoności implementacyjnej.

1.1. Zastosowania bezstratnej kompresji obrazów

Dzięki możliwości kompresji obrazów obniżają się znacznie koszty związane zarówno z ich przechowywaniem, jak i transmisją. Stopień kompresji zależy od typów kodowanych obrazów, ale najistotniejszym parametrem jest stopień zniekształceń wprowadzanych na etapie kodowania. Istnieje wiele zastosowań, w których nie możemy sobie pozwolić na utratę jakości obrazu, np. w przypadku obrazów medycznych oraz przy archiwizacji ważnych dokumentów, map, zdjęć cyfrowych, stanowiących wersję pierwotną przeznaczoną do dalszej obróbki. W takich sytuacjach stosuje się kodowanie bezstratne.

Oprócz obrazów statycznych kompresję wykorzystuje się też do kodowania sekwencji wideo. I właśnie tu, w dobie w pełni cyfrowej struktury obróbki sygnału wideo, istotnym

* Instytut Architektury Komputerów i Telekomunikacji, Wydział Informatyki, Politechnika Szczecińska

** Praca finansowana ze środków budżetowych na naukę w ramach grantu przyznanego na lata 2007–2010

problemem są wysokie wymagania pamięciowe związane z przechowywaniem danych cyfrowych. Przykładowo zapis 60 minut filmu jakości telewizyjnego standardu PAL (25 klatek na sekundę o rozdzielczości 720×576 pikseli, co daje 9,89 Mpikseli/s, gdy każdą ze składowych RGB traktujemy jako liczbę ośmiobitową) wymaga 104,28 GB miejsca na dysku (lub innym nośniku danych). Inspiracją do przeprowadzenia badań było rozwiązanie problemu kodowania sekwencji wideo do celów montażu nieliniowego (obróbki materiałów źródłowych wykorzystywanych przy produkcji audycji telewizyjnych, filmów itp.), co wymagało wykorzystania trybu bezstratnego, kodującego w czasie rzeczywistym. Autor tego artykułu jest członkiem zespołu badawczego realizującego, w ramach grantu badawczego, projekt sprzętowego systemu bezstratnej kompresji sekwencji wideo, opartego na architekturze Network on Chip [5].

1.2. Klasyfikacja metod kodowania bezstratnego

Nieustanne dążenie do uzyskania coraz większej efektywności bezstratnej kompresji obrazu prowadzi do opracowywania metod o wzrastającej złożoności implementacyjnej. Lata 90. XX wieku były okresem największej aktywności projektantów nowych metod. Do dziś uznawana jest za jedną z najefektywniejszych – metoda, zaprezentowana w roku 1996, CALIC [28]. Na owe czasy metoda ta okazała się zbyt wymagająca obliczeniowo w porównaniu z metodą LOCO-I, której modyfikacja stała się standardem JPEG-LS [27]. Wśród algorytmów o wysokiej złożoności implementacyjnej (obliczenia zmiennoprzecinkowe, wstępna analiza danych, rozwiązywanie równań macierzowych) można wyróżnić prace trzech zespołów badawczych: metoda TMW (1997) [16] i jej późniejsze rozwinięcie TMW^{LEGO} (2001) [17], WAVE-WLS (2002) [29, 31] oraz najnowsza wersja MRP 0.5 zaprezentowana pod nazwą «VBS & new-cost» (2005) [14]. Zakodowanie jedego obrazu przy użyciu każdej z tych propozycji wymaga wielu godzin (lub wielu minut, jeśli dysponujemy najwydajniejszym obecnie procesorem dostępnym na rynku) pracy programu kodującego. Oprócz wspomnianych wcześniej metod wykorzystujących modelowanie predykcyjne istnieją także bezstratne wersje koderów falkowych stosowane zarówno do kodowania w trybie intraframe (np. JPEG-2000 [11]), jak i interframe [18]. Jednak uzyskiwane wyniki nie dorównują najlepszym metodom predykcyjnym.

Największą elastycznością cechuje się technika mieszania predyktorów (*blending predictors*), użyta choćby we wspomnianych metodach TMW oraz WAVE-WLS. Pomysł mieszania predyktorów (stanowiących zbiór subpredyktorów) przedstawiony szerzej w pracy [20] był rozwijany między innymi przez G. Denga oraz H. Ye i prezentowany np. w pracy [3]. Jest to metoda konkurencyjna do selekcji aktualnie najlepszego predyktora [12]. Po dogłębnej analizie cech charakterystycznych można sklasyfikować obecne metody bezstratnej kompresji obrazu (sygnału luminancji) jako metody z (analiza czasów kodowania/dekodowania została przeprowadzona dla obrazu Lena 512×512 pikseli i odnosi się do procesora Pentium4 3.0 GHz):

– bardzo szybkim kodowaniem i dekodowaniem, czas liczony w milisekundach – JPEG-LS [27];

– szybkim kodowaniem i dekodowaniem, czas w okolicach 1 sekundy – CALIC [28], JPEG2000 [11], Blend-A13$_+$ (wersje Hard-13 i Soft-13 omówione w dalszej części pracy);
– akceptowalnym czasem kodowania i dekodowania (kilkanaście/kilkadziesiąt sekund) – Glicbawls [15], SWAP [9];
– nieakceptowalnym czasem kodowania (kilkadziesiąt minut) – TMW$^{\mathrm{LEGO}}$ [17], MRP [14], WAVE-WLS [29].

Pojęcie akceptowalności odnosi się do aktualnie zaprojektowanych metod, które po etapie testowania, mogły by zostać zaproponowane (z wyprzedzeniem co najmniej 5 lat) jako przyszłe standardy kodowania. Należy bowiem uwzględnić fakt wzrostu wydajności współczesnych komputerów na przełomie kilku lat, dotyczy to zarówno takich parametrów jak częstotliwość zegara, wielkość pamięci podręcznych, liczba potoków, jak i odgrywająca wśród komputerów osobistych coraz większą rolę kostrukcja wielordzeniowa.

Możemy też zdefiniować drugi typ klasyfikacji, w której decydującym czynnikiem są proporcje czasowe między kodowaniem a dekodowaniem. Metody symetryczne czasowo mają zbliżoną złożoność obliczeniową kodera i dekodera. Drugi typ charakteryzujący się dłuższym czasem kodowania względem dekodowania, to metoda asymetryczna. Wśród metod o czasie kodowania nieprzekraczającym sekundy nie ma większego znaczenia identyfikacja asymetryczności. Dopiero w przypadku długich czasów kodowania warto zaakcentować fakt, gdy metoda jest asymetryczna i przykładowo czas dekodowania może być akceptowalnie mały (np. kilka sekund). Jest to ważna cecha, gdyż operację kodowania najczęściej przeprowadza się raz, a dekodowania wielokrotnie. Wsród wymienionych trzech najefektywniejszych metod TMW$^{\mathrm{LEGO}}$ oraz MRP należą do kategorii asymetrycznych (iteracyjne poszukiwanie parametrów na etapie kodowania), natomiast WAVE-WLS jest metodą symetryczną czasowo (adaptacyjna metoda predykcyjna o dużej złożoności obliczeniowej kodowania i dekodowania każdego piksela).

