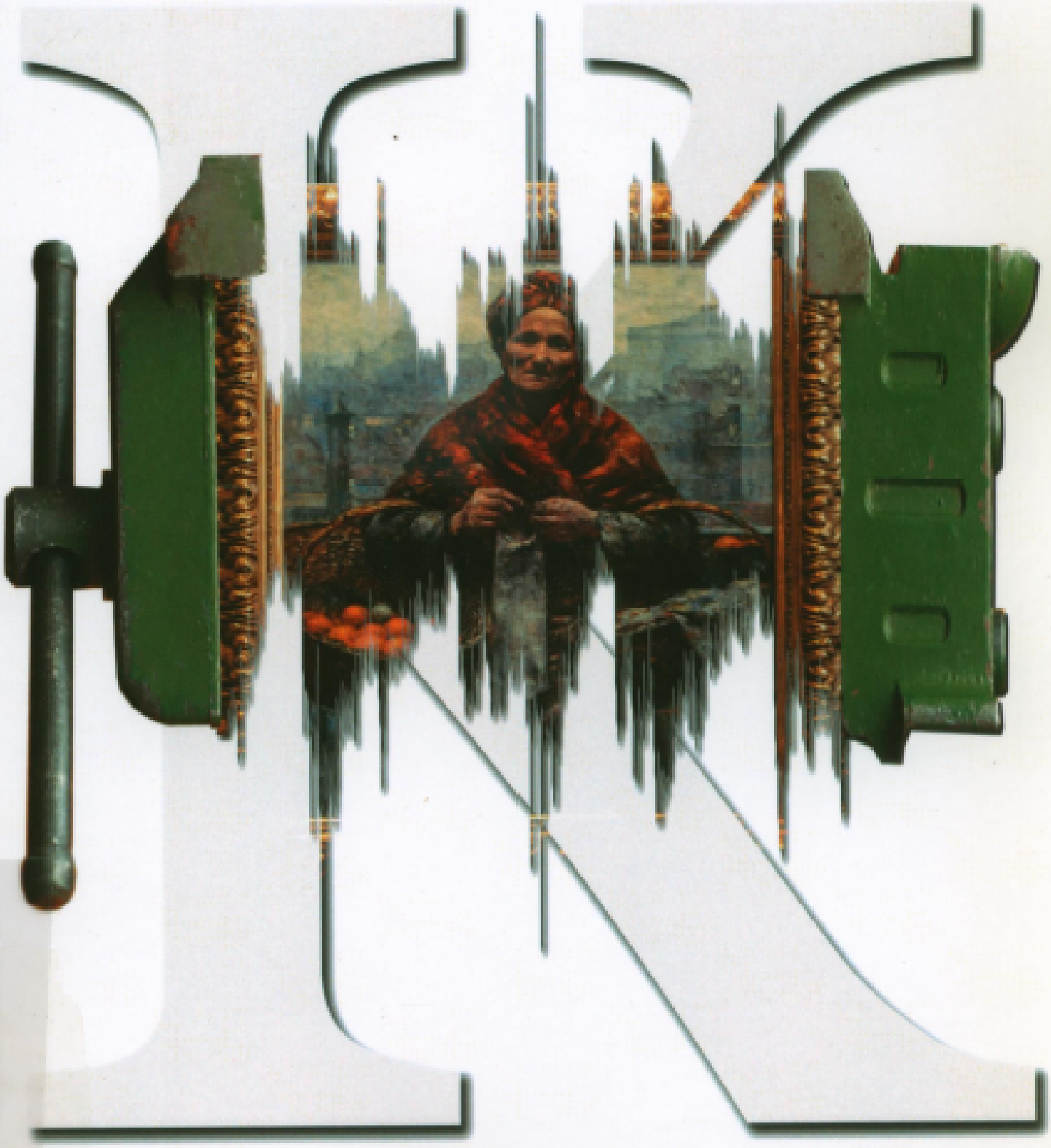
Grzegorz Ulacha

Bezstratna kompresja obrazów cyfrowych z wykorzystaniem mieszania predykcyjnego



Szczecin 2013

Zachodniopomorski Uniwersytet Technologiczny w Szczecinie

Grzegorz Ulacha

Bezstratna kompresja obrazów cyfrowych z wykorzystaniem mieszania predykcyjnego

Szczecin 2013

Recenzenci
ARTUR PRZELASKOWSKI
JAKUB SWACHA

Opracowanie redakcyjne
Katarzyna Mitan

Projekt okładki
Marcin Sadowski

WYDANO ZA ZGODĄ
REKTORA ZACHODNIOPOMORSKIEGO UNIWERSYTETU TECHNOLOGICZNEGO W SZCZECINIE

ISBN 978-83-7663-156-1

WYDAWNICTWO UCZELNIANE ZACHODNIOPOMORSKIEGO UNIWERSYTETU TECHNOLOGICZNEGO W SZCZECINIE
70-311 Szczecin, al. Piastów 50, tel. 91 449 47 60, e-mail: wydawnictwo@zut.edu.pl
Druk: PPH „Zapol” Dmochowski, Sobczyk, Sp.j., 71-062 Szczecin, al. Piastów 42, tel. 91 434 10 21
e-mail: zarzad@zapol.com.pl

Spis treści

Spis ważniejszych oznaczeń

$\mathbf{B} = [b_1, \ldots, b_r]$	–	wektor współczynników b_j predykcji liniowej modelu rzędu r
C_{mix}	–	wartość korekcji mieszanej służąca do usuwania skumulowanego błędu predykcji (patrz podrozdział 4.3)
d	–	oznaczenie w trybie prawie bezstratnym określające maksymalną wartość błędu w zdekodowanym obrazie
$\overline{d}_j$	–	odwrotność odległości euklidesowej między pikselami $P(j)$ a $P(0)$
$e = e(0)$	–	aktualnie kodowany błąd predykcji
$e(j)$	–	błąd predykcji o numerze j odnoszący się do najbliższego sąsiedztwa (patrz rys. 1.1)
L	–	liczba poziomów kwantyzacji
M	–	wykładnik we wzorze (2.9) na odległość Minkowskiego
N_{GoP}	–	liczba klatek stanowiących grupę obrazów (GoP)
n_h	–	liczba neuronów w warstwie ukrytej sieci neuronowej
$O(\ldots)$	–	symbol odnoszący się do złożoności obliczeniowej
$P(j)$	–	wartość piksela o numerze j odnosząca się do najbliższego sąsiedztwa (patrz rys. 1.1, na którym umieszczono numerację pikseli)
Q	–	odpowiednio zdefiniowany (w metodach OLS, WLS, AdNN$_+$) obszar treningowy przedstawiony na rys. 6.1
r	–	rząd predykcji
W	–	wysokość obszaru treningowego Q (patrz rys. 6.1)
α_i	–	waga istotności i-tego subpredyktora
δ	–	suma m kolejnych wartości $\overline{d}_j$ wykorzystywana do celów normalizacji
μ	–	współczynnik adaptacji w metodach RLS, NLMS, ALCM$_+$, CoBALP$_+$
φ	–	wartość ogranicznika zabezpieczającego przed przesterowaniem procesu adaptacji w metodach RLS, NLMS, ALCM$_+$, CoBALP$_+$
$a := b$	–	operacja przypisania liczby b do zmiennej a
$\overline{a,b}$	–	domknięty przedział liczb całkowitych od a do b

1. Wprowadzenie

1.1. Zdefiniowanie problemu badawczego

W dobie w pełni cyfrowej struktury obróbki obrazów cyfrowych i sekwencji wideo istotnym problemem są wysokie wymagania pamięciowe związane z przechowywaniem danych wizyjnych. Zmniejszenie wymagań pamięciowych jest możliwe dzięki kompresji. Na przykład zapis 60 minut filmu jakości telewizyjnego standardu PAL (25 klatek na sekundę o rozdzielczości SD 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).

Kompresję można podzielić na stratną i bezstratną, przy czym w tej pracy skupiono się głównie na tym drugim typie. Do istotnych zastosowań bezstratnej kompresji obrazów i sekwencji wideo należy między innymi archiwizacja obrazów medycznych 2D, 3D oraz 4D (trójwymiarowe sekwencje wideo) [52, 97, 100, 106, 144], astronomicznych, a także kompresja zdjęć satelitarnych [21, 69]. Ponadto często tryb bezstratny jest wymagany na etapie graficznej obróbki zdjęć, materiałów reklamowych oraz przy produkcji audycji telewizyjnych, filmów (ang. *post-production* [7]) itp. W takiej sytuacji nie można korzystać ze stratnych wersji metod kompresji, takich jak JPEG, JPEG2000 (dla obrazów statycznych) czy MPEG2, MPEG4 (dla sekwencji wideo). Choć te standardy mają odpowiednie tryby bezstratne, to nie można ich zaliczyć do kategorii najwydajniejszych, co zostało przebadane na wstępnym etapie, przed postawieniem tezy zawartej w tej pracy. Pewnym kompromisem między kodowaniem bezstratnym a stratnym jest tryb prawie bezstratny, pozwalający na zdefiniowanie maksymalnych błędów w obrazie zdekodowanym (patrz podrozdział 10.2).

Nowym rodzajem zastosowań jest technologia *slow motion video*, służąca do zapisu nagrań z eksperymentów naukowych z dokładnością nawet do kilkudziesięciu tysięcy klatek na sekundę. Na przykład nagranie sekundy sekwencji wideo (sygnał luminancji osiem bitów/piksel) przy 1000 klatkach na sekundę i rozdzielczości SD 720×576 pikseli wymaga 3955 MB pojemności pamięci. Obecnie kamery w trybie *slow motion* zapisują obraz bez kompresji i czas trwania sekwencji jest ograniczony maksymalną pojemnością pamięci kamery. Nagrywanie i archiwizacja plików o większych rozmiarach staje się problemem, który może zostać rozwiązany przez wprowadzenie szybkich sprzętowych zrównoleglonych realizacji kompresji np. w trybie prawie bezstratnym, co w połączeniu z macierzą wydajnych dysków SSD, może pozwolić na zwiększenie funkcjonalności stanowiska badawczego wykorzystującego szybkozmienne zjawiska fizyczne.

W przypadku stosowania współczesnych metod kompresji wykorzystuje się zwykle dwa etapy: dekompozycję danych oraz kompresję za pomocą jednej z wydajnych metod entropijnych, wśród których najefektywniejsze to kodowanie arytmetyczne i kodowanie Huffmana [98]. Problemem badawczym poruszanym w tej pracy będzie zatem analiza możliwości dalszego wzrostu efektywności bezstratnej kompresji obrazów w odniesieniu do istniejących

rozwiązań znanych z literatury. Rozpoczynając prace nad rozwojem nowych metod, należało bowiem odpowiedzieć na pytanie, czy słuszna jest opinia o tym, że zdecydowana większość problemów badawczych w zakresie bezstratnej kompresji obrazów została już rozwiązana? Takie argumenty bowiem pojawiały się w rozmowach autora z naukowcami, a potwierdzać to miała znacząco spadająca liczba publikacji dotyczących zagadnienia bezstratnej kompresji obrazów.

Autor podjął się trudnego zadania, chcąc opracować nowe metody z wykorzystaniem najlepszych cech rozmaitych już istniejących rozwiązań. Projektowane tu metody będą wykorzystywać możliwość wspólnego stosowania wielu technik dekompozycji obrazu, jak i rozbudowaną metodę adaptacyjnego kodowania arytmetycznego.

1.2. Istniejące rozwiązania bezstratnej kompresji obrazów

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 za jedną z najefektywniejszych metod uznaje się zaprezentowaną w roku 1996 CALIC (ang. *Context Based Adaptive Lossless Image Coding*) [141]. Na owe czasy metoda ta okazała się zbyt wymagająca obliczeniowo w porównaniu z metodą LOCO-I, której modyfikacja stała się standardem JPEG-LS [136].

Wśród najefektywniejszych algorytmów o wysokiej złożoności implementacyjnej można wyróżnić prace trzech zespołów badawczych: metoda TMW (1997) [86] i jej późniejsze rozwinięcie TMW^{LEGO} (2001) [87], WAVE-WLS (2002) [147] oraz najnowsza wersja MRP 0.5 zaprezentowana pod nazwą 'VBS & new-cost' (2005) [78]. Zakodowanie jedego obrazu z użyciem każdej z tych propozycji wymaga wielu minut (lub nawet godzin, jeśli nie dysponujemy najwydajnieszym 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. JPEG2000 [72]), jak i *interframe* [91]. Jednak uzyskiwane wyniki nie dorównują najlepszym metodom predykcyjnym. Podobnie wygląda sytuacja z kodowaniem hierarchicznym wykorzystanym w metodzie typu HINT i pierwotnej wersji kodera CALIC [138]. Z tego względu w tej pracy cechy charakterystyczne tych metod nie zostaną omówione. Informacje o różnych podejściach do kodowania bezstratnego można znaleźć w pracach przeglądowych [18, 26, 82]. Obecnie dość powszechnie stosowanym formatem jest PNG ze względu na jego uniweralność w kodowaniu zarówno obrazów naturalnych, jak i sztucznie generowanych [93, 105]. Jednak jego efektywność jest niższa od metod takich jak JPEG-LS czy CALIC. Wersja PNG-crush koduje obraz kolejno dla wszystkich możliwych ustawień parametrów PNG, wybierając najlepszy rezultat. Wyniki działań tego oprogramowania, wraz z porównaniem efektywności innych metod znanych z literatury oraz opracowanych przez autora tej pracy, zaprezentowano w podsumowującej tab. 9.10. Warto przy tym zaznaczyć, że techniki oparte na kodowaniu słownikowym, takie jak choćby PNG czy archiwizery ogólnego

przeznaczenia, np. WinRAR, sprawdzają się wyjątkowo dobrze w przypadku kodowania obrazów generowanych w sposób sztuczny. Jednak w odniesieniu do obrazów naturalnych pozyskiwanych z kamer i aparatów cyfrowych zdecydowaną przewagę uzyskują metody wykorzystujące różne techniki predykcyjne.