2. Proste metody predykcyjne

W podpunkcie 2.1 przedstawione zostaną podstawy predykcyjnego modelowania obrazów. Po analizie literatury zdecydowaliśmy się na metodę mieszania predyktorów (zasada działania opisana zostanie w rozdziale 3), gdyż zwłaszcza prace G. Denga ukazują jej wysoką skuteczność dla szerokiej klasy obrazów. Z tego powodu w podpunktach 2.2 oraz 2.3 przedstawiono proste predyktory stanowiące części składowe mieszania predykcyjnego.

2.1. Podstawy predykcyjnego modelowania obrazów

W przypadku sygnałów dwuwymiarowych, jakimi są obrazy, możemy zaobserwować podobieństwo między sąsiednimi pikselami. Modelowanie danych sprowadza się w takim przypadku do próby usunięcia jak największej ilości informacji wzajemnej występującej między sąsiadującymi ze sobą pikselami. Po etapie modelowania obrazu możemy uznać, iż

przekształcony sygnał cyfrowy stanowi (w przybliżeniu) zbiór symboli wzajemnie niezależnych. Wówczas entropia bezwarunkowa dla danych tego typu (entropia źródła bez pamięci – rzędu zerowego) może być wyznaczona ze wzoru:

$$H = -\sum_{i=e_{\min}}^{e_{\max}} p_i \cdot \log_2 p_i \tag{2.1}$$

gdzie p_i jest prawdopodobieństwem wystąpienia symbolu o wartości i należącym do zbioru symboli o wartościach z przedziału od $e_{\min}$ do $e_{\max}$. Entropia informuje nas o minimalnej średniej liczbie bitów potrzebnej do zakodowania jednego piksela obrazu przy użyciu jednej z metod statycznego kodowania entropijnego (np. kodu Huffmana, Golomba, czy kodu arytmetycznego) [19].

Typowy predyktor liniowy rzędu r jest wartością przewidywaną n-tego piksela na podstawie r sąsiednich (zgodnie z zachowaniem zasady przyczynowości znanych koderowi i dekoderowi) pikseli. Ma on postać:

$$\hat{x}_n = \sum_{j=1}^{r} b_j \cdot x_{n-j} \tag{2.2}$$

gdzie elementy x_{n-j} są wartościami pikseli z najbliższego otoczenia (sąsiedztwa) aktualnie kodowanego piksela x_n, natomiast elementy b_j to współczynniki predykcji [19]. Użycie predyktora liniowego polega na kodowaniu tylko błędów predykcji, czyli różnic e_n między wartością rzeczywistą a przewidywaną (zaokrągloną do najbliższej liczby całkowitej), które najczęściej są niewielkimi wartościami oscylującymi w pobliżu zera:

$$e_n = x_n - \hat{x}_n \tag{2.3}$$

Do wyznaczenia współczynników predykcji wykorzystuje się często kryterium minimalizacji błędu średniokwadratowego (MMSE), jednak przy realizacji sprzętowej czasu rzeczywistego nie rozpatrywano tej metody (wymaga ona wprowadzenia opóźnień oraz rozwiązywania układu równań liniowych z wykorzystaniem liczb zmiennoprzecinkowych), ani innych metod wymagających wstępnego przygotowania parametrów modelowania i kompresji.

Jedną z propozycji zmniejszania entropii bezwarunkowej kodowanych obrazów jest odpowiedni dobór sąsiednich pikseli oraz rzędu predykcji. Ze względu na kierunek kodowania (koduje się kolejne wiersze od góry do dołu, a każdy z nich od lewej do prawej) zarówno koder, jak i dekoder mają dostęp do pikseli powyżej oraz po lewej stronie aktualnie kodowanego (dekodowanego) piksela. Rząd predykcji r, to liczba pikseli z najbliższego sąsiedztwa brana pod uwagę przy wyznaczaniu wartości przewidywanej $\hat{x}_n$. Dla obrazów naturalnych wzrost rzędu predykcji teoretycznie pozwala na uzyskanie spadku wartości wariancji zakodowanego sygnału, jednak im dalsze sąsiedztwo, tym należy się spodziewać mniejszego poziomu informacji wzajemnej. Korzystając z tej zasady, możemy dokonać numeracji sąsiadów kodowanego piksela zgodnie z odległością metryczną (euklidesową)

$\bar{d}(j) = \sqrt{(\Delta x_j)^2 + (\Delta y_j)^2}$ ich środków. Numeracja pikseli o tej samej odległości $\bar{d}$ jest wyznaczana zgodnie z ruchem wskazówek zegara. Rysunek 1 obrazuje 30 najbliższych pikseli sąsiedztwa x_n. Teoretycznie im wyższy numer piksela tym mniejsze jego znaczenie dla poprawy efektywności kodowania. Możemy zatem przyjąć, że korzystając z predyktora rzędu r używamy pikseli o numerach od 1 do r.

			26	24	27			
	29	20	16	14	17	21	30	
	19	11	8	6	9	12	22	
25	15	7	3	2	4	10	18	28
23	13	5	1	x_n				

Rys. 1. Numeracja pikseli sąsiedztwa

2.2. Analiza prostych predyktorów liniowych

Mając dany zbiór prostych, stałych predyktorów, można zauważyć, iż wartości entropii uzyskiwane przy kodowaniu każdym z predyktorów z osobna nie są imponujące, jednak użycie choćby siedmiu prostych predyktorów wspólnie (traktowanych dalej jako subpredyktory) pozwala na istotne zmniejszenie wartości entropii [20] i to dla znaczącej większości obrazów testowych. Dlatego właśnie ta propozycja polegająca na mieszaniu prostych subpredyktorów stała się podstawą modelowania w naszym układzie modelowania. Jej dodatkową zaletą jest możliwość równoległego wykonywania wielu obliczeń w ramach kodowania pojedynczego piksela.

Nie można przełożyć wprost zasady, że spośród v stałych predyktorów dobór n subpredyktorów dających indywidualnie najmniejsze wartości entropii pozwoli uzyskać najlepszy rezultat w metodzie mieszania subpredyktorów. Wręcz przeciwnie, opłacalne staje się użycie subpredyktorów jak najbardziej różnorodnych i dających małe błędy predykcji w odrębnych sytuacjach (przy różnych typach najbliższego otoczenia). Dlatego podążając tym tropem, nasze badania ukierunkowano na znalezienie różnych typów metod predykcyjnych, które indywidualnie nie muszą przodować w ogólnych testach na dużych zbiorach obrazów.

Dobór subpredyktorów oparto na wynikach badań średniej entropii 45 obrazów testowych dla zestawu ponad 30 stałych predyktorów, który został zebrany z prac takich jak [2, 4, 10, 13, 19, 20]. Wśród 45 obrazów o ośmiobitowej głębi (256 odcieni szarości – luminancji) znalazło się 26 o rozdzielczości 512×512 pikseli, 10 o rozdzielczości 256×256 oraz 9 testowych („Balloon”, „Barb”, „Barb2”, „Board”, „Boats”, „Girl”, „Goldhill”, „Hotel”, „Zelda”) obrazów o rozdzielczości 720×576.

Każdy z autorów algorytmów wykorzystujących metodę mieszania prezentował własne zestawy subpredyktorów. Ze względu na problemy ze złożonością implementacyjną były to najczęściej proste stałe predyktory rzędu od 1 do 3. W przypadku predyktorów rzędu 1 proponowano wartości najbliższych sąsiednich pikseli $P(i)$, gdzie i jest numerem pik-

sela z sąsiedztwa (patrz rys. 1). Do tych najprostszych subpredyktorów najczęściej dołączane były także stałe predyktory, np. planarny (określany jako Plane lub JPEG4), predyktor Pirsha itp. W projektowanym przez nas algorytmie Blend-13_+ wykorzystane zostały piksele $P(i)$ o numerach $i = \{1, 2, 3, 4, 5, 10, 18, 28\}$, liniowe subpredyktory [20]: GradWest = $= 2P(1) - P(5)$, GradNorth = $2P(2) - P(6)$, Plane = $P(1) + P(2) - P(3)$, Plane2 = $P(1) - P(2) + P(4)$ oraz ulepszona wersja metody nieliniowej predykcji stosowanej w algorytmie CALIC – jest to modyfikacja GAP_+ zaprezentowana w pracy [26] (patrz podpunkt 2.3).

2.3. Metoda predykcji z podziałem kontekstowym GAP

Metoda predykcji z podziałem kontekstowym GAP (Gradient-Adjusted Predictor) o stałych współczynnikach została zaproponowana jako wstępna metoda predykcji w algorytmie CALIC (Context Based Adaptive Lossless Image Coding) [28]. Metoda GAP korzysta z 7 sąsiednich pikseli do wyznaczania wartości przewidywanej i jednego z 7 kontekstów. Przy czym są to piksele o numerach $\{1, 2, 3, 4, 5, 6, 9\}$. Decyzja o wyborze odpowiedniego kontekstu podejmowana jest na podstawie liczby $d_{GAP} = d_h - d_v$, gdzie [28]:

$$\begin{aligned} d_h &= |P(1) - P(5)| + |P(2) - P(3)| + |P(4) - P(2)| \\ d_v &= |P(1) - P(3)| + |P(2) - P(6)| + |P(4) - P(9)| \end{aligned} \tag{2.4}$$

Zakres wartości d_{GAP} jest podzielony na 7 przedziałów na podstawie 3 progów wyznaczonych eksperymentalnie przez autorów metody. Dla obrazów o 256 odcieniach szarości wartości progów wynoszą odpowiednio: $T_1 = 8$, $T_2 = 32$, $T_3 = 80$ (nasze badania doprowadziły do zmiany tych progów do wartości: $T_1 = 6$, $T_2 = 25$, $T_3 = 78$ [24]). Poniżej przedstawiono w postaci pseudokodu algorytm wyznaczania numeru kontekstu [28]:

```
if (d_GAP >T_3) kontekst = 7;
else if (d_GAP < -T_3) kontekst = 6;
else {
              kontekst = 1;
              if (d_GAP >T_2) kontekst = 5;
              else if (d_GAP >T_1)  kontekst = 4;
              else if (d_GAP < -T_2) kontekst = 3;
              else if (d_GAP < -T_1) kontekst = 2;
}
```

Zmodyfikowaną wersję metody GAP zaprezentowano w pracy [26]. W niniejszej pracy będzie ona oznaczana jako GAP_+. Każdemu z kontekstów przyporządkowano indywidualny stały predyktor liniowy wykorzystujący od 1 do 5 spośród 6 sąsiednich pikseli. Tabela 1 zawiera współczynniki predykcji dla każdego z 7 kontekstów.

W pierwszych badaniach [23] uwzględniono także modyfikację MED_+ zaprezentowaną w pracy [8] (składającą się również z 7 kontekstów, choć podstawowa wersja MED [27] była trójkontekstowa), jednak w trakcie dalszych badań lepsze rezultaty uzyskano, rezygnując z tego subpredyktora.

Tabela 1
Zestaw współczynników predykcji odpowiadających poszczególnym kontekstom metody GAP_+

Współczynnik \ Kontekst	1	2	3	4	5	6	7
b_1	1/2	7/8	5/4	3/8	1/4	2	0
b_2	1/2	3/8	1/4	7/8	5/4	0	2
b_3	–1/4	–3/16	–1/8	–3/16	–1/8	0	0
b_4	1/4	3/16	1/8	3/16	1/8	0	0
b_5	0	–1/4	–1/2	0	0	–1	0
b_6	0	0	0	–1/4	–1/2	0	–1

3. Metoda mieszania predyktorów

Wysoką efektywność mieszania predykcyjnego dla każdego z obrazów uzyskuje się dzięki elastyczności metody adaptacji decydującej o doborze subpredyktora dominującego (tych dominujących może być kilka w różnych częściach obrazu). Istotność danego subpredyktora jest tym większa, im mniejsze błędy predykcji uzyskano w najbliższym otoczeniu. Całkowitą wartość błędu otoczenia E_i skojarzonego z i-tym subpredyktorem wyznaczamy ze wzoru:

$$E_i = 1 + 2(e_n^2(1) + e_n^2(2)) + \sum_{j=3}^{p} e_n^2(j) \tag{3.1}$$

gdzie $e_n(j)$ oznacza wartość błędu predykcji uzyskaną na podstawie i-tego subpredyktora w sąsiedztwie o numerze względnym j (patrz rys. 1). Aby wartość w_i wagi subpredyktora odpowiadała aktualnej jego istotności, należy ją wyznaczać z poniższego wzoru:

$$w_i = \frac{\alpha(i)}{E_i} \tag{3.2}$$

gdzie $\alpha(i)$ jest współczynnikiem wzmocnienia subpredyktora. Biorąc pod uwagę zbiór m subpredyktorów, możemy wyznaczyć przypisane im dodatnie współczynniki predykcji a_i, uwzględniając przy tym ich normalizację (suma wszystkich współczynników a_i powinna wynosić 1):

$$a_i = \frac{w_i}{\sum_{j=1}^{m} w_j} \tag{3.3}$$

Wówczas wynikowa wartość przewidywana predyktora głównego ma postać:

$$\hat{x} = \sum_{i=1}^{m} a_i \cdot \hat{x}_i \tag{3.4}$$

Jak można zauważyć, mieszanie to kombinacja liniowa m subpredyktorów, a ich wagi właściwe a_i zależą od poziomu błędów predykcji E_i z najbliższego otoczenia. Pomysł mieszania predyktorów (stanowiących zbiór subpredyktorów) przedstawiony w pracy [20] był rozwijany między innymi przez G. Denga oraz H. Ye i prezentowany np. w pracy [3]. Jednakże autorzy tych opracowań w dość uproszczony sposób podchodzili do sprawy doboru wielkości otoczenia branego pod uwagę przy wyznaczaniu współczynników wagowych każdego z używanych subpredyktorów. W pracy [25] podjęto próbę przeanalizowania wpływu wielkości otoczenia na efektywność predykcyjnego modelowania danych. Problemem, jaki należało rozwiązać, był fakt, iż dla badanych obrazów optymalne (wspólne dla wszystkich subpredyktorów) otoczenia mieściły się w całym analizowanym zakresie p z przedziału od 3 do 30. Seemann i Tisher [20] zaproponowali wartość $p = 3$, Deng w swych pracach używał $p = 4$ [3]. Na podstawie naszych pierwszych badań (dla zbioru 45 różnych obrazów testowych) dobre średnie rezultaty uzyskano przy $p = 6$ [23] – metoda Blend-14. W wyniku dalszych analiz po zastosowaniu kontekstowej korekcji błędu (patrz punkt 4) i usunięciu subpredyktora MED_+ zwiększono wartość p do 10 uzyskując metodę oznaczaną jako Blend-13_+ (patrz tab. 2). Szary obszar zaznaczony na rysunku 1 przedstawia wielkość tego otoczenia.

W literaturze nie pojawia się pojęcie współczynnika wzmocnienia subpredyktora, jednak na podstawie eksperymentów można stwierdzić przydatność tego parametru. Dla dobranych przez nas $m = 13$ subpredyktorów: $x_1 = GAP_+$, x_2 = GradWest, x_3 = GradNorth, x_4 = Plane, x_5 = Plane2, $x_6 = P(1)$, $x_7 = P(2)$, $x_8 = P(3)$, $x_9 = P(4)$, $x_{10} = P(5)$, $x_{11} = P(10)$, $x_{12} = P(18)$, $x_{13} = P(28)$ ustalono odpowiadający im zestaw współczynników $\alpha(i) = \{1, 2, 2, 1, 3/2, 1, 1, 1, 1, 1, 1, 1, 1\}$.

4. Kontekstowa korekcja błędu predykcji

Oprócz zastosowania podstawowej zasady, jaką jest mieszanie predykcyjne, w proponowanej metodzie modelowania użyto dodatkowo korekcji wartości przewidywanej, która uwzględnia omówiony w tym punkcie podział kontekstowy.

4.1. Podział kontekstowy

Główna idea podziału kontekstowego pochodzi z algorytmu CALIC [28], gdzie oprócz metody wstępnej predykcji GAP (z podziałem na 7 kontekstów) wykorzystuje się podział na 576 kontekstów służących do korekcji błędu predykcji skojarzonego z odpowiednim kontekstem. Kontekst charakteryzuje się indywidualnymi cechami najbliższego otoczenia aktualnie kodowanego piksela, uwzględnia wzajemne zależności występujące między sąsiednimi pikselami, a często także ich poziom wariancji.

Najprostszą metodą wyznaczania wartości bazowej jest obliczenie średniej z n stałych subpredyktorów, np. dla $n = 8$ wykorzystywane są subpredyktory $P(1)$, $P(2)$, $P(3)$, $P(4)$, $P(5)$, $P(6)$, GradNorth oraz GradWest. Następnie porównuje się wartość każdego z nich ze średnią z tych ośmiu subpredyktorów. Jeśli wartość i-tego predyktora jest większa od średniej, to znacznik bitowy z_i ustawiany jest na 1, w przeciwnym przypadku na 0. Ze znaczników tworzymy n bitową liczbę $z_{n-1}...z_3z_2z_1z_0$, która stanowi numer kontekstu. W przypadku $n = 8$ mamy 256 kontekstów otoczenia. Dodatkowo można wyznaczyć miarę poziomu odchylenia od średniej mierząc poziom wariancji tych n predyktorów. Wartość ta może zostać skwantyzowana do Q przedziałów, co łącznie daje $Q \cdot 2^n$ kontekstów. Podobną ideę podziału kontekstowego zaprezentowano w pracy [7].

W naszych badaniach wykorzystano $n = 8$ powyżej opisanych predyktorów i $Q = 4$ poziomy kwantyzacji wariancji. Daje to łącznie 1024 konteksty, przy czym średnią arytmetyczną z 8 predyktorów zastąpiono poniższą średnią wagową:

$$\bar{x} = \frac{102}{1024}\left(3 \cdot (P(1) + P(2)) + \sum_{j=3}^{6} P(j)\right) \tag{4.1}$$

Wartość wariancji (pomnożonej przez 8) możemy wyznaczyć ze wzoru:

$$\sigma^2 = (\bar{x} - \text{GradNorth})^2 + (\bar{x} - \text{GradWest})^2 + \sum_{i=1}^{6} (\bar{x} - P(i))^2 \tag{4.2}$$

Aby uzyskać podział na 4 poziomy kwantyzacji, wyznaczono eksperymentalnie 3 progi wartości σ^2 odpowiednio wynoszące 400, 2500, 8000 (z progami odpowiednio równymi 0,18, 1,14 oraz 3,65 wartości średniej wariancji zbioru uczącego nie związanego ze zbiorem obrazów testowych).

4.2. Metody korekcji skumulowanego błędu predykcji

Metody predykcji mogą w wielu sytuacjach wprowadzać składowe stałe o indywidualnych cechach skojarzonych z danym kontekstem. W tym celu proponuje się wykorzystanie adaptacyjnej metody usuwania składowej stałej zwanej także korekcją błędu predykcji skojarzonego z odpowiednim kontekstem.

Metoda usuwania składowej stałej stosowana jest zarówno w algorytmie CALIC, jak i w JPEG-LS. Dla każdego kontekstu odnotowuje się liczbę wystąpień oraz skumulowaną sumę błędów i na ich podstawie koryguje się aktualnie wyznaczany błąd predykcji [28]. Różnice między metodami są nieznaczne, wersja opracowana dla JPEG-LS jest lepiej przygotowana pod względem realizacji sprzętowej, gdyż nie wymaga użycia operacji mnożenia i dzielenia. Poniżej przedstawiono oba algorytmy, gdzie i oznacza numer kontekstu, $B[i]$ to aktualna wartość skumulowana błędu predykcji e_n w danym kontekście, $N[i]$ jest licznikiem wystąpień danego kontekstu, natomiast $C[i]$ oznacza aktualną wartość korekcji,

którą należy dodać do wartości przewidywanej (przy najbliższym wystąpieniu danego kontekstu) jako jej wartość korygującą:

$$\dot{x} = \hat{x} + C[i] \tag{4.3}$$

Zakładając, iż sytuacja nagłego pojawienia się dużego błędu predykcji e_n może zbyt mocno zmienić wartość korygującą $C[i]$, nie wykonujemy adaptacji $B[i]$, gdy $|e_n| \geq 32$. Inicjalizację tablic liczności poszczególnych kontekstów dokonuje się wartością większą od zera, którą ustalono na $N[i] = 4$ (według standardu JPEG-LS jest to wartość 1), co zapobiega początkowo zbytniemu rozrzutowi wartości korekcji $C[i]$, która powinna być traktowana jako uśrednienie wystarczająco dużej liczby reprezentantów kontekstu.