Spośród różnych typów modelowania największą elastycznością cechuje się metoda mieszania predyktorów BP (ang. *Blending Predictors*), użyta choćby we wspomnianych metodach TMW oraz WAVE-WLS. Pomysł mieszania predyktorów (stanowiących zbiór subpredyktorów), przedstawiony szerzej w pracy [102], był rozwijany między innymi przez G. Denga oraz H. Ye i prezentowany np. w pracach [27] i [147]. Szczegółowy opis metody znaduje się w rozdziale 9. Jest to metoda konkurencyjna względem selekcji aktualnie najlepszego predyktora [74].

Po dogłębnej analizie cech charakterystycznych obecne metody bezstratnej kompresji obrazu (sygnału luminancji) można sklasyfikować jako metody z (analiza na podstawie czasów kodowania/dekodowania obrazu Lennagrey 512×512 pikseli odnosi się do procesora Pentium4 2,8 GHz):

– bardzo szybkim kodowaniem i dekodowaniem, czas liczony w milisekundach, np. JPEG-LS [136];

– szybkim kodowaniem i dekodowaniem, czas w okolicach 1 s, np. CALIC [141], JPEG2000 [72];

– akceptowalnym czasem kodowania i dekodowania (kilkanaście/kilkadziesiąt sekund), np. GLICBAWLS [85], SWAP [55];

– trudno akceptowalnym czasem kodowania (kilkadziesiąt minut), np. TMW^{LEGO} [87], MRP [78], WAVE-WLS [147].

Pojęcie akceptowalności odnosi się do obecnie projektowanych metod, które po etapie testowania, mogłyby zostać zaproponowane (z wyprzedzeniem co najmniej pięcioletnim) jako przyszłe standardy kodowania. Należy bowiem uwzględnić wzrost wydajności współczesnych komputerów na przełomie kilku lat, dotyczy to takich parametrów jak częstotliwość zegara, wielkość pamięci podręcznych, liczba potoków oraz konstrukcja wielordzeniowa odgrywająca wśród komputerów osobistych coraz większą rolę.

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 charakteryzuje się dłuższym czasem kodowania względem dekodowania, co można uznać za przykład metody asymetrycznej czasowo. Wśród metod o czasie kodowania nieprzekraczającym sekundy nie ma większego znaczenia identyfikacja asymetryczności. Dopiero w przypadku długich czasów kodowania warto podkreślić fakt, że metoda jest asymetryczna, a czas dekodowania może być akceptowalnie mały (np. kilka sekund). Jest to ważna cecha, gdyż operację kodowania najczęściej przeprowadza się raz, a dekodowania wielokrotnie. Wśród wymienionych wcześniej trzech najefektywniejszych metod TMW^{LEGO} oraz MRP należą do kategorii asymetrycznych (ze względu na iteracyjne poszukiwanie parametrów na etapie kodowania), natomiast WAVE-WLS jest metodą symetryczną czasowo (adaptacyjna metoda predykcyjna o dużej złożoności implementacyjnej kodowania i dekodowania każdego piksela).

Istnieją jeszcze i inne możliwości kategoryzacji, np. ze względu na rodzaj interpretacji wartości pikseli. Traktuje się je jako zbiór bitów, co w przypadku obrazów o wielu odcieniach (w tym wielobarwnych zapisywanych z użyciem trzech składowych barw) prowadzi do powstania wielu map bitowych kodowanych osobno lub z uwzględnieniem odpowiednich schematów zależności danych [93, 98]. Jednak te zagadnienia wykraczają poza ramy niniejszej pracy, gdyż autor postanowił się skupić na odpowiednio określonej koncepcji jednoczesnego łączenia wielu metod, wybierając te, które wspólnie będą mogły prowadzić do osiągnięcia postawionego celu.

1.3. Cel i zakres pracy

Celem niniejszej pracy **jest opracowanie metody opartej na mieszaniu predykcyjnym, która umożliwi kompresję obrazów cyfrowych uzyskującą średnią efektywność na poziomie wyższym od efektywności klasycznych rozwiązań znanych z literatury.** Należy przy tym zaznaczyć, iż metoda ukierunkowana będzie na kodowanie obrazów naturalnych uzyskiwanych z aparatów i kamer cyfrowych. Rezultaty kompresji obrazów generowanych w sposób sztuczny odrywają drugorzędną rolę przy wyznaczaniu wymagań stawianych projektowanej metodzie.

Definiując taki cel, badacz powinien odpowiedzieć na wciąż nurtujące naukowców pytanie, czy wynik poprawiający rezultat o zaledwie kilka procent względem rozwiązań konkurencyjnych jest wart poświęcania wielu lat badań? A także, czy uzyskanie takiej niewielkiej poprawy wysokim wzrostem złożoności implementacyjnej ma przełożenie na przyszłe praktyczne rozwiązania? Odpowiedzi na te pytania zawsze dzielą grono naukowców na dwie grupy i trudno pogodzić ich argumenty. Jednak osiągnięcie celu pracy dzięki wzrostowi wydajności sprzętu komputerowego może w przyszłości zaowocować wieloma rozwiązaniami praktycznymi, a część pomysłów opracowanych przez autora i zaprezentowanych w publikacjach naukowych może okazać się inspiracją dla kolejnych badaczy udoskonalających własne metody (co już autor mógł zaobserwować w literaturze).

Ze względu na specyfikę badań pojawiających się w literaturze przeprowadzone analizy odnosić się będą do sygnału luminancji, czyli obrazów w odcieniach szarości, przy czym przyjmuje się tu ośmiobitową skalę składowej, czyli 256 poziomów luminancji. Badania dotyczące obrazów barwnych i sekwencji wideo będą miały charakter jedynie uzupełniający. Ich celem było wykazanie, że główne metody zaproponowane w pracy sprawdzają się nie tylko dla sygnału luminancji.

Ważnym aspektem przedstawionym w tej pracy jest wykazanie, że osiągnięcie postawionego celu, którym jest wzrost efektywności metody kompresji, nie musi oznaczać nieproporcjonalnego wzrostu złożoności już istniejących rozwiązań.

W pracy formułuje się tezę, że **dzięki mieszaniu predykcyjnemu możliwe jest jednoczesne efektywne wykorzystanie wielu metod łączonych ze sobą zarówno w sposób szeregowy, jak i równoległy**. Tego typu podejście pozwoli jednocześnie na możliwość elastycznego dopasowywania efektywności projektowanych odmian metody do założeń dotyczą-

cych ograniczeń czasowych. Jest to zatem pewnego rodzaju kompleksowe podejście do określenia reguł budowania metod bezstratnej kompresji obrazów z wykorzystaniem wielu bloków funkcjonalnych, z których każdy może sam w sobie być metodą kompresji znaną z literatury lub opracowaną przez autora tej pracy.

Wielokrotnie autor spotykał się z opinią, że temat bezstratnej kompresji obrazów został wyczerpany i zajmowanie się tą kwestią nie ma sensu, co niejako dało się uzasadnić w odniesieniu do tego tematu dużym spadkiem w ostatnich latach liczby publikacji w światowej literaturze. Takie opinie tym bardziej utwierdzały autora niniejszej pracy w przekonaniu, iż osiągnięcie celu będzie zadaniem niezwykle trudnym, lecz z tym większym zapałem należało przystąpić do pracy, gdyż po dogłębnej analizie literatury dało się wskazać kilka luk, które były warte przebadania.

Aby osiągnąć cel, należało zastosować dwa przeciwstawne podejścia. Po pierwsze – aby przeciwstawić się tezie, że wszelkie koncepcje zostały już rozważone, należało odrzucić ogólnie znane zasady, a zwłaszcza podręcznikowe metody optymalizacyjne, tworząc zupełnie nowe pomysły, czasem pozbawione podstaw teoretycznych, a następnie sprawdzać je w praktyce, bo tylko otwartość umysłu potrafi ukazać nowe spojrzenie na dotychczasowy zbiór zaakceptowanych i uznanych metod. Jak można się domyślać, większość takich amatorskich pomysłów okazywała się małowartościowa i nie ma w tej pracy miejsca na ich opis, jednak niektóre przyniosły efekt i tylko te zostały tu zaprezentowane, jak choćby wykorzystanie algorytmów genetycznych do tworzenia efektywnych modeli predykcyjnych (patrz podrozdział 2.5) czy też mieszana metoda korekcji skumulowanego błędu predykcji (podrozdział 4.3).

Należało wykorzystać także drugie podejście, najczęściej uznawane za pierwszorzędne, mające przybliżyć osiągnięcie celu przez analizę szerokiej literatury tematu. Trudno bowiem zignorować zestaw metod podanych wręcz na tacy przez najlepszych badaczy w danej dziedzinie. Analiza literaturowa i wstępne badania przydatności opisanych w niej metod pozwoliły nakreślić odpowiedni kierunek i schemat właściwych badań, których rezultaty opisano w dalszej części tej pracy. W tym przypadku przy opisie również pominięto wiele podejść, które okazały się mało przydatne w dążeniu do osiągnięcia głównego celu, zadaniem bowiem tej pracy nie jest powielenie wiedzy opublikowanej już w tak dobrych podręcznikach, jak choćby [31, 93]. Wyjątek stanowią podrozdziały od 1.4 do 1.6 zawierające podstawy, niezbędne do prawidłowego zrozumienia kolejnych rozdziałów tej pracy, w których opisano jedynie najlepsze i mogące się nawzajem uzupełniać metody kompresji. Przy czym metody te zostały w znacznym stopniu rozbudowane, a następnie przebadane, aby ukazać ich zalety i wady oraz zastosować w docelowym mechanizmie, czyli **metodzie mieszania predykcyjnego**, która w umiejętny sposób potrafi wykorzystać cechy poszczególnych metod, aby ich wspólne połączenie pozwoliło uzyskać wysoką efektywność kompresji. W pracy skupiono się głównie na etapie dekompozycji danych z wykorzystaniem różnych modeli predykcyjnych (podobne proporcje rozłożenia akcentów występują również w literaturze), choć prace nad rozbudowanym koderem arytmetycznym były równie ważnym krokiem pozwalającym się zbliżyć do osiągnięcia założonego celu.

Drugorzędnym celem pracy było stworzenie elastycznej możliwości równoległego wykorzystania wielu metod, pozwalającej na regulację złożoności implementacyjnej kodera i dekodera w taki sposób, aby można było uzyskać stosunkowo wysoką efektywność kompresji nie tylko dzięki użyciu najbardziej czasochłonnych obliczeń. Postępując w ten sposób, udało się zaprojektować zestaw kilku metod o rosnącej efektywności przy jednocześnie nieproporcjonalnym, jak się można domyślać, wzroście czasu kodowania.

Trudnym zadaniem było ustalenie najwłaściwszej kolejności opisywanych tematów, gdyż między poszczególnymi zagadnieniami występuje wiele zależności. Rozpoczynając opisy dotyczące metod predykcyjnych, trudno je w połowie przerwać, aby nie zaburzyć struktury pracy. Z drugiej jednak strony nie można zostawić na sam koniec opisu bloku kodowania błędów predykcji, w niektórych bowiem miejscach pracy pojawią się nawiązania do tego właśnie bloku. Dlatego pracę podzielono na cztery główne bloki tematyczne. W pierwszym z nich, składającym się z rozdziałów od 1 do 7, przedstawiono wszystkie podstawowe podejścia dotyczące modelowania predykcyjnego.

W rozdziale 2 omówiono podstawowe zasady projektowania stałych i statycznych modeli predykcyjnych, czyli takich, których współczynniki nie zmieniają się w czasie kodowania całego obrazu. Zdefiniowano problem rozbieżności między podręcznikową sugestią minimalizacji błędu średniokwadratowego a kryterium minimalizacji opartym na funkcji entropii w odniesieniu do projektowania modeli predykcyjnych. Ponadto przedstawiono kilka autorskich prób wybiórczego poszukiwania współczynników predykcji, w tym z użyciem algorytmów genetycznych.

Rozdział 3 dotyczy zasad adaptacyjnej predykcji z przełączanym modelem. Zaprezentowano w nim podstawowe i zmodyfikowane metody predykcji wykorzystujące zasady przełączania kontekstów. Opracowano także autorską metodę podziału na konteksty główne i uzupełniające, która stała się podstawą do rozwinięcia predykcyjnych metod opisanych w kolejnych rozdziałach.

W rozdziale 4 omówiono zasady dotyczące adaptacyjnej metody usuwania składowej stałej, zwanej także korekcją błędu predykcji skojarzonego z odpowiednim kontekstem. Wykorzystując różne rozwiązania, opracowano autorską metodę, w której zastosowano mieszanie ośmiu różnych wartości w celu ustalenia ostatecznej wartości składowej stałej dla aktualnie kodowanego piksela. Jako praktyczny przykład potwierdzający zalety metody mieszanej w stosunku do pojedynczych metod korekcji błędu predykcji przedstawiono metodę statycznej predykcji liniowej z przełączaniem kontekstowym.

Następne trzy rozdziały dotyczą adaptacyjnego podejścia do wyznaczania wartości przewidywanych kolejno kodowanych pikseli obrazu. W rozdziale 5 przedstawiono metody prostej adaptacji współczynników predykcji, omawiając podstawowe i rozbudowane przez autora pracy metody LMS, ALCM, CoBALP. Rozdział kończy się propozycją szybkiej i efektywnej metody M-LMS, która jest połączeniem rozbudowanych metod $ALCM_+$ i $CoBALP_+$. W rozdziale tym podjęto także próbę odpowiedzi na pytanie, dlaczego w odniesieniu do obrazów cyfrowych klasyczna metoda LMS charakteryzuje się małą efektywnością przy rastrowym skanowaniu obrazów.