Algorytm adaptacji wartości korekcji $C[i]$ dla metody CALIC:

```
Wartości początkowe: B[i] = C[i] = 0, N[i] = 4 dla każdego i;

if (abs(e_n) < 32) {
B[i] = B[i] + e_n;
N[i] = N[i] + 1;
C[i] = B[i]/N[i];
}
```

Algorytm adaptacji wartości korekcji $C[i]$ dla metody JPEG-LS:

```
Wartości początkowe: B[i] = C[i] = 0, N[i] = 4 dla każdego i;

if (abs(e) < 32) {
B[i] = B[i] + e_n;
N[i] = N[i] + 1;
if (B[i] Ł -N[i]) {
              C = C[i] - 1;
     B[i] = B[i] + N[i];
              if (B[i] Ł -N[i]) B[i] = -N + 1;
}
else if (B[i] > 0) {
              C = C[i] + 1;
              B[i] = B[i] - N[i];
              if (B[i] > 0) B[i] = 0;
}
}
```

W obu algorytmach należy zastosować algorytm zapominania okresowego, który dodatkowo pozwala dostosować wartości składowych stałych $C[i]$ do lokalnych właściwości kontekstu:

```
if (N[i] > 127) {
              N[i] = 64;
              B[i] = B[i]/2;
}
```

W naszym rozwiązaniu zastosowaliśmy obie opisane metody, przy czym wartość składowej stałej wyznaczana jest jako średnia arytmetyczna obu wartości $C_{\text{CALIC}}[i]$ oraz $C_{\text{JPEG-LS}}[i]$, wówczas ostateczna wartość przewidywana wynosi:

$$\dot{x} = \hat{x} + \frac{C_{\text{CALIC}}[i] + C_{\text{JPEG-LS}}[i]}{2} \tag{4.4}$$

4.3. Porównanie metod predykcyjnych

W tabeli 2 zaprezentowano porównanie wartości entropii kilku podstawowych metod predykcyjnych. Metoda GAP_{+} została opisana w podpunkcie 2.3, jej rozwinięcie wykorzystujące metody korekcji skumulowanych błędów predykcji (patrz podpunkt 4.2) oznaczono jako GAP_{++}. Kolejne dwie kolumny zawierają wyniki konkurencyjnych propozycji mieszania i pochodzą z prac [20] oraz [3]. Najlepsze rezultaty uzyskano stosując opisane w tej pracy metody Blend-14 oraz Blend-13_{+}.

Tabela 2
Pomiar wartości entropii standardowych obrazów testowych

Obrazy	GAP_{+} [26]	GAP_{++}	Blend-7 [20]	WAVE [3]	Blend-14 [23]	Blend-13_{+}
„Balloon”	2,998	2,936	2,93	2,873	2,837	2,806
„Barb”	5,012	4,787	4,90	4,936	4,712	4,485
„Barb2”	5,003	4,905	5,02	4,872	4,900	4,800
„Board”	3,892	3,762	3,81	3,679	3,641	3,607
„Boats”	4,229	4,132	4,16	4,095	4,032	3,991
„Girl”	4,044	3,928	3,91	3,844	3,818	3,766
„Gold”	4,699	4,649	4,65	4,598	4,567	4,558
„Hotel”	4,676	4,583	4,58	4,500	4,479	4,445
„Zelda”	3,936	3,848	3,90	3,817	3,802	3,756
Średnia	4,276	4,170	4,207	4,135	4,087	4,024

5. Adaptacyjny koder arytmetyczny z podziałem kontekstowym

Omówione metody modelowania przygotowują dane do etapu kodowania właściwego, który generuje plik wynikowy. Proponowany rodzaj kompresji opiera się na zasadzie działania kodu arytmetycznego [19], którą rozbudowano o system adaptacji i przełączania między wieloma rozkładami prawdopodobieństw.

5.1. Podstawowa forma adaptacji rozkładu prawdopodobieństwa

Ze względu na fakt kodowania wartości bezwzględnych błędów predykcji (bit znaku koduje się osobno – patrz podpunkt 5.4), mamy do czynienia z rozkładem zbliżonym do jednostronnego rozkładu Laplace'a. Początkowy rozkład prawdopodobieństwa, a właściwie wektor liczebności wystąpień wartości i (poszczególnych wartości bezwzględnych błędów predykcji), możemy potraktować jako jego przybliżenie wykorzystując uproszczony wzór [15]:

$$n_e(i) = \left\lfloor A \cdot 0.8^i \right\rfloor + 1 \tag{5.1}$$

Dobre rezultaty uzyskuje się dla $A = 10$. Adaptacja przejawia się tym, iż po wczytaniu i zakodowaniu każdej kolejnej wartości $|e_n|$ należy uaktualnić wektor liczebności zwiększając wartość pod indeksem $|e_n|$ o jeden: $n_e(|e_n|) = n_e(|e_n|) + 1$. Dodatkowo możemy wprowadzić efekt zapominania, który sprzyja zmniejszaniu wag tym liczebnościom, które wśród ostatnio kodowanych wartości się nie pojawiły lub pojawiały się rzadziej, niż we wcześniejszym etapie kodowania [6]. W tym celu kontrolujemy łączną liczbę zakodowanych do tej pory wartości, a dokładniej wartość licznika, który jest zwiększany o jeden po każdym zakodowaniu liczby $|e_n|$. Jeśli licznik osiągnie z góry założoną wartość 2^s, wówczas wszystkie elementy wektora liczebności zmniejszane są o połowę:

$$n_e(i) = \left\lfloor \frac{n_e(i)}{2} \right\rfloor + 1, \text{ dla każdego } i \text{ od } e_{\min} \text{ do } e_{\max} \tag{5.2}$$

Po przeskalowaniu ponownie obliczana jest wartość licznika jako suma wszystkich elementów $n_e(i)$ wektora liczebności. Dobre rezultaty uzyskuje się dla $s \geq 10$. W naszym koderze użyto $s = 13$ dla kontekstów kodujących wartości skwantyzowanych błędów predykcji oraz $s = 10$ dla pozostałych kontekstów (kodujących, omówione w podpunktach 5.3 oraz 5.4, błędy kwantyzacji oraz bit znaku).

5.2. Podział kontekstowy

Opisany w podpunkcie 5.1 koder adaptacyjny potrafi dostosować się do rozkładu w sensie długookresowym, lecz można wykorzystać jeszcze fakt istnienia krótkookresowych zależności między kolejno kodowanymi danymi wykorzystując najbliższe sąsiedztwo dwuwymiarowego sygnału błędów predykcji. Cechy sąsiednich błędów predykcji (bez uwzględniania znajomości cech samego sąsiedztwa pikseli) potrafią dość dokładnie określić właściwy typ rozkładu aktualnie kodowanej wartości $|e_n|$. Wychodząc z takiego założenia, możemy zaprojektować kontekstowy koder arytmetyczny posiadający nie jeden, lecz N rozkładów prawdopodobieństw skojarzonych z poszczególnymi numerami konteksów od 0 do $N - 1$. Teoretycznie wraz ze wzrostem liczby kontekstów można oczekiwać zwiększenia

efektywności kompresji, jednakże pojawia się problem rozrzedzenia kontekstów, czyli zbyt wolnej adaptacji ich rozkładu. Adaptacyjny charakter budowy rozkładów wymaga szybkiego ustalenia się przybliżonej docelowej postaci każdego z N rozkładów, dlatego należy ustalić pewien kompromis między liczbą kontekstów, a szybkością ich adaptacji. Często wykorzystuje się 8 [28], 16 a nawet 20 kontekstów [4].