Rozdział 6 zawiera adaptacyjne metody wyznaczania współczynników predykcji charakteryzujące się wysoką złożonością implementacyjną. Omówiono w nim metody RLS_+, OLS, WLS i jej odmiany AVE-WLS. Skupiono się także na analizie ich złożoności. Zaprezentowano też autorskie podejście, które wprowadza metodę NLMS jako drugi blok modelowania predykcyjnego, czego wcześniej nie stosowano w literaturze dotyczącej bezstratnej kompresji obrazów cyfrowych.

W rozdziale 7 zaprezentowano metodę wyznaczania wartości predykcji dzięki użyciu sieci neuronowych. Opisano istniejące rozwiązania i na ich podstawie zaprojektowano efektywną metodę opartą na wielowarstwowej sieci perceptronowej, badając przy tym wpływ poszczególnych parametrów sieci na stopień kompresji.

Drugi blok tej pracy to rozdział 8, w którym omówiono adaptacyjny koder arytmetyczny z przełączaniem kontekstów, który określa się jako drugi etap bezstratnego kodowania obrazów występujący po etapie dekompozycji danych. W autorskiej metodzie zdecydowano się na rozwiązanie w pełni adaptacyjne, z jak najmniejszą liczbą parametrów inicjalizujących. Koder został zaprojektowany w taki sposób, aby bit znaku i wartość bezwzględna błędu predykcji były osobno kompresowane.

Wykorzystanie wiedzy zawartej w pierwszych 8 rozdziałach pozwoliło na zaprojektowanie i omówienie w trzecim bloku tej pracy (rozdział 9) metody kompresji, która opiera się na możliwości jednoczesnego wykorzystania wielu metod predykcyjnych celem ustalenia końcowej przewidywanej wartości piksela. Jest to metoda mieszania predykcyjnego, której różne warianty zademonstrowano w poszczególnych podrozdziałach rozdziału 9. Metody określone mianem Blend-20, Blend-24 oraz Blend-25 spełniają założenie przyjęte w celu pracy i mówiące o tym, aby średnia bitowa dla zbioru testowanych obrazów była niższa od wszystkich pozostałych wyników uzyskanych metodami klasycznymi znanymi z literatury.

Ostatni dający się wydzielić w tej pracy blok to opis w rozdziale 10 praktycznych zastosowań uproszczonej wersji mieszania predykcyjnego do kodowania obrazów i sekwencji wideo. Przy kodowaniu sekwencji wideo, oprócz przestrzennej zależności występującej między sąsiednimi pikselami kodowanej klatki, istnieje także zależność czasowa (ang. *temporal correlation*), czyli podobieństwo między kolejno występującymi po sobie klatkami sekwencji [7, 79, 98], a także zależność spektralna między składowymi barw *R*, *G* i *B* [77]. Dlatego też, jako materiał uzupełniający, w rozdziale 10 omówiono aspekty kodowania barwnego oraz kodowania sekwencji wideo. Zaprezentowano także zastosowanie autorskiej metody Blend-24 do trybu prawie bezstratnego, w którym przy średnich bitowych wyższych od 1 bitu na piksel (dla obrazów w odcieniach szarości) metoda ta daje lepsze rezultaty w porównaniu ze stratnym koderem JPEG2000.

Rozdział 11 zawiera podsumowanie z wykazem własnych rozwiązań autora opisanych w tej pracy. Przedstawiono też kierunki potencjalnych badań, których rezultaty według autora mogą pozwolić na uzyskanie dalszego wzrostu efektywności opracowanych do tej pory rozwiązań w dziedzinie bezstratnej i prawie bezstratnej kompresji obrazów cyfrowych.

1.4. Podstawy modelowania obrazów i sekwencji wideo

W przypadku sygnałów dwuwymiarowych, jakimi są obrazy, możemy zaobserwować podobieństwo między sąsiednimi pikselami, czyli zależność przestrzenną/płaszczyznową sąsiedztwa (ang. *spatial correlation*). Modelowanie danych sprowadza się w takim przypadku do próby usunięcia jak największej ilości informacji wzajemnej występującej między sąsiadującymi ze sobą pikselami, jest to zatem pewien proces dekompozycji danych. 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 (ang. *zero order entropy*)) może być wyznaczona ze wzoru:

$$H = -\sum_{i=e_{\min}}^{e_{\max}} p_i \cdot \log_2 p_i \,, \tag{1.1}$$

gdzie p_i jest prawdopodobieństwem wystąpienia symbolu z alfabetu, o indeksie i. Za alfabet w tym przypadku można uznać zbiór liczb całkowitych o wartościach z przedziału od $e_{\min}$ do $e_{\max}$. Entropia ta informuje nas o minimalnej średniej liczbie bitów potrzebnej do zakodowania jednego symbolu z użyciem jednej z metod statycznego kodowania entropijnego (np. kodu Huffmana, Golomba czy kodu arytmetycznego rzędu zerowego) [34, 98]. W praktyce wzór (1.1) jest jedynie pewnym teoretycznym przybliżeniem średniej bitowej (jej ograniczeniem dolnym), wyrażanej jako stosunek całkowitej liczby bitów niezbędnych do zakodowania obrazu do liczby wszystkich pikseli obrazu. Entropia bezwarunkowa, jako uproszczona funkcja celu, będzie wykorzystywana w początkowych rozdziałach do porównawczego szacowania poszczególnych rozwiązań. Użycie kontekstowego kodera arytmetycznego (patrz rozdział 8) pozwala uzyskać średnią bitową niższą od entropii rzędu zerowego, bowiem dekompozycja danych na etapie predykcji nie w pełni zdoła usunąć wzajemne zależności między pikselami. Koder tego typu będzie wykorzystywany na etapie dalszych badań nad bardziej złożonymi metodami kompresji obrazu.

Jedną z propozycji usuwania informacji wzajemnej kodowanych obrazów jest zastosowanie predykcji liniowej z odpowiednim doborem sąsiednich pikseli oraz rzędu predykcji. Ze względu na kierunek kodowania obrazu założony w tej pracy (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 znajdujących się powyżej oraz po lewej stronie aktualnie kodowanego (dekodowanego) piksela – określamy to zasadą przyczynowości. Korzystając z założenia o korelacji malejącej wraz ze wzrostem odległości pikseli, możemy ponumerować sąsiadów kodowanego piksela zgodnie z rosnącą odległością metryczną (euklidesową) $\sqrt{(\Delta x_j)^2 + (\Delta y_j)^2}$ ich środków. Numeracja pikseli o tej samej odległości jest wyznaczana zgodnie z ruchem wskazówek zegara. Pozwala to na uzyskanie jednowymiarowej dziedziny sygnału, co ułatwia matematyczny zapis wielu wzorów i zależności znanych z literatury, dotyczących sygnałów jednowymiarowych. Rysunek 1.1 obrazuje 30 ponumerowanych najbliższych sąsiadów kodo-

wanego piksela *x*, gdzie dany *j*-ty numer wskazuje na piksel o wartości *P*(*j*). Teoretycznie im wyższy numer piksela, tym mniejsze jego znaczenie dla poprawy efektywności kodowania.

			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*				

Rys. 1.1. Numeracja pikseli sąsiedztwa

Bardziej złożone kierunki kodowania (w tym po krzywej Hilberta) na ogół nie przynoszą dla kategorii obrazów naturalnych lepszego stopnia kompresji [82], zwłaszcza gdy używa się złożonych metod mieszania predykcyjnego, które zostaną szczegółowo opisane w rozdziale 9.

Do modelowania stosuje się różne techniki, przy czym najczęściej w bezstratnej kompresji obrazów jest używany typowy predyktor liniowy rzędu *r*, który jest wartością przewidywaną aktualnie kodowanego piksela *x* na podstawie *r* sąsiednich (zgodnie z zachowaniem zasad przyczynowości znanych koderowi i dekoderowi) pikseli. Zasady te odnoszą się do kodowania kolejnych wierszy obrazu od góry do dołu, a w ramach wiersza kolejnych pikseli od lewej strony do prawej. Predyktor liniowy postać:

$$\hat{x} = \sum_{j=1}^{r} b_j \cdot P(j), \tag{1.2}$$

gdzie elementy *P*(*j*) są wartościami pikseli z najbliższego otoczenia (sąsiedztwa) aktualnie kodowanego piksela *x*, a elementy b_j to współczynniki predykcji tworzące wektor **B** [98]. Często w rozwiązaniach praktycznych stosuje się założenie, że suma współczynników takiego modelu powinna wynosić 1 (jest to warunek nieobciążania estymatora predykcji). Przy tym założeniu oraz ośmiobitowej skali zapisu odcieni szarości danych wejściowych wartość przewidywana $\hat{x} \in <-255; 255>$. Przykładowy model kontekstu rzędu $r = 10$ pocieniowano, przedstawiając go na rys. 1.1. Oprócz modeli liniowych stosuje się wiele innych rozwiązań nieliniowych, jak choćby modele z przełączaniem kontekstu omówione w podrozdziale 3.1 czy też metody oparte na sieciach neuronowych zaprezentowane w rozdziale 7.

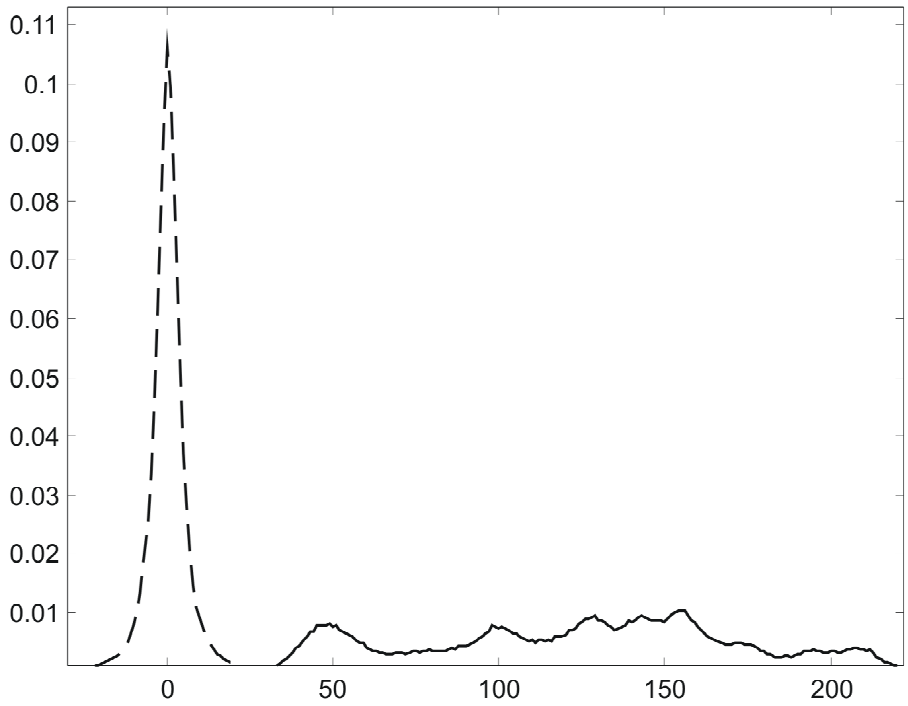
Użycie predyktora liniowego (zwanego także liniowym modelem predykcyjnym) lub nieliniowego pozwala kodować tylko błędy predykcji, czyli różnice *e* między wartością rzeczywistą a przewidywaną (zaokrągloną do najbliższej liczby całkowitej, wartość przewidywana bowiem należy do zbioru liczb rzeczywistych), które najczęściej są niewielkimi wartościami oscylującymi w pobliżu zera:

$$e = x - [\hat{x}]. \tag{1.3}$$

W ten sposób uzyskujemy obraz różnicowy, w którym rozkład błędów *e* badacze oceniają jako zbliżony do rozkładu Laplace'a, co pozwala na efektywne ich kodowanie z użyciem

jednej ze statycznych lub adaptacyjnych metod entropijnych. Przyjmuje się bowiem, że im bardziej niezrównoważone są wartości poszczególnych prawdopodobieństw (duża dysproporcja między prawdopodobieństwami najczęściej i najrzadziej występującymi), tym lepszą efektywność kodowania można uzyskać. Na rysunku 1.2 przedstawiono porównanie rozkładów dla obrazu Lennagrey (linia ciągła) oraz różnicowego (linia przerywana), powstałego z użyciem predyktora trzeciego rzędu.

W dalszej części pracy wartość x aktualnie kodowanego piksela zostanie oznaczona jako $P(0)$, co znacznie ułatwi opis poszczególnych metod odwołujących się do sygnałów dwuwymiarowych.



Rys. 1.2. Porównanie rozkładów dla obrazu Lennagrey (linia ciągła) oraz różnicowego (linia przerywana)

Należy zaznaczyć, iż w sytuacji, gdy kodowany piksel $P(0)$ znajduje się blisko jednej z krawędzi obrazu, część pikseli sąsiedztwa może się znaleźć poza kodowanym obrazem. Najrozsądniejszym (z punktu widzenia przeprowadzonych licznych eksperymentów) sposobem rozwiązania tego problemu jest wprowadzenie dodatkowych marginesów okalających obraz. Aby były one identyczne po stronie kodera i dekodera, należy przyjąć we wszystkich metodach predykcji następującą zasadę postępowania: Lewy górny piksel jest zapisywany do pliku wynikowego bez zmian. Pozostałe piksele w pierwszym wierszu są kodowane różnicowo, z wykorzystaniem lewego sąsiada $e = P(0) - P(1)$, a pozostałe piksele w pierwszej kolumnie z wykorzystaniem górnego sąsiada $e = P(0) - P(2)$. Lewa pierwsza kolumna jest kopiowana odpowiednio dużą liczbę razy jako lewostronny margines, podobnie dzieje się z prawą (ostatnią) kolumną, którą powielamy jako prawy margines. Następnie pierwszy wiersz zostaje skopiowany nad górną częścią powiększanego obrazu, tak że tworzy jego górny margines.

1.5. Redukcja zakresu odcieni

Dla obrazów o niewielkiej liczbie odcieni szarości istnieje wiele numerów odcieni, które nie występują w obrazie, a mogą być wyznaczane jako wartość przewidywana. Aby temu zapobiec, można zredukować zakres odcieni, zapisując w nagłówku pliku informacje o tym, które odcienie wystąpiły w danym obrazie. Kolejnym odcieniom występującym w obrazie przypisujemy numery od 0 do $k - 1$, gdzie k jest całkowitą liczbą odcieni pojawiających się w kodowanym obrazie [92]. Rysunek 1.3 przedstawia przykład redukcji dziewięciu odcieni występujących w obrazie z palety 16 odcieni na ciągły obszar dziewięciu zmodyfikowanych odcieni (symbol X oznacza, że odcień o tym numerze nie wystąpił w danym obrazie). W ostatnim wierszu wartości indeksów przy numerze zmodyfikowanego odcienia informują o numerze odcienia pierwotnego. W nagłówku pliku umieszcza się zestaw 256 bitów znacznikowych z k jedynkami (reprezentującymi odcienie występujące w obrazie) oraz $256 - k$ zerami (reprezentującymi odcienie, które nie wystąpiły w obrazie). Informacja ta umożliwia dekoderowi prawidłowe odtworzenie numerów pierwotnych odcieni. Dołączenie takiego nagłówka do pliku przekłada się na wzrost średniej bitowej o 0,00098 bitu na piksel (dla obrazów o rozdzielczości 512×512).

Wejściowy odcień	0	1	X	3	4	X	X	7	8	9	10	X	X	X	14	X
Znacznik nagłówka	1	1	0	1	1	0	0	1	1	1	1	0	0	0	1	0
Wyjściowy odcień	0_0	1_1	2_3	3_4	4_7	5_8	6_9	7_{10}	8_{14}	X	X	X	X	X	X	X

Rys. 1.3. Przykładowy schemat redukcji zakresu odcieni

Pomysł ten pozwala dokonywać predykcji na zredukowanym zakresie zakresu odcieni. Na podstawie eksperymentów można uznać, iż użycie tej metody staje się opłacalne dla obrazów mających poniżej 200 odcieni (warunek ten, jak i przedstawiony sposób opisu danych nagłówkowych stanowią autorski wkład w poprawę efektywności metod predykcyjnych).

Redukcja zakresu odcieni zostanie użyta we wszystkich omawianych metodach kodowania, z wyjątkiem metody Blend-13, opisanej w podrozdziale 9.3. Ponadto redukcji tej nie stosuje się w trybie prawie bezstratnym (patrz podrozdział 10.2), bo nie gwarantuje ona uzyskania maksymalnego i nieprzekraczalnego, z góry zaplanowanego błędu d.

1.6. Domyślne założenia przyjęte w pracy

W pracy wykorzystuje się wiele domyślnych określeń i oznaczeń. W przypadku projektowanych metod, jeśli nie podano inaczej, czas kodowania i dekodowania nie zależy od zawartości danych obrazowych, ale jedynie liniowo od liczby pikseli obrazu. Z tego też powodu pojawiają się wyniki pomiarów czasowych dla obrazu Lennagrey o rozdzielczości 512×512 pikseli, będące w pełni reprezentatywną informacją związaną z określoną techniką kodowania. Czas kodowania oparto na wynikach pomiarów dla procesora Pentium4 2,8 GHz.

Wybór tego procesora został dokonany już w 2006 roku i nie stosowano nowszych rozwiązań w miarę rozwoju technologicznego po to, aby publikowane w kolejnych latach artykuły omawiające poszczególne etapy prac badawczych autora mogły zawierać pomiary czasowe rzetelnie porównywalne przez Czytelników analizujących owe rezultaty.

Wyniki przedstawiające w tabelach wartości pomiarów entropii (domyślnie bezwarunkowej) i średnich bitowych poszczególnych kodowanych obrazów są podawane przy założeniu domyślnej jednostki, jaką jest liczba bitów na piksel.

Prowadząc liczne badania nad poprawkami poszczególnych metod i ich każdego etapu z osobna, autor musiał dość precyzyjne analizować nawet drobne różnice w wynikach poszczególnych kroków badawczych. Z tego powodu, aby uniknąć błędów zaokrągleń, zdecydowano się na podawanie wyników aż z pięcioma znaczącymi miejscami po przecinku. Jeśli bowiem wprowadzi się do rozbudowanej metody kilkadziesiąt usprawnień, a rezultat każdego z nich z osobna może ginąć na etapie zaokrąglania np. średniej bitowej do trzech miejsc po przecinku, to już po uwzględnieniu wszystkich udoskonaleń jednocześnie rezultaty wydają się dość znaczące.

2. Stała i statyczna predykcja liniowa

2.1. Stałe modele predykcyjne

W tym rozdziale będą rozważane podstawowe techniki wyznaczania liniowych modeli predykcyjnych, począwszy od predyktorów stałych przez otrzymane z wykorzystaniem klasycznych technik MMSE aż po metody oparte na algorytmach genetycznych.

Predyktorem stałym nazywamy taki, który ma niezmienny zestaw współczynników predykcji i może być wykorzystywany do efektywnego kodowania różnych obrazów. Podstawowym założeniem jego uniwersalności jest fakt, iż suma współczynników powinna wynosić 1 (jest to warunek nieobciążania estymatora predykcji). Kolejne założenie jest związane z poziomem korelacji między sąsiednimi pikselami, który zmniejsza się wraz z odległością euklidesową między środkami pikseli branych pod uwagę. Liczba wykorzystywanych pikseli sąsiedztwa oznacza rząd predykcji (patrz podrozdział 1.4), który w przypadku predyktora uniwersalnego (nieskojarzonego z konkretnym obrazem) nie powinien być zbyt duży (z licznych badań eksperymentalnych wyciągnięto wniosek, iż $r \leq 6$). Dlatego większość stałych predyktorów rozważanych w literaturze korzysta z zaledwie trzech, czterech pikseli z najbliższego sąsiedztwa. Zasada ta nie dotyczy np. predyktorów statycznych, w których współczynniki wyznacza się dla poszczególnych obrazów metodą minimalizacji błędu średniokwadratowego (patrz podrozdział 2.2).

Zdecydowana większość stałych predyktorów proponowanych w literaturze ma współczynniki będące pewną potęgą dwójki (lub sumą małej liczby takich potęg dwójki, np. 0,625 = 0,5 + 0,125). Ułatwia to sprzętową realizację wyznaczania wartości przewidywanej. Mnożenie można wówczas zastąpić operacjami dodawania, odejmowania i przesunięć bitowych, co zostało opisane w pracy [62], z której pochodzą predyktory Kuroki6, Kuroki7, Kuroki9 oraz Kuroki10. Wśród badanych tu wyjątek stanowią predyktory zaprezentowane w publikacji [23], których współczynniki, po zaokrągleniu do dwóch miejsc po przecinku, zostały wykorzystane także w niniejszej pracy (pod nazwą TLR85, Plane3, Daab1, Daab2, Daab5, Daab8). Podstawowy najczęściej wykorzystywany zestaw predyktorów stanowią cztery sąsiednie piksele: $P(1)$, $P(2)$, $P(3)$, $P(4)$ (patrz rys. 1.1), a także ich kombinacje liniowe JPEG5, JPEG6, JPEG7 zaproponowane w pierwszej wersji bezstratnego kodera organizacji JPEG [34, 98]. Predyktor FIR1 pochodzi z pracy [76], SK z książki [104], GradNorth, GradWest, Pirsh, Mean-4, Plane oraz Plane2 z pracy [102], a Deng10, Deng11, Deng12 oraz Deng13 z artykułu [28].

W tabeli 2.1 znajduje się zestaw współczynników dla 30 stałych modeli predykcyjnych pozyskanych ze wspomnianych wyżej prac. Każdy z nich został poddany procedurze testowej, polegającej na wyznaczeniu średniej wartości entropii rzędu zerowego błędów predykcji uzyskanych na podstawie pomiaru dla 45 obrazów testowych. Najlepszy rezultat otrzymano z użyciem predyktora Plane3, dla którego uzyskano średnią wartość entropii równą 4,65963 bitu na piksel. Duża rozbieżność wartości entropii między Plane3 a predyktorami

o najwyższych średnich (np. dla GradNorth 5,39148) nie świadczy o ich bezużyteczności, gdyż są one cennymi składnikami metody mieszania predykcyjnego opisanej w pracach [28, 102], której poświęcono cały rozdział 9. Warto też zauważyć, że nie zawsze model o niższej entropii charakteryzuje się niższym odchyleniem standardowym błędów predykcji, liczne tego przykłady można znaleźć, porównując wyniki dwóch ostatnich kolumn tab. 2.1. Z tego powodu podczas początkowych etapów badań (wyniki prezentowane w rozdziałach 2 i 3) autor posługiwał się wartością entropii bezwarunkowej jako wstępnym kryterium optymalizacji. W dalszych badaniach jako kryterium optymalizacji będzie stosowana praktyczna średnia bitowa, którą otrzymuje się, wykorzystując do pomiarów kompresję zbioru błędów predykcji za pomocą adaptacyjnego kodera arytmetycznego, którego szczegóły zostaną przedstawione w rozdziale 8.

Tab. 2.1. Zestaw współczynników dla 30 stałych predyktorów

Predyktor	b_1	b_2	b_3	b_4	b_5	b_6	b_7	b_8	b_9	b_{10}	Entropia	Odchyl. stand.
Plane3	0,66	0,66	–0,32	0	0	0	0	0	0	0	4,65963	9,16152
Pirsh	0,5	0,25	0	0,25	0	0	0	0	0	0	4,69181	9,60921
FIR1(Deng)	0,875	0,75	–0,625	0	0	0	0	0	0	0	4,69305	9,44015
JPEG6	0,5	1	–0,5	0	0	0	0	0	0	0	4,72581	9,95932
Kuroki6	0,75	1	–0,75	0	0	0	0	0	0	0	4,73202	9,87738
JPEG7	0,5	0,5	0	0	0	0	0	0	0	0	4,73214	9,87370
Kuroki7	1	0,75	–0,75	0	0	0	0	0	0	0	4,75099	9,93786
Kuroki9	0,25	0,75	0	0	0	0	0	0	0	0	4,76389	10,49113
JPEG5	1	0,5	–0,5	0	0	0	0	0	0	0	4,77718	10,15225
Daab2	0,67	0,33	0	0	0	0	0	0	0	0	4,79240	10,36259
Mean-4	0,25	0,25	0,25	0,25	0	0	0	0	0	0	4,81181	10,77055
SK	0,625	0,125	0,125	0,125	0	0	0	0	0	0	4,81304	10,65123
Plane	1	1	–1	0	0	0	0	0	0	0	4,82284	10,67700
Kuroki10	0,75	0,25	0	0	0	0	0	0	0	0	4,82706	10,80077
TLR85	1	0	–0,25	0,25	0	0	0	0	0	0	4,84745	11,05663
Daab5	0,6	0,2	0,2	0	0	0	0	0	0	0	4,88828	11,32796
Daab8	0,2	0,2	0,2	0,2	0	0	0	0	0	0,2	4,90450	11,39137
P2	0	1	0	0	0	0	0	0	0	0	4,90547	12,45490
P1	1	0	0	0	0	0	0	0	0	0	4,99074	12,97382
Daab1	0,67	0	0,33	0	0	0	0	0	0	0	5,01920	12,82066
Deng10	0	0,5	0	0,5	0	0	0	0	0	0	5,02628	12,97303
Deng13	1	1	–0,5	0	0	0	–0,5	0	0	0	5,02951	11,82520
Deng12	1	–0,5	–0,5	1	0	0	0	0	0	0	5,25460	14,72167
Far	0	0	0	0	0,25	0,25	0,125	0,125	0,125	0,125	5,26055	14,58898
GradWest	0	2	0	0	0	–1	0	0	0	0	5,30247	15,32461
P3	0	0	1	0	0	0	0	0	0	0	5,31389	16,21074
Plane2	1	–1	0	1	0	0	0	0	0	0	5,31402	16,00248
P4	0	0	0	1	0	0	0	0	0	0	5,32856	16,24633
GradNorth	2	0	0	0	–1	0	0	0	0	0	5,39148	16,05819
Deng11	1	–1	1	0	0	0	0	0	0	0	5,74984	21,97527

Większość analiz stałych predyktorów pochodzi z lat 90. XX wieku. W miarę postępu technologicznego algorytmy o coraz to większej złożoności implementacyjnej można uzna-

wać za akceptowalne nawet do zastosowań czasu rzeczywistego. Z tego też względu często wykorzystuje się liniowe predyktory statyczne ze współczynnikami dostosowanymi do kodowanego obrazu będące rozwinięciem predyktorów stałych. W kolejnych podrozdziałach tego rozdziału zostaną omówione metody wyznaczania współczynników predykcji statycznej oparte na różnych kryteriach minimalizacji.