Istotnym czynnikiem wpływającym na efektywność kompresji jest odpowiedni dobór reguły decyzyjnej wyznaczającej numer kontekstu. Zaprezentowana tu reguła jest rozwinięciem pomysłu z prac Denga. Wartość *temp* jest wyznaczana jako liczba całkowita i bazuje na błędach $e(i)$ znajdujących w najbliższym otoczeniu, gdzie indeks i określa położenie zgodnie z rysunkiem 1:

$$temp = \max\{2|e(1)|,\ 2|e(2)|,\ \frac{9}{8}(|e(3)|+|e(4)|),\ |e(5)|+|e(10)|,\ |e(6)|+|e(7)|,$$

$$\frac{13}{8}|e(4)|,\ \frac{3}{2}|e(3)|,\ \frac{7}{8}(|e(8)|+|e(9)|),\ \frac{11}{8}(|e(1)|+|e(2)|)\}.$$

Wartość *temp* poddajemy kwantyzacji z użyciem $N-1$ progów $Th_{(i)}$, aby uzyskać numer kontekstu wskazującego na aktualny rozkład prawdopodobieństwa. Przykładowo dla $N = 16$ progi można ustalić następująco: $Th_{(i)}$ = {2, 4, 7, 10, 13, 17, 21, 27, 33, 39, 50, 60, 75, 90, 120}.

5.3. Kwantyzacja błędów predykcji

Aby zwiększyć prędkość adaptacji rozkładów skojarzonych z poszczególnymi kontekstami, często stosuje się kwantyzację wartości $|e_n|$. Jest to możliwe dzięki zrzutowaniu przedziału liczb $|e_n|$ od 0 do 255 na mniejszy zakres liczb k, np. od 0 do 17. Idea taka stosowana jest w wielu metodach kodowania, np. w standardzie JPEG. Technika kwantyzacji z bezstratnym kodowaniem ma za zadanie podzielić wartość $|e_n|$ na dwie liczby, które są kodowane z użyciem odrębnych rozkładów (czasem przyjmuje się, iż błąd kwantyzacji zapisywany jest jako ciąg bitów bez kompresji). Kwantyzacja i etap kodowania przebiega następująco:

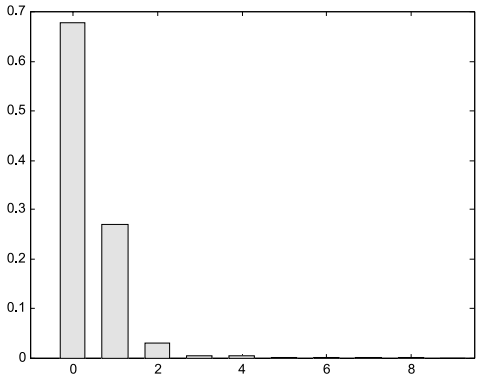
Wyznacza się wartość k z zależności $T_{(k)} \le |e_n| < T_{(k+1)}$, mając dany zbiór kolejnych progów kwantyzacji T = {0, 1, 2, 3, 4, 5, 6, 7, 8, 10, 12, 14, 16, 20, 24, 32, 64, 128}. Skwantyzowana wartość oznaczona jako k jest wysyłana do kodera arytmetycznego i kodowana po skojarzeniu jej z odpowiednim kontekstem wyznaczonym według zasady opisanej w podpunkcie 5.2. Błąd kwantyzacji $e_q = |e_n| - T_{(k)}$ jest jest traktowany jako $q_{(k)}$ bitowa liczba, gdzie $q_{(k)}$ odczytujemy jako k-tą wartość z wektora liczb q = {0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 2, 2, 3, 5, 6, 7}. Jeśli $q_{(k)} > 0$, wartość e_q może być wysłana na wyjście w postaci $q_{(k)}$ bitów lub kodowana za pomocą jednego z 7 uniwersalnych adaptacyjnych koderów arytmetycznych (o numerze q_k).

Dekodowanie polega na odczytaniu wartości k będącej indeksem do tablicy wartości progowych $T_{(k)}$ oraz do tablicy liczby bitów kwantyzacji q. Jeśli $q_{(k)} > 0$, to odczytaj z de-

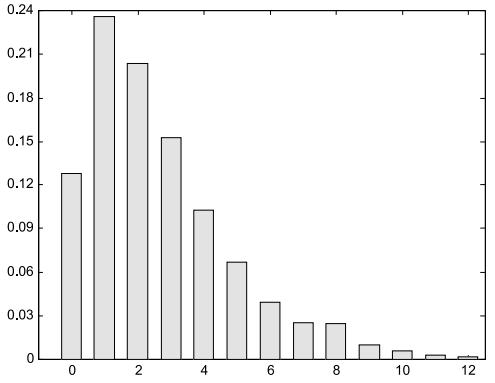
kodera (o numerze $q_{(k)}$) wartość błędu kwantyzacji e_q, w przeciwnym przypadku $e_q = 0$. Następnie wyznacz wartość $|e_n|$ z następującej zależności: $|e_n| = T_{(k)} + e_q$.

Przykładowo niech $|e_n| = 29$. Wówczas $T_{(14)} \le 29 < T_{(15)}$, czyli $k = 14$, $T_{(14)} = 24$, $e_q = 29 - 24 = 5$, $q_{(14)} = 3$, zatem liczbę 5 kodujemy używając trzeciego uniwersalnego kodera arytmetycznego błędów kwantyzacji (kodującego liczby od 0 do 7). W dekoderze $|e_n| = T_{(14)} + e_q = 24 + 5 = 29$.

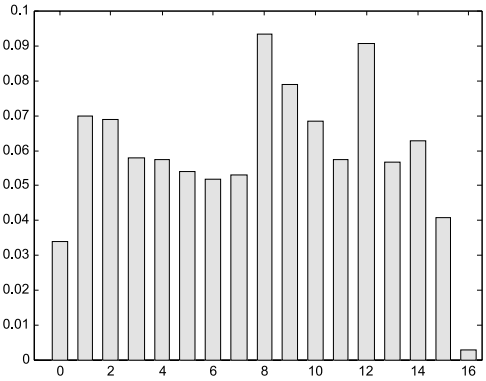
Na rysunkach 2–4 przedstawiono rozkłady prawdopodobieństw wartości $|e_n|$ w kontekstach odpowiednio 0, 4 oraz 11. Jeśli porównać rysunki 2 i 3 (przypominające rozkład Laplace'a) z rysunkiem 4, daje się wyraźnie zauważyć wpływ zarówno kwantyzacji samej wartości $|e_n|$, jak i kwantyzacji zmiennej *temp*.



Rys. 2. Rozkład prawdopodobieństwa wartości k w kontekście nr 0 dla obrazu „Hotel"



Rys. 3. Rozkład prawdopodobieństwa wartości k w kontekście nr 4 dla obrazu „Hotel"



Rys. 4. Rozkład prawdopodobieństwa wartości k w kontekście nr 11 dla obrazu „Hotel"

5.4. Kodowanie bitu znaku

Ze względu na symetryczność rozkładu błędów predykcji wygodniej kodować jest ich wartości bezwzględne $|e_n|$, co pozwala na szybszą adaptację rozkładów w poszczególnych kontekstach kodera arytmetycznego (patrz podpunkt 5.2). Osobno kodowany jest bit znaku, przy czym informację o nim trzeba zapisywać jedynie dla niezerowych wartości e_n. W niektórych algorytmach nie przewiduje się specjalnego sposobu kodowania bitu znaku, zapisując wartość tego bitu bez zmian do pliku wynikowego. W tej pracy zaproponowano użycie adaptacyjnego kodera arytmetycznego z podziałem na 16 kontekstów. Rozkład tego dwusymbolowego źródła jest początkowo ustawiany jako jednostajny w każdym z kontekstów (liczniki wystąpień inicjalizowane są wartością 5). Wartość kontekstu wyznaczana jest jako liczba składająca się z 4 bitów, dwa pierwsze są bitami znaków błędów predykcji lewego i górnego sąsiada: sign(e(1)) oraz sign(e(2)) kodowanego błędu. Kolejne dwa bity wyznaczane są jako liczba będąca wynikiem kwantyzacji liczby *temp* z progami $Th_{(i)} = \{4, 10, 90\}$.

5.5. Analiza parametrów kodera adaptacyjnego

Analizując poszczególne parametry kodera adaptacyjnego, możemy uzyskaną na podstawie pomiaru dla 45 obrazów testowych (omówionych w podpunkcie 2.2) średnią 4,20532 uznać za wzorcową.

W pierwszym teście wyłączono podział na 16 kontekstów bitu znaku pozostawiając tylko jeden kontekst, co spowodowało zwiększenie średniej do 4,21971. Rezygnując całkowicie z adaptacyjnego kodowania arytmetycznego bitu średnia wzrosła do 4,23192, co ukazuje dużą przydatność kompresji bitu znaku.

Kolejne badanie, to pomiar wykonany bez podziału na 16 kontekstów błędów predykcji, gdzie uzyskano średnią 4,38554 (przy średniej entropii rzędu zerowego wynoszącej 4,43199), co pokazuje, jak dużą rolę odgrywa podział kontekstowy. Równie istotne,

zwłaszcza przy użyciu podziału na wiele kontekstów, jest użycie kwantyzacji wartości bezwzględnych błędów predykcji (patrz podpunkt 5.3), gdyż po wyłączeniu tej opcji średnia wynosiła 4,32121.

Odpowiedni dobór reguły decyzyjnej wyznaczającej numer kontekstu też przyczynił się do uzyskania dobrej efektywności. Reguła opisana w podpunkcie 5.2 składa się z 9 argumentów wyznaczania funkcji maksimum. W przypadku użycia dwóch pierwszych argumentów średnia wyniosła zaledwie 4,25904, ale już przy wykorzystaniu czterech warunków spadła do 4,21331, dołączenie kolejnych dwóch już tylko nieznacznie poprawiło średnią zmniejszając ją do 4,20859.

Najmniejszy wpływ na poprawę efektywności miał sposób inicjalizacji (zgodnie ze wzorem (5.1)) rozkładów w poszczególnych kontekstach, gdyż zainicjalizowanie rozkładem jednostajnym powodowało spadek średniej o zaledwie 0,00079 bitu na symbol.

6. Podsumowanie

Zaproponowana w artykule metoda kompresji została zaprojektowana z myślą o jego sprzętowej implementacji jako systemu czasu rzeczywistego (wersja Hard-13). Od algorytmu wymagano zatem niskiej złożoności implementacyjnej (małe wymagania pamięciowe, stosunkowo duża możliwość równoległego wykonywania obliczeń, niewykorzystywanie liczb zmiennoprzecinkowych), co skutkować musiało licznymi kompromisami między stopniem kompresji a czasem kodowania obrazu. Dzięki zastosowaniu metody mieszania predykcyjnego oraz adaptacyjnego kodera arytmetycznego pracującego w trybie przełączania kontekstów, udało się uzyskać efektywność wyższą, niż w przypadku metod takich jak JPEG-LS, czy CALIC. Wyniki zawarte w tabeli 3 zawierają porównanie opisanej w tym artykule metody Blend-A13$_+$ zarówno w wersji sprzętowej – Hard-13, jak i programowej – Soft-13 z innymi metodami znanymi z literatury (patrz podpunkt 1.2). Wersja programowa od wcześniej opisanej wersji Hard-13 różni się nieznacznie, lecz dzięki zastosowaniu liczb zmiennoprzecinkowych i lepszemu dostosowaniu parametrów kompresji udało się uzyskać jeszcze większą efektywność. Odbywa się to kosztem zwiększenia złożoności implementacyjnej, ale nadal zaliczyć ją można do szybkich metod symetrycznych, przykładowo obraz *Lena* o rozdzielczości 512×512 pikseli kodowany jest w czasie 1,27 sekundy na procesorze Pentium4 2.8 GHz. Wyniki średnich bitowych obu wersji zaproponowanej metody ustępują jedynie pozycjom zaklasyfikowanym w podpunkcie 1.2 jako nieakceptowalne czasowo.

W toku dalszych prac wyodrębnione zostaną dwa niezależne wątki badawcze. Pierwszy z nich dotyczyć będzie rozwinięcia metody do postaci umożliwiającej kodowanie sekwencji wideo. Drugi wątek, to rozwinięcie wersji programowej kodera o wiele dodatkowych możliwości przyczyniających się do dalszego wzrostu efektywności, w tym uwzględnienie bloku wstępnej analizy parametrów kodowania indywidualnie dobieranych do każdego kodowanego obrazu. Obie wersje zostaną także rozbudowane o możliwość kodowania prawie bezstratnego (tryb *near-lossless* [1]) oraz kompresji obrazów barwnych (o trzech składowych koloru RGB).