2.2. Wyznaczanie współczynników predykcji metodą MMSE

Powszechnie przyjęło się, że dla poszczególnych obrazów wyznaczenie współczynników predykcji metodą minimalizacji błędu średniokwadratowego MMSE (ang. *Minimum Mean Square Error*) daje bardzo dobre rezultaty. Jednak kryterium to nie jest jednoznaczne z otrzymaniem modelu pozwalającego czy to uzyskać możliwie najmniejszą wartość entropii rzędu zerowego, czy najmniejszą średnią bitową otrzymaną po uwzględnieniu danych nagłówkowych i kodowania arytmetycznego, co zostanie pokazane w kolejnych podrozdziałach tego rozdziału. Dzięki swej prostocie metoda ta pozwala stosunkowo szybko obliczyć współczynniki predykcji modeli wysokich rzędów ($r > 10$). Wyznaczenie predyktora statycznego (o stałych współczynnikach dobranych indywidualnie do cech konkretnego obrazu) metodą MMSE wymaga znajomości całego kodowanego obrazu przed rozpoczęciem kodowania. Poniżej znajduje się praktyczny sposób obliczania współczynników metodą MMSE. Wektor **B** współczynników predykcji jest wyznaczany z równania macierzowego:

$$\mathbf{B} = \mathbf{R}^{-1} \cdot \mathbf{P}, \tag{2.1}$$

gdzie **R** jest macierzą kwadratową o wymiarach $r \times r$ elementów $\mathbf{R}(j, i)$ takich, że:

$$\mathbf{R}(j,i) = \sum_{y=2}^{\text{wys}} \sum_{z=2}^{\text{szer}} P_{(y,z)}(i) \cdot P_{(y,z)}(j), \tag{2.2}$$

P jest wektorem o wymiarach $r \times 1$ elementów $\mathbf{P}(j)$ takich, że:

$$\mathbf{P}(j) = \sum_{y=2}^{\text{wys}} \sum_{z=2}^{\text{szer}} P_{(y,z)}(0) \cdot P_{(y,z)}(j), \tag{2.3}$$

a indeks (y, z) wskazuje na aktualne umiejscowienie piksela $P(0)$ w przestrzeni obrazu (y – numer wiersza, z – numer kolumny). Jak widać, jest to sumowanie po całym obrazie (z wyjątkiem pierwszego wiersza i pierwszej kolumny, co zaobserwowano eksperymentalnie, otrzymując wyższą efektywność kodowania, i uzasadniono w podrozdziale 1.4), przy czym zmienna wys to liczba wierszy, a zmienna szer to liczba kolumn obrazu. Indeksy i oraz j pojawiające się we wzorach dotyczą rzutowania pikseli sąsiedztwa na dziedzinę jednowymiarową względem aktualnego piksela $x = P(0)$ (patrz rys. 1.1). Wyprowadzenie wzorów od (2.1) do (2.3) znajduje się w pracach [34, 98].

Najczęściej zakłada się (na podstawie badań eksperymentalnych), że dokładność zapisu każdego współczynnika wynosi od ośmiu do dziesięciu bitów części ułamkowej, a do tego jeden bit znaku i jeden bit wartości całkowitej przy praktycznym założeniu, że współczynnik predykcji mieści się w przedziale od –1,999 do 1,999. Jeśli weźmiemy pod uwagę dziewięć bitów części ułamkowej, wówczas łączny koszt informacji o współczynnikach, które należy umieścić w nagłówku pliku, to $11 \cdot r$ bitów. Redukcję do liczby z dziewięcioma najbardziej znaczącymi bitami reprezentującymi część ułamkową uzyskujemy dzięki operacji zaokrąglania rzeczywistych wartości wektora **B**:

$$\overline{\mathbf{B}} = \lfloor \mathbf{B} \cdot 2^{9} + 0{,}5 \rfloor \cdot 2^{-9}. \tag{2.4}$$

Dzięki temu zabiegowi wielkość nagłówka pliku nie jest zbyt duża nawet dla większych rzędów predykcji. Choć ze wzrostem rzędu zmniejsza się błąd średniokwadratowy, to nie mamy gwarancji poprawy efektywności kompresji. Należy zatem dążyć do znalezienia kompromisowej wartości r lub dokonywać pomiarów entropii dla kolejnych rzędów predykcji.

Wadą zastosowania MMSE jest potrzeba wykonywania obliczeń z wykorzystaniem liczb o dużej dokładności ich reprezentacji, co sprowadza się do użycia liczb zmiennopozycyjnych, a to z kolei jest dużą przeszkodą przy projektowaniu wydajnych i jednocześnie energooszczędnych implementacji sprzętowych. W takich rozwiązaniach zawsze trzeba podejmować kompromisowe decyzje.

W kolejnych podrozdziałach zostaną omówione konkurencyjne metody doboru współczynników predykcji dostosowane do cech danego obrazu.

2.3. Dobór predyktora statycznego

2.3.1. Metoda prostej selekcji oparta na predyktorach stałych

Pierwszym pomysłem wyznaczenia efektywnego modelu predykcji zaproponowanym w tej pracy jest wybór dla danego obrazu najlepszego spośród 30 predyktorów stałych znajdujących się w tab. 2.1, który pozwala na uzyskanie średniej wartości entropii często mniejszej niż otrzymana z wykorzystaniem wyznaczenia współczynników predykcji metodą MMSE z doborem najlepszego rzędu $r \le 4$ (patrz lewa część tab. 2.3). Wymaga to podjęcia decyzji o doborze predyktora dopiero po 30-krotnym zakodowaniu obrazu. Jednak operację taką można w łatwy sposób zrównoleglić (zwłaszcza w przypadku rozwiązań sprzętowych), gdyż każdy koder może działać całkowicie niezależnie. Dopiero po analizie entropii każdego z 30 zakodowanych zestawów błędów predykcji podejmuje się decyzję o przesłaniu na wyjście tego bufora wynikowego, w którym znajduje się najefektywniej zakodowany obraz. Nagłówek zawiera jedynie numer jednego z 30 predyktorów, który został użyty do kodowania obrazu. Koszt nagłówka to zaledwie pięć bitów.

Wyniki działania takiego algorytmu przedstawia trzecia kolumna tab. 2.2. W tej metodzie uzyskano średnią 4,54569, co oznacza wyraźną poprawę w stosunku do zastosowania dla każdego kodowanego obrazu predyktora Plane3.

Tab. 2.2. Pomiar entropii dla najlepszego z 30 oraz z 31 predyktorów

Obrazy	Predyktor	Entropia (30 predyktorów)	Entropia (31 predyktorów)
Aerial	Plane3	5,31407	5,29816
Airfield	Plane3	5,21899	5,19447
Airplane	Plane3	4,23576	4,22887
Baboon	Daab2	6,20949	6,20065
Barbara	Kuroki9	5,49449	5,48057
Boat	Plane3	5,14432	5,12724
Bridge	Plane3	3,78952	3,77570
Couple	FIR1	4,56562	4,55002
Crowd	FIR1	4,38343	4,38319
Elaine	Mean-4	4,96070	4,93586
Finger	FIR1	5,55965	5,54495
Frog	P(3)	4,76723	4,76723
Goldhill	Plane3	4,90578	4,90045
Harbour	P(1)	5,11231	5,10351
Lax	Kuroki9	5,90551	5,88633
Lennagrey	Pirsh	4,50269	4,46329
$Lena_{TMW}$	Pirsh	4,81338	4,79786
Man	Plane3	4,81127	4,79933
Peppers	Pirsh	4,70288	4,69326
Sailboat	Pirsh	5,05211	5,05211
Seismic	JPEG5	2,81418	2,80187
Shapes	Plane	1,35035	1,35035
Tank	Pirsh	4,04491	4,03404
Truck	Kuroki10	4,36650	4,35726
Woman1	Pirsh	4,39386	4,36638
Woman2	Plane3	3,53922	3,52173
Balloon	JPEG6	3,17814	3,12768
Barb	JPEG6	5,29917	5,25347
Barb2	P(2)	5,19746	5,16264
Board	Plane3	4,14415	4,14415
Boats	FIR1	4,42455	4,40189
Girl	Plane3	4,24452	4,20951
Gold	Plane3	4,75525	4,74809
Hotel	Plane3	4,85943	4,85425
Zelda	Kuroki9	4,07126	4,04126
Bridge256	Plane3	5,93175	5,91336
Camera	Plane3	4,81850	4,81700
Couple256	Kuroki6	4,12660	4,09306
Earth	Plane3	3,73941	3,73941
Elif	FIR1	3,34454	3,34146
Noisesquare	Daab8	5,37101	5,31577
Omaha	P(2)	5,96855	5,96855
Sena	TLR85	3,61563	3,56751
Sensin	FIR1	3,86959	3,84714
Sinan	Plane	3,63841	3,56938
Średnia	–	4,54569	4,52734

Rozwinięciem tego pomysłu może być podział obrazu na bloki (np. kwadraty o boku ośmiu pikseli) i dobór najlepszego predyktora osobno dla każdego z bloków. Utrudnia to analizę wartości entropii i wymaga przesyłania dodatkowych informacji o numerze predyktora skojarzonego z każdym kwadratem, dlatego ten pomysł nie będzie dalej analizowany w tej pracy. Metodę tę zaprezentowano w pracy [81], a jej rozwinięcia z doborem predyktorów statycznych zaproponowano w publikacjach [4, 38, 78, 86, 121] (patrz podrozdział 2.6).

2.3.2. Metoda uśredniania współczynników

Poszukując modelu dającego najmniejszą wartość entropii lub średniej bitowej zgodnie z zasadą opisaną w podrozdziale 2.3.1, można dodatkowo wyznaczyć kolejny model predykcyjny (31. predyktor) będący kombinacją liniową najlepszych dla danego obrazu czterech (posortowanych według rosnących wartości entropii i oznaczonych poniżej jako modele $\mathbf{B}_1$, $\mathbf{B}_2$, $\mathbf{B}_3$, $\mathbf{B}_4$) spośród 30 stałych predyktorów:

$$\mathbf{B} = 0{,}5 \cdot \mathbf{B}_1 + 0{,}25 \cdot \mathbf{B}_2 + 0{,}125 \cdot \mathbf{B}_3 + 0{,}125 \cdot \mathbf{B}_4 . \tag{2.5}$$

Wówczas wybieramy najlepszy spośród 31 zestawów współczynników predykcji. Pozwala to uzyskać średnią 4,52734 – dzięki temu pomysłowi jedynie w pięciu przypadkach (na 45 badanych) nie udało się poprawić efektywności kompresji. Wyniki działania takiego autorskiego algorytmu przedstawia ostatnia kolumna tab. 2.2.

2.4. Metoda poszukiwania współczynników

2.4.1. Pierwsza propozycja

W literaturze często przyjmuje się minimalizację błędu średniokwadratowego jako metodę wyznaczania najlepszego zestawu współczynników predykcji. Bardzo łatwo można udowodnić, że istnieją możliwości uzyskiwania współczynników predykcji pozwalających otrzymać niższe wartości entropii rzędu zerowego niż z użyciem metody MMSE (patrz tab. 2.3), co wykorzystano choćby w pracy [78], gdzie fakt niestacjonarności sygnału, jakim są obrazy cyfrowe, pozwolił na zaprojektowanie wielu lokalnych predyktorów liniowych (patrz podrozdział 2.6) optymalizowanych w sposób odmienny od lokalnego użycia kryterium MMSE. Poniżej przedstawiono trzy w pełni autorskie propozycje poszukiwania współczynników liniowego predyktora statycznego.

Jedną z podstawowych możliwości doboru liniowego modelu predykcyjnego jest metoda pełnego przeszukiwania przy założonej dokładności wartości współczynników predykcji, co można potraktować jako uogólnienie metody zaprezentowanej w podrozdziale 2.3.1. Przeszukiwanie wiąże się z wykonaniem dużej liczby pomiarów wartości entropii. Dla uproszczenia możemy uznać, iż suma współczynników predykcji wynosi 1 oraz że każdy ze współczynników b_j należy do przedziału od –1 do 1. Przy tych założeniach, ustalając

wartości współczynników predykcji z dokładnością 0,125 (trzy bity części ułamkowej), do wyznaczenia najlepszego zestawu współczynników predykcji czwartego rzędu należy wykonać 2445 pomiarów. Tabela 2.3 zawiera porównanie metod MMSE i pełnego przeszukiwania przy doborze najlepszego zestawu współczynników predykcji dla rzędu $r \le 4$. Jak widać, średnia wartość entropii metody poszukiwań wyniosła 4,51105 i okazała się niższa o 0,04648 bitu w porównaniu z metodą MMSE. Jednocześnie należy podkreślić, iż wartości odchylenia standardowego błędów predykcji są wyższe średnio o 3% w stosunku do klasycznej MMSE.

Zwiększanie rzędu predykcji wiąże się ze znacznym wzrostem liczby pomiarów ze względu na wykładniczy charakter algorytmu przeszukiwania. Dlatego w następnym badaniu, polegającym na doborze predyktora rzędu $r = 6$, posłużono się kolejnymi uproszczeniami. Na podstawie wstępnych badań zauważono, że dla testowych obrazów współczynniki b_1, b_2 oraz b_4 w metodzie pełnego przeszukiwania nie są ujemne, ponadto w zdecydowanej większości przypadków współczynniki b_5 oraz b_6 są niewielkimi niedodatnimi wartościami, zatem do ich zapisu można użyć tylko jednej z trzech wartości {–0,25; –0,125; 0}. Współczynnik b_3 jest wyznaczany z poniższej zależności:

$$b_3 = 1 - b_1 - b_2 - b_4 - b_5 - b_6 . \tag{2.6}$$

Przy takich założeniach należy wykonać 5961 testów. Czas wyznaczenia współczynników predykcji tą metodą dla obrazu o rozdzielczości 512 × 512 pikseli wynosi aż 308 s (przy wykorzystaniu procesora Pentium4 2,8 GHz).

Współczynniki modeli predykcyjnych wyznaczone w ten sposób przedstawiono w ostatniej kolumnie tab. 2.3 ($r = 4$) oraz ostatniej kolumnie tab. 2.4 ($r = 6$). Tabela 2.4 zawiera porównanie metod MMSE i pełnego przeszukiwania przy doborze najlepszego zestawu współczynników predykcji dla rzędu $r \le 6$. Choć uzyskana średnia wartość entropii przy $r = 6$ wynosi 4,47065 i jest niższa o 0,03349 bitu w porównaniu z metodą MMSE, to jednak czas poświęcony na wyznaczenie współczynników jest zbyt duży, aby uznać tę metodę za praktyczną przy większym rzędzie predykcji.

Tab. 2.3. Pomiar entropii dla predyktorów statycznych i stałych czwartego rzędu

Obrazy	Entropia MMSE ($r \leq 4$)	Odchyl. stand.	r	Entropia ($r = 4$)	Odchyl. stand.	Współczynniki przy $r = 4$ (metoda poszukiwań, 2445 pomiarów)
Aerial	5,29204	13,10456	4	5,29620	13,19239	0,750 0,625 –0,375 0,000
Airfield	5,19960	10,34857	4	5,19082	10,51920	0,500 0,500 –0,125 0,125
Airplane	4,25611	6,48218	4	4,21087	6,95954	0,750 0,500 –0,375 0,125
Baboon	6,18243	19,23614	4	6,18487	19,31095	0,750 0,250 –0,125 0,125
Barbara	5,45198	14,21329	4	5,43122	14,83355	0,375 0,625 –0,250 0,250
Boat	5,09598	9,17541	4	5,09192	9,29620	0,500 0,625 –0,250 0,125
Bridge	3,75542	3,68094	4	3,75886	3,69353	0,625 0,500 –0,250 0,125
Couple	4,53481	6,58782	4	4,54521	6,67680	0,750 0,875 –0,625 0,000
Crowd	4,34340	7,15953	4	4,34322	7,24685	0,750 0,625 –0,500 0,125
Elaine	4,90094	7,59982	4	4,89796	7,65151	0,375 0,000 0,250 0,375
Finger	5,53578	11,25926	4	5,54488	11,33010	0,750 0,750 –0,500 0,000
Frog	5,15859	11,47120	1	4,76723	12,09222	0,000 0,000 1,000 0,000
Goldhill	4,89005	7,89367	4	4,89282	7,92306	0,625 0,625 –0,375 0,125
Harbour	5,12570	17,00562	1	5,09819	15,10846	0,875 0,250 –0,125 0,000
Lax	5,88337	16,87310	4	5,87999	17,01661	0,375 0,500 0,000 0,125
Lennagrey	4,43116	6,42586	4	4,41801	6,47028	0,500 0,500 –0,250 0,250
Lena_{TMW}	4,76276	7,80133	4	4,75333	7,95782	0,375 0,500 –0,125 0,250
Man	4,78491	8,86440	4	4,78536	8,88001	0,625 0,625 –0,375 0,125
Peppers	4,76670	7,72490	4	4,66587	8,51892	0,375 0,125 0,125 0,375
Sailboat	5,08592	9,08709	4	5,05062	9,29874	0,625 0,250 –0,125 0,250
Seismic	2,79764	1,72550	4	2,80773	1,73685	1,000 0,750 –0,625 –0,125
Shapes	2,17568	6,72140	3	1,35035	7,96968	1,000 1,000 –1,000 0,000
Tank	4,02417	4,56981	4	4,01941	4,63083	0,625 0,250 –0,125 0,250
Truck	4,32081	5,59537	4	4,31954	5,65026	0,750 0,250 –0,125 0,125
Woman1	4,35277	6,49094	4	4,34555	6,56106	0,375 0,500 –0,125 0,250
Woman2	3,48241	3,12461	4	3,48347	3,15083	0,750 0,375 –0,375 0,250
Balloon	3,18915	2,55902	4	3,10107	3,12941	0,500 0,750 –0,375 0,125
Barb	5,21432	12,15212	4	5,21137	12,22133	0,375 0,750 –0,250 0,125
Barb2	5,12679	10,75041	4	5,11515	11,00215	0,250 0,875 –0,250 0,125
Board	4,13846	4,98996	4	4,11543	5,28593	0,750 0,625 –0,500 0,125
Boats	4,39228	5,89569	4	4,38880	5,97256	0,750 0,750 –0,625 0,125
Girl	4,19160	4,74746	4	4,17513	4,90695	0,625 0,750 –0,500 0,125
Gold	4,73696	7,19556	4	4,73912	7,27281	0,750 0,500 –0,375 0,125
Hotel	4,85361	8,13463	4	4,84838	8,53827	0,625 0,625 –0,375 0,125
Zelda	4,07572	4,35824	4	4,00280	4,72943	0,375 0,500 –0,125 0,250
Bridge256	5,89670	15,68813	4	5,89550	15,77272	0,750 0,375 –0,250 0,125
Camera	4,84027	14,12334	4	4,82224	14,56351	0,625 0,500 –0,250 0,125
Couple256	4,08989	6,07431	4	4,09926	6,12984	0,750 0,875 –0,625 0,000
Earth	3,74006	8,61043	3	3,74237	8,62073	0,625 0,625 –0,250 0,000
Elif	3,33790	3,16835	4	3,29884	3,24571	0,875 0,375 –0,500 0,250
Noisesquare	5,39697	10,44103	4	5,40056	10,46342	0,250 0,250 0,250 0,250
Omaha	6,36096	27,31859	2	5,96855	32,45343	0,000 1,000 0,000 0,000
Sena	3,55786	3,71202	4	3,55652	3,73176	0,875 0,250 –0,375 0,250
Sensin	3,81354	4,01973	4	3,82307	4,03854	0,875 0,625 –0,625 0,125
Sinan	3,54480	3,19847	4	3,55939	3,25758	0,875 0,625 –0,625 0,125
Średnia	4,55753	8,60800	–	4,51105	8,86694	–

Tab. 2.4. Pomiar entropii dla predyktorów statycznych i stałych szóstego rzędu

Obrazy	Entropia MMSE ($r \leq 6$)	r	Entropia ($r = 6$)	Współczynniki przy $r = 6$ (metoda poszukiwań, 5961 pomiarów)
Aerial	5,22609	6	5,22831	0,875 0,500 –0,250 0,125 –0,125 –0,125
Airfield	5,18798	6	5,17930	0,625 0,500 –0,125 0,125 –0,125 0,000
Airplane	4,21614	6	4,16863	0,750 0,625 –0,250 0,125 –0,125 –0,125
Baboon	6,16692	6	6,17336	0,750 0,250 0,000 0,125 –0,125 0,000
Barbara	5,23613	6	5,29455	0,250 0,750 0,000 0,250 0,000 –0,250
Boat	5,04637	6	5,04130	0,375 0,875 –0,125 0,125 0,000 –0,250
Bridge	3,74923	6	3,75886	0,625 0,500 –0,250 0,125 0,000 0,000
Couple	4,53397	5	4,54521	0,750 0,875 –0,625 0,000 0,000 0,000
Crowd	4,25940	6	4,26328	0,875 0,500 –0,375 0,250 –0,125 –0,125
Elaine	4,90094	4	4,89796	0,375 0,000 0,250 0,375 0,000 0,000
Finger	5,37755	6	5,40000	0,750 0,875 –0,375 0,125 –0,125 –0,250
Frog	5,15859	1	4,76723	0,000 0,000 1,000 0,000 0,000 0,000
Goldhill	4,89005	4	4,89282	0,625 0,625 –0,375 0,125 0,000 0,000
Harbour	5,12570	1	5,09819	0,875 0,250 –0,125 0,000 0,000 0,000
Lax	5,87787	6	5,87999	0,375 0,500 0,000 0,125 0,000 0,000
Lennagrey	4,39570	6	4,39177	0,500 0,625 0,000 0,250 –0,125 –0,250
Lena_{TMW}	4,74117	6	4,73156	0,500 0,500 0,000 0,250 –0,125 –0,125
Man	4,74824	6	4,76189	0,500 0,750 –0,250 0,125 0,000 –0,125
Peppers	4,76119	6	4,66359	0,375 0,125 0,125 0,500 0,000 –0,125
Sailboat	5,07406	6	5,04149	0,500 0,250 0,000 0,375 0,000 –0,125
Seismic	2,73033	6	2,80885	1,000 0,625 –0,625 0,000 0,000 0,000
Shapes	2,17568	3	1,35035	1,000 1,000 –1,000 0,000 0,000 0,000
Tank	4,02365	6	4,01941	0,625 0,250 –0,125 0,250 0,000 0,000
Truck	4,30225	6	4,30433	0,875 0,250 –0,125 0,125 –0,125 0,000
Woman1	4,34794	6	4,33859	0,375 0,625 –0,125 0,250 0,000 –0,125
Woman2	3,38467	6	3,39052	0,750 0,500 –0,125 0,250 –0,125 –0,250
Balloon	3,15826	6	3,06974	0,625 0,750 –0,250 0,125 –0,125 –0,125
Barb	4,97175	6	5,06298	0,375 1,000 –0,250 0,125 0,000 –0,250
Barb2	5,05425	6	5,04277	0,250 1,000 –0,125 0,125 0,000 –0,250
Board	4,12430	6	4,09330	0,625 0,750 –0,375 0,125 0,000 –0,125
Boats	4,36880	6	4,35778	0,625 0,875 –0,500 0,125 0,000 –0,125
Girl	4,12476	6	4,07593	0,500 0,875 –0,250 0,125 0,000 –0,250
Gold	4,73247	6	4,73432	0,625 0,625 –0,250 0,125 0,000 –0,125
Hotel	4,83058	6	4,81517	0,625 0,750 –0,375 0,125 0,000 –0,125
Zelda	4,03792	6	3,95156	0,250 0,750 0,000 0,250 0,000 –0,250
Bridge256	5,89389	6	5,89550	0,750 0,375 –0,250 0,125 0,000 0,000
Camera	4,83992	5	4,82224	0,625 0,500 –0,250 0,125 0,000 0,000
Couple256	4,08989	4	4,09926	0,750 0,875 –0,625 0,000 0,000 0,000
Earth	3,72441	6	3,72858	0,625 0,625 –0,125 0,125 –0,125 –0,125
Elif	3,12626	6	3,17837	1,000 0,250 –0,250 0,250 –0,250 0,000
Noisesquare	5,33669	6	5,40056	0,250 0,250 0,250 0,250 0,000 0,000
Omaha	6,36096	2	5,96855	0,000 1,000 0,000 0,000 0,000 0,000
Sena	3,37874	6	3,45746	1,000 0,125 –0,125 0,250 –0,250 0,000
Sensin	3,57400	6	3,65004	1,000 0,125 –0,125 0,375 –0,250 –0,125
Sinan	3,32051	6	3,38393	1,000 0,375 –0,250 0,250 –0,250 –0,125
Średnia	4,50414	–	4,47065	–

2.4.2. Druga propozycja

Zasadę pełnego przeszukiwania omówioną w poprzednim podrozdziale przy założonej dokładności wartości współczynników predykcji można uprościć, zastępując iteracyjnym przeszukiwaniem wybiórczym. Wystarczy zainicjalizować pierwotny wektor współczynników $\mathbf{B} = [1, 0, \ldots, 0]$, a następnie wygenerować zestaw wektorów modyfikujących $\Delta\mathbf{B} = [\Delta b_1, \Delta b_2, \ldots, \Delta b_r]$, których elementy Δb_j składają się wyłącznie z liczb całkowitych ze zbioru $\{-1, 0, 1\}$ przy założeniu, że ich suma wynosi zero. Na przykład przy $r = 4$ istnieje 19 takich wektorów, które przedstawiono w tab. 2.5. W wektorze modyfikującym suma elementów Δb_j jest równa zeru, gdy istnieje tyle samo liczb 1 i –1 przy dowolnej liczbie zer. Oznacza to, że dla dowolnego r uzyskujemy następującą liczbę N_B wektorów modyfikujących:

$$N_B = \sum_{k=0}^{\left\lfloor \frac{r}{2} \right\rfloor} \binom{r}{k} \cdot \binom{r-k}{k} = \sum_{k=0}^{\left\lfloor \frac{r}{2} \right\rfloor} \frac{r!}{k! \cdot k! \cdot (r-2k)!}, \tag{2.7}$$

gdzie k określa liczbę par liczb $\{-1, 1\}$, a $(r - 2k)$ jest liczbą zer w wektorze $\Delta\mathbf{B}$ o r elementach.