Tabela 3
Porównanie średnich bitowych znanych metod bezstratnego kodowania obrazów

Obrazy	Hard-13	Soft-13	JPEG-LS [22]	HBB [21]	CALIC [30]	P13 [4]	TMW^{LEGO} [17]	Multi-WLS [31]
Balloon	2,746	2,700	2,889	2,80	2,78	2,74	2,60	2,60
Barb	4,172	4,134	4,690	4,28	4,31	4,29	3,84	3,75
Barb2	4,441	4,389	4,684	4,48	4,46	4,47	4,24	4,18
Board	3,465	3,433	3,674	3,54	3,51	3,48	3,27	3,27
Boats	3,716	3,685	3,930	3,80	3,78	3,75	3,53	3,53
Girl	3,640	3,613	3,922	3,74	3,72	3,67	3,47	3,45
Gold	4,323	4,300	4,475	4,37	4,35	4,33	4,22	4,20
Hotel	4,189	4,155	4,378	4,27	4,18	4,19	4,01	4,01
Zelda	3,673	3,645	3,884	3,72	3,69	3,68	3,50	3,51
Średnia	3,818	3,784	4,058	3,889	3,864	3,844	3,631	3,611

Literatura

[1] Avcibas I., Memon N., Sankur B., Sayood K., A progressive Lossless/Near-Lossless image compression algorithm. *IEEE Signal Processing Letters*, 2002, vol. 9, no. 10, s. 312–314 (dokument w wersji rozszerzonej).

[2] Daaboul A., Local prediction for lossless image compression. *Proceedings of the Prague Stringology Club Workshop '98*, s. 44–50.

[3] Deng G., Ye H., A general framework for the second-level adaptive prediction. *Proceedings of ICASSP '03*, April 2003, vol. 3, s. III_237–240.

[4] Deng G., Ye H., Lossless image compression using adaptive predictor combination, symbol mapping and context filtering. *Proceedings of IEEE 1999 International Conference on Image Processing*, Kobe, Japan, Oct. 1999, vol. 4, s. 63–67.

[5] Dziurzański P., Mąka T., Ulacha G., Stasiński R., A lossless compression system realization utilizing phit-serial network-on-chip paradigm. *Proceedings of 13th European Signal Processing Conference EUSIPCO-07 CD*, September 2007.

[6] Gallager R. G., Variations on a theme by Huffman. *IEEE Transactions on Information Theory*, November 1978, vol. 24, no. 6, s. 668–674.

[7] Golchin F., Paliwal K. K., A lossless image coder with context classification, adaptive prediction and adaptive entropy coding. *Proceedings of IEEE International Conference on Acoustics, Speech and Signal Processing*, Seattle, Washington, USA, May 1998, s. 2545–2548.

[8] Jiang J., Grecos C., Towards an improvement on prediction accuracy in JPEG-LS. *Optical Engineering*, SPIE, 2002, vol. 41, no. 2, s. 335–341.

[9] Lih-Jen Kau, Yuan-Pei Lin, Lossless image coding using a switching predictor with run-length encodings. *Proceedings of IEEE International Conference on Multimedia and Expo*, June 2004, vol. 2, s. 1155–1158.

[10] Kuroki Y., Ueshige Y., Ohta T., An estimation of the predictors implemented by shift operation, addition, and/or substraction. *Proceedings of International Conference on Image Processing 2001*, s. 474–477.

[11] Marcellin M., Gormish M., Bilgin A., Boliek M., An Overview of JPEG-2000. *Proceedings Data Compression Conference*, Snowbird, Utah, March 2000, s. 523–541.
[12] Marusic S., Deng G., A study of two new adaptive predictors for lossless imagecompression. *Proceedings of IEEE 1997 International Conference on Image Processing*, Oct. 1997, vol. 2, s. 286–289.
[13] Marusic S., Deng G., New prediction schemes for lossless coding of fullband and subband images. *Signal Processing: Image Communication*, 1999, vol. 14, s. 869–878.
[14] Matsuda I., Ozaki N., Umezu Y., Itoh S., Lossless coding using Variable Blok-Size adaptive prediction optimized for each image. *Proceedings of 13th European Signal Processing Conference EUSIPCO-05 CD*, September 2005.
[15] Meyer B., Tischer P., Glicbawls – Grey Level Image Compression by Adaptive Weighted Least Squares. *Proceedings of Data Compression Conference 2001*, s. 503.
[16] Meyer B., Tischer P., TMW – a new method for lossless image compression. *Proceedings of International Picture Coding Symposium (PCS97)*, Berlin, Germany, September 1997, s. 533–538.
[17] Meyer B., Tischer P., TMW^{Lego} – An Object Oriented Image Modelling Framework. *Proceedings of Data Compression Conference 2001*, s. 504.
[18] Park S.-G., Delp E. J., Adaptive lossless video compression using an integer wavelet transform. *Proceedings of International Conference on Image Processing ICIP'04*, 24–27 October 2004, vol. 4, s. 2251–2254.
[19] Sayood K., *Introduction to Data Compression*. 2nd ed., Morgan Kaufmann Publ., 2002.
[20] Seemann T., Tisher P., Generalized locally adaptive DPCM. *Department of Computer Science Technical Report CS97/301*, Monash University, Australia, s. 1–15.
[21] Seemann T., Tischer P., Meyer B., History-Based Blending of Image Sub-Predictors. *Proceedings of Picture Coding. Symposium*, Berlin, Germany, 1997, s. 147–151.
[22] Strutz T., Context-Based Adaptive Linear Prediction for Lossless Image Coding. *4th International ITG Conference on Source and Channel Coding*, Berlin, Germany, 28–30 January, 2002, s. 105–109.
[23] Ulacha G., Stasinski R., Dziurzanski P., Olejnik R., Lossless compression system architecture dedicated to Networks on Chips. *Proceedings of Int. Conf. on Signals and Electronic Systems (ICSES'06)*, Łódź, Poland 2006, s. 235–238.
[24] Ulacha G., Stasiński R., On context-based predictive techniques for lossless image compression. *Proceedings of 12th International Workshop on Systems, Signals & Image Processing – IWSSIP 2005*, Chalkida, Greece 2005, s. 345–348.
[25] Ulacha G., Stasiński R., Parameter choice for predictor blending and application in lossless image coding. *Pomiary, automatyka, kontrola*, 2006, no. 7bis, s. 95–97.
[26] Wang H., Zhang D., A linear model and its application in lossless image coding. *Signal Processing: Image Communication*, 2004, vol. 19, s. 955–958.
[27] Weinberger M. J., Seroussi G., Sapiro G., LOCO-I: Lossless Image Compression Algorithm: Principles and Standardization into JPEG-LS. *IEEE Trans. on Image Processing*, vol. 9, No. 8, August 2000, s. 1309–1324.
[28] Wu X., Memon N. D., CALIC – A Context Based Adaptive Lossless Image Coding Scheme. *IEEE Trans. on Communications*, May 1996, vol. 45, s. 437–444.
[29] Ye H., *A study on lossless compression of greyscale images*. Department of Electronic Engineering, La Trobe University, October 2002 (PhD Thesis).
[30] Ye H., Deng G., Devlin J. C., Adaptive linear prediction for lossless coding of greyscale images. *Proc. IEEE Int. Conf. on Image Processing* (CDROM), Vancouver, Canada, September 2000.
[31] Ye H., Deng G., Devlin J. C., A weighted least squares method for adaptive prediction in lossless image compression. *Proc. Picture Coding Symposium*, Saint-Malo, France, 2003, s. 489–493.