Tab. 2.5. 19 różnych czteroelementowych wektorów o sumie równej zero

Δb_1	–1	–1	–1	–1	–1	–1	0	0	0	0	0	0	0	1	1	1	1	1	1
Δb_2	–1	0	0	1	1	1	–1	–1	0	0	0	1	1	–1	–1	–1	0	0	1
Δb_3	1	0	1	–1	0	1	0	1	–1	0	1	–1	0	–1	0	1	–1	0	–1
Δb_4	1	1	0	1	0	–1	1	0	1	0	–1	0	–1	1	0	–1	0	–1	–1

Następnie ustalamy dokładność poszukiwania każdego ze współczynników predykcji na trzy bity po przecinku (podobnie jak w podrozdziale 2.4.1). Przystępujemy do poszukiwań, dodając do wektora **B** każdy kolejny wektor $\Delta\mathbf{B}$ i sprawdzając wartość entropii. Jeśli entropia się zmniejsza, to taki wektor $\mathbf{B} + \Delta\mathbf{B}$ staje się nowym domyślnym wektorem **B**. Po przejściu przez cały zestaw wektorów modyfikujących sytuację ponawiamy. W praktyce wystarczą dwie takie serie pomiarów (iteracje), po czym rozpoczynamy następne dwie iteracje, dodając kolejne wektory modyfikujące pomnożone przez ujemną potęgę dwójki, czyli ogólnie dokonujemy dodawań:

$$\mathbf{B}(t+1) = \mathbf{B}(t) + 2^{-i} \cdot \Delta\mathbf{B}, \tag{2.8}$$

gdzie po każdych dwóch iteracjach zwiększamy i, przy czym $i = \{0, 1, 2, 3\}$. Ten sposób nie wprowadza takich ograniczeń jak poprzednia propozycja i w związku z tym współczynniki b_j docelowo mogą przyjmować też wartości większe od 1 i mniejsze od –1. Mimo wybiórczej techniki poszukiwań złożoność obliczeniowa, choć mniejsza, to i tak jest wykładnicza (liczba wektorów modyfikujących została eksperymentalnie oszacowana na około $2{,}7^{(r-1)}$). Już przy

$r = 8$ istnieje $N_B = 1107$ wektorów modyfikujących, co oznacza $2 \cdot 4 \cdot 1107 = 8856$ pomiarów. Czas wyznaczenia współczynników predykcji tą metodą przy $r = 8$ dla obrazu o rozdzielczości 512×512 pikseli wynosi aż 515 s. Przy tym rzędzie uzyskana średnia entropia dla zestawu 45 obrazów testowych wynosi 4,43446 (patrz druga kolumna tab. 2.6).

Metodę tę można uprościć do jeszcze bardziej wybiórczej przez dalsze redukcje schematu budowy wektorów modyfikujących $\Delta\mathbf{B}$ (patrz podrozdział 2.4.3) lub zaprojektować odpowiedni algorytm genetyczny (patrz podrozdział 2.5), który w wielu przypadkach jest skuteczny w znajdowaniu przybliżonych rozwiązań problemów o złożoności wykładniczej.

2.4.3. Trzecia propozycja

Kolejna propozycja opiera się na propozycji z podrozdziału 2.4.2, pozwalającej na uzyskanie zadziwiająco dobrych rezultatów przy dość naiwnym założeniu, że wybiórcze poszukiwanie jest iteracyjne w tym sensie, że na początku daje tylko zgrubne dopasowanie, a potem jest doprecyzowane coraz mniej znaczącymi bitami po przecinku każdego ze współczynników predykcji. Omawiana tu propozycja wykorzystuje to samo założenie. Także proces inicjalizacji jest identyczny, czyli na początku ustalamy pierwotny wektor współczynników jako $\mathbf{B} = [1, 0, \ldots, 0]$.

W propozycji z podrozdziału 2.4.2 generowano zestaw wszystkich wektorów modyfikujących $\Delta\mathbf{B} = [\Delta b_1, \Delta b_2, \ldots, \Delta b_r]$, których elementy Δb_j składały się wyłącznie z liczb całkowitych ze zbioru $\{-1, 0, 1\}$ przy założeniu, że ich suma wynosi zero. Uproszczeniem tego pomysłu jest generowanie zestawu wektorów modyfikujących $\Delta\mathbf{B} = [\Delta b_1, \Delta b_2, \ldots, \Delta b_r]$ w taki sposób, aby w wektorze pojawiła się (jako elementy Δb_j) para dwóch liczb niezerowych $\{-1, 1\}$, a pozostałe elementy Δb_j miały wartość zero. Takie rozwiązanie sprawia, że dla dowolnego r uzyskujemy wielomianową, a nie wykładniczą liczbę N_{B+} wektorów modyfikujących $\Delta\mathbf{B}$. Jest ich zaledwie $N_{B+} = r \cdot (r - 1)$.

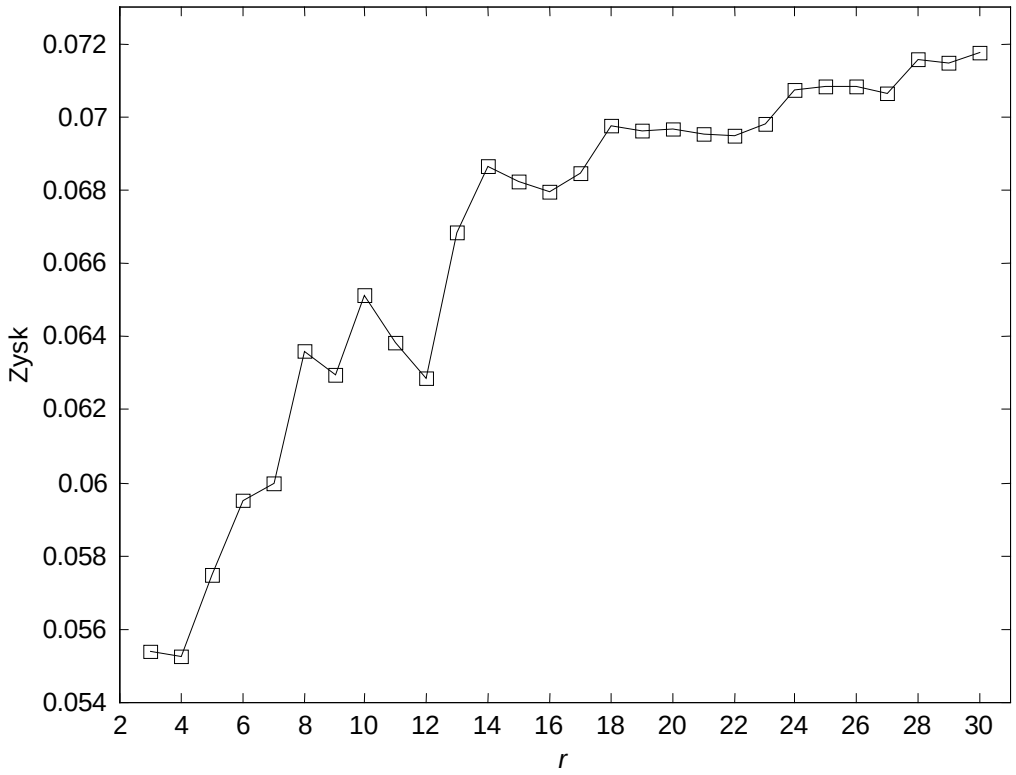
Dzięki temu możemy zwiększyć dokładność poszukiwania każdego ze współczynników predykcji do siedmiu bitów po przecinku (w podrozdziałach 2.4.1 oraz 2.4.2 były to tylko trzy bity po przecinku). Przystępujemy do poszukiwań, dodając do wektora $\mathbf{B}$ każdy kolejny wektor $\Delta\mathbf{B}$ i sprawdzając wartość entropii. Jeśli entropia się zmniejsza, to taki wektor $\mathbf{B} + \Delta\mathbf{B}$ staje się nowym domyślnym modelem predykcyjnym $\mathbf{B}$. Po przejściu przez cały zestaw wektorów modyfikujących sytuację ponawiamy, jeśli znaleziono choć jeden lepszy wektor $\mathbf{B}$. W praktyce wystarczy jedna taka seria pomiarów (iteracja) dla każdego z czterech najstarszych bitów, a dla czterech najmłodszych mogą wystąpić maksymalnie po cztery iteracje (w propozycji z podrozdziału 2.4.2 były to tylko dwie iteracje).

Po zakończeniu tego etapu sytuację powtarzamy, dodając kolejne wektory modyfikujące pomnożone przez ujemną potęgę dwójki, czyli ogólnie dokonujemy dodawań zgodnie ze wzorem (2.8), przy czym $i = \overline{0;7}$.

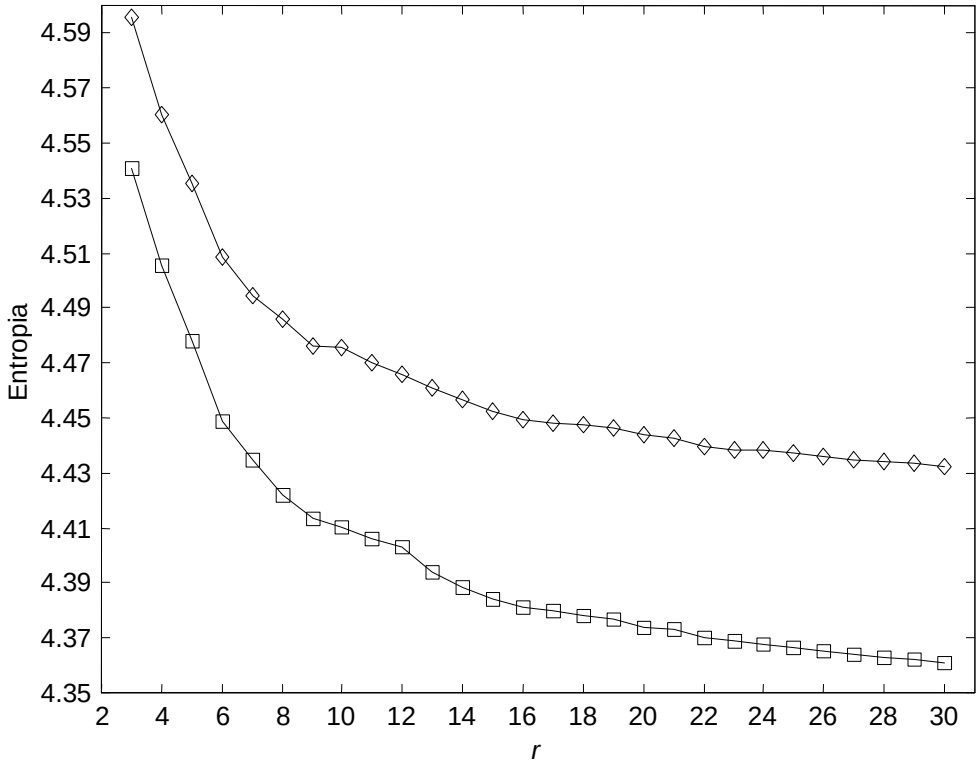
Dzięki zwiększeniu dokładności zapisu współczynników predykcji oraz liczby iteracji poszukiwania dla czterech najmniej znaczących bitów tych współczynników udało się (mimo

istnienia zaledwie 56 wektorów modyfikujących $\Delta\mathbf{B}$ przy $r = 8$) uzyskać zauważalny spadek średniej bitowej (dla zestawu 45 obrazów testowych) z 4,43446 w przypadku drugiej propozycji do 4,42221 przy wykorzystaniu trzeciej propozycji (patrz trzecia kolumna tab. 2.6).

Drugą istotną zaletą omawianego tu rozwiązania (trzecia propozycja) jest możliwość jego zastosowania w skończonym czasie dla znacznie wyższych rzędów predykcji, co nie było możliwe w przypadku dwóch wcześniejszych propozycji. Bardzo istotnym i zaskakującym wnioskiem jest fakt, że wraz ze wzrostem rzędu predykcji rośnie zysk, czyli różnica między wartością entropii uzyskaną dzięki wyznaczeniu współczynników predykcji klasyczną metodą MMSE a entropią uzyskaną z użyciem modelu predykcyjnego otrzymanego metodą udoskonalonego poszukiwania wybiórczego (trzecia propozycja). Zależność tego zysku od rzędu predykcji przedstawiono na rys. 2.1, a na rys. 2.2 porównano obie wartości entropii w funkcji rzędu predykcji z przedziału $r = \overline{3;30}$, gdzie wartość entropii dla współczynników uzyskanych metodą MMSE przedstawiono w postaci punktów pomiarowych oznaczonych rombem, a dla współczynników uzyskanych metodą z podrozdziału 2.4.3 jako punkty pomiarowe oznaczone kwadratem.



Rys. 2.1. Zysk wartości entropii po zastosowaniu metody z podrozdziału 2.4.3 w stosunku do klasycznej MMSE w zależności od rzędu predykcji



Rys. 2.2. Zależność średniej entropii od rzędu predykcji (współczynniki z metody MMSE – punkty pomiarowe oznaczone rombem oraz współczynniki z metody z podrozdziału 2.4.3 – punkty pomiarowe oznaczone kwadratem)

2.5. Wyznaczanie współczynników predykcji z wykorzystaniem algorytmów genetycznych

2.5.1. Wiadomości ogólne

Rezultaty uzyskane w podrozdziale 2.4.2 pokazują, jak istotną i nierozwiązaną kwestią jest metoda poszukiwania współczynników predykcji. W przypadku nawet wybiórczego przeszukiwania uzyskano wykładniczą złożoność obliczeniową sugerowanego algorytmu. Dalsze uproszczenie poszukiwania współczynników pozwoliło w podrozdziale 2.4.3 zredukować złożoność obliczeniową do wielomianowej. Idąc tym tropem, dochodzimy do algorytmów genetycznych, które również w wybiórczy sposób (choć z elementami losowości, a nie przeglądania systematycznego) mogą dokonywać poszukiwań najlepszego zestawu współczynników predykcji. W znacznie uproszczonej wersji próbowano z tego pomysłu korzystać w pracy [78], przedstawiony tam bowiem pomysł był oparty na losowości i małej liczbie pomiarów, lecz współczynniki predykcji modyfikowano w sposób opisany w podrozdziale 2.4.3.

Próby znalezienia podobnych badań, które wykorzystywałyby algorytmy genetyczne do wyznaczania liniowych modeli predykcyjnych, doprowadziły do znalezienia niewielu pozycji. Jednak np. w pracach takich jak [96, 108, 152] algorytmy były projektowane dla modeli predykcyjnych małego rzędu ($r = 3$ oraz $r = 4$), a w takich przypadkach metoda wybiórczego poszukiwania współczynników sprawdza się bardzo dobrze i trudno wykazać jej wady

w porównaniu z innymi technikami, w tym rozwiązaniami wykorzystującymi algorytmy genetyczne. Oczywiście istnieją i inne rozwiązania modelowania predykcyjnego, w tym nieliniowe, które do swych celów używają algorytmów genetycznych, jednakże odbiegają one tematycznie od zagadnień omawianych w rozdziale 2.

Aby uniknąć wpisania się w ramy myślowe rozwiązań opublikowanych w literaturze, należało zaprojektować samodzielnie cały schemat algorytmu genetycznego pasujący do określonych założeń, których zabrakło w większości dotychczasowych publikacji związanych z tą tematyką. Podstawowym warunkiem była reguła, że suma współczynników modelu predykcyjnego powinna wynosić 1, ponadto każdy ze współczynników powinien mieścić się w przedziale otwartym od –2 do 2. Przy okazji warto podkreślić ideę projektowania algorytmów genetycznych: dla każdego nowego zagadnienia minimalizacji należy opracować własne podejście wykorzystujące zasady działania poszczególnych kroków algorytmu genetycznego. Nie ma bowiem sprawdzonych reguł działających wysoce skutecznie dla wielu różnych zagadnień minimalizacji. Większości etapów projektowania poszczególnych bloków funkcjonalnych (i ich odmian omawianych w literaturze związanej z algorytmami genetycznymi) i wykonanych eksperymentów testujących skuteczność wstępnych rozwiązań nie zaprezentowano w tej pracy. Skupiono się na opisie ostatecznego rozwiązania, przy którym uzyskano najlepsze rezultaty.

2.5.2. Podstawy projektowania algorytmu genetycznego

Pierwszym krokiem projektowania przez autora własnego algorytmu genetycznego było ustalenie funkcji przystosowania, czyli funkcji celu. W badaniach zastosowano dwie funkcje. Podstawowa była inspirowana minimalizacją średniego błędu w sensie odległości Minkowskiego (przy $M = 2$ otrzymujemy odległość euklidesową, dla której można w prosty sposób znaleźć rozwiązanie minimalizujące funkcję celu, wykorzystując metodę MMSE, którą opisano w podrozdziale 2.2):

$$L_M = \left(\sum_{y=1}^{\text{wys}} \sum_{z=1}^{\text{szer}} |e|^M \right)^{\frac{1}{M}}. \tag{2.9}$$

Drugą funkcją celu była entropia rzędu zerowego (stosowana w badaniach w podrozdziale 2.4, a także w pracach [96, 152]) obliczana dla zbioru błędów predykcji i to wynik tej funkcji, po końcowym zakodowaniu obrazu, jest traktowany jako ostateczna ocena efektywności wyznaczonego modelu predykcyjnego. Badania przeprowadzono dla obu funkcji, gdyż nie zawsze na etapie udoskonalania chromosomów w kolejnych epokach bardziej skomplikowana funkcja entropii (docelowa) musi być bardziej pomocna od uproszczonego modelu funkcji celu, co potwierdzają badania eksperymentalne.

Minimalizacja każdej z tych funkcji oznacza próbę wyznaczenia najlepszego chromosomu, który należy zaprojektować jako zbiór genów, jakimi w tym przypadku są współczynniki predykcji. W skład każdego współczynnika wchodzą $n + 2$ bity. Pierwszy z nich oznacza

znak liczby, drugi wartość części całkowitej współczynnika, a pozostałe n bitów to część ułamkowa. Łącznie zatem chromosom składa się z $r\cdot(n+2)$ bitów. Do celów poprawnego działania algorytmu potrzeba całej populacji chromosomów roboczych, eksperymentalnie ustalono ich liczbę na $N_{\mathrm{cr}} = 200$.

Inicjalizacja tych chromosomów nie jest przeprowadzana w sposób losowy, co pozwala uzyskać szybszą zbieżność efektywnego działania algorytmu. Pierwsze 30 chromosomów to stałe modele zamieszczone w tab. 2.1. Pozostałe powstają przez losowy wybór jednego z tych 30 stałych modeli, dla którego następnie jest losowany jeden z jego niezerowych współczynników, po czym jego wartość jest losowo dodawana do innego współczynnika oraz odejmowana od kolejnego losowo wybranego współczynnika.

Po zakończeniu inicjalizacji wielokrotnie jest wykonywany proces (epoka) składający się z trzech etapów: selekcji, krzyżowania i mutacji [40].

Podczas etapu selekcji przetestowano metody ruletki i nadawania rang, a także pojedynczą i podwójną metodę turniejową. Przy projektowaniu ostatecznie wybrano selekcję turniejową, w której z populacji jest losowana ustalona liczba N_{tour} chromosomów rywalizujących następnie o możliwość pojawienia się w nowej populacji. Liczbę N_{tour} wyznaczamy z eksperymentalnie dobranego wzoru:

$$N_{\mathrm{tour}} = \left\lfloor \frac{N_{\mathrm{cr}}}{25} + 1 \right\rfloor, \tag{2.10}$$

gdzie N_{cr} jest liczbą chromosomów w populacji. W turnieju zwycięzcą jest ten, który uzyskuje najmniejszą wartość funkcji celu L_M. Proces powtarzany jest do momentu wypełnienia nowej populacji, przy czym chromosomy, które już brały udział w turnieju, nadal mogą zostać wybrane.

W procesie krzyżowania dochodzi do modyfikacji genów na zasadzie zastąpienia rodziców (dwóch dotychczasowych chromosomów) ich dwoma potomkami, które otrzymujemy na podstawie z góry określonych reguł. W projektowanym algorytmie brano pod uwagę krzyżowanie wielopunktowe i krzyżowanie arytmetyczne. Jako docelowe zaakceptowano krzyżowanie arytmetyczne pozwalające zachować zasadę mówiącą, że suma współczynników modelu predykcyjnego wynosi 1. Otrzymanie nowych genów G_c i G_d na podstawie genów G_a i G_b następuje dzięki przekształceniom (operacje te są wykonywane równolegle na wszystkich genach danego chromosomu, interpretowanych jako współczynniki predykcji, czyli zmiennoprzecinkowe liczby rzeczywiste, a nie ciąg bitów):

$$\begin{aligned} G_c &= k\cdot G_a + (1-k)\cdot G_b, \\ G_d &= k\cdot G_b + (1-k)\cdot G_a, \end{aligned} \tag{2.11}$$

przy czym parametr k jest losową liczbą z przedziału od 0 do 1. Jeśli na wejściu były dwa identyczne geny ($G_a = G_b$), to przetrwają one bez zmian. W projektowanym rozwiązaniu

przyjęto eksperymentalnie, że dobre rezultaty są uzyskiwane, gdy w przybliżeniu co druga para rodzicielska jest poddawana procesowi krzyżowania.

Kolejny etap to mutacja, która zapobiega zbyt szybkiej zbieżności do lokalnego minimum. W projektowanym algorytmie zastosowano zamianę jednego bitu na przeciwny w losowo wybranym genie losowo wybranego chromosomu. Ustalono nierównomierny rozkład prawdopodobieństwa wylosowania danego bitu w ramach genu, w którym dochodzi do mutacji. Wynika to z faktu, iż chcemy, aby częściej pojawiały się modyfikacje mniej znaczących bitów współczynnika predykcji niż tych bardziej znaczących. Przyjmując n-bitową reprezentację części ułamkowej, prawdopodobieństwo $P_{\text{bit}}(i)$ wylosowania i-tego bitu, który poddany zostanie procesowi mutacji, wyliczamy ze wzoru:

$$P_{\text{bit}}(i) = \frac{i^2}{\sum_{j=1}^{n} j^2}. \tag{2.12}$$

Dodatkowo po zakończeniu każdej epoki jest wybierany najlepiej przystosowany chromosom i kopiowany do specjalnej tabeli po to, aby nie uległ zniszczeniu, gdyby okazało się, że w kolejnych epokach wartości funkcji przystosowania dla najlepszego chromosomu okazały się gorsze. W ten sposób ostatecznym modelem predykcyjnym będzie ten, który na etapie całego procesu modyfikacji genetycznych pozwolił uzyskać najmniejszą wartość funkcji celu.

2.5.3. Badania eksperymentalne

Aby móc porównać rezultaty otrzymane z użyciem algorytmów genetycznych z propozycją opisaną w podrozdziale 2.4.2, zdecydowano się na rząd $r = 8$ oraz 39 epok, dzięki czemu czas kodowania obu metod (około 515 s) był porównywalny. Po dokonaniu wielu pomiarów w celu uzyskania najlepszych rezultatów, dobrano następujące parametry: $N_{\text{cr}} = 200$ chromosomów, $n = 11$ bitów części ułamkowej każdego współczynnika predykcji. W czwartej i piątej kolumnie tab. 2.6 umieszczono wyniki pomiarów opisanej tu metody wyznaczania współczynników predykcji z wykorzystaniem algorytmu genetycznego dla, odpowiednio, funkcji celu wyrażonej wzorem (2.9) przy $M = 0{,}5$ oraz funkcji celu zdefiniowanej jako entropia rzędu zerowego obliczana dla zbioru błędów predykcji. Średnia z drugiej funkcji celu okazała się wyższa, lecz jest to spowodowane tym, iż nie zawsze algorytm genetyczny radzi sobie ze znalezieniem dostatecznie dobrego modelu w przypadku bardziej nietypowych obrazów, takich jak Shapes czy Omaha. A zatem, mimo że funkcja celu wzorowana na odległości Minkowskiego była lepsza jedynie w 14 na 45 przypadków, to należy ją uznać za stabilniejszą, co potwierdza wynik średniej entropii uzyskanej z pomiaru wszystkich 45 obrazów testowych.

Tab. 2.6. Pomiar entropii dla predyktorów ósmego rzędu

Obrazy	Metoda z pod-rozdziału 2.4.2	Metoda z pod-rozdziału 2.4.3	Algorytm genetyczny	Algorytm genetyczny (funkcja entropii)	Algorytm genetyczny + MMSE	Algorytm genetyczny + MMSE (funkcja entropii)	Algorytm genetyczny + MMSE (rozszerzony)
Aerial	5,21144	5,19565	5,20470	5,21049	5,20818	5,20834	5,20420
Airfield	5,16339	5,15814	5,17230	5,16856	5,16917	5,16649	5,17028
Airplane	4,16864	4,15363	4,16359	4,16152	4,16343	4,16314	4,16138
Baboon	6,17336	6,16558	6,16655	6,16651	6,16636	6,16600	6,16638
Barbara	5,20475	5,20083	5,22070	5,22044	5,20222	5,20225	5,20236
Boat	4,95178	4,94260	5,03298	4,99321	4,94493	4,94500	4,94539
Bridge	3,75886	3,74515	3,75859	3,75360	3,75009	3,74735	3,75570
Couple	4,53673	4,51919	4,52192	4,52145	4,53210	4,53057	4,52295
Crowd	4,23606	4,21897	4,23323	4,23801	4,23570	4,23279	4,23539
Elaine	4,88498	4,87351	4,89697	4,87736	4,87986	4,87787	4,87835
Finger	5,31239	5,30375	5,33694	5,33021	5,32789	5,32746	5,32471
Frog	4,76723	4,76723	4,79835	5,02421	4,79835	5,02328	4,79835
Goldhill	4,89062	4,88041	4,88927	4,88593	4,88705	4,88721	4,88364
Harbour	5,08529	5,06647	5,11233	5,07939	5,11233	5,08493	5,11233
Lax	5,87280	5,86453	5,86861	5,86555	5,86587	5,86552	5,86502
Lennagrey	4,39177	4,37965	4,38588	4,37994	4,38213	4,38019	4,38148
$Lena_{TMW}$	4,73156	4,72527	4,73705	4,72633	4,72703	4,72637	4,72805
Man	4,76189	4,74674	4,76062	4,75875	4,75632	4,75587	4,75624
Peppers	4,59458	4,58972	4,59421	4,60011	4,60277	4,60026	4,59205
Sailboat	4,98878	4,98034	4,98417	4,98449	4,99088	4,98855	4,99096
Seismic	2,41814	2,40828	2,94280	2,73109	2,94280	2,71580	2,44511
Shapes	1,35035	1,35035	1,35119	1,68529	1,35119	1,66893	1,35119
Tank	4,01941	4,01535	4,01972	4,01615	4,01639	4,01622	4,01657
Truck	4,30433	4,29711	4,30275	4,30372	4,29860	4,29824	4,29850
Woman1	4,33859	4,33370	4,34174	4,34236	4,33651	4,33598	4,33600
Woman2	3,36453	3,35957	3,36576	3,36623	3,36665	3,36785	3,36593
Balloon	3,06651	3,04946	3,06144	3,05996	3,05747	3,06842	3,06173
Barb	4,98012	4,96342	4,96560	4,96478	4,96582	4,96508	4,96596
Barb2	5,03901	5,02491	5,03406	5,03383	5,03122	5,03006	5,02953
Board	3,97487	3,96350	3,99327	4,00446	3,97069	3,97865	3,97176
Boats	4,32026	4,30500	4,36736	4,35176	4,30965	4,30729	4,31899
Girl	4,06299	4,04597	4,05228	4,05410	4,08454	4,08088	4,06947
Gold	4,72386	4,71377	4,71777	4,73109	4,72539	4,72393	4,72015
Hotel	4,77407	4,75925	4,77251	4,76165	4,78835	4,78680	4,76716
Zelda	3,88649	3,87282	3,89790	3,89421	3,89030	3,90117	3,88219
Bridge256	5,89550	5,88870	5,89609	5,89424	5,89115	5,88970	5,89154
Camera	4,82224	4,81247	4,81839	4,81629	4,81997	4,81555	4,81971
Couple256	4,09211	4,07778	4,09792	4,08408	4,08976	4,08556	4,08632
Earth	3,73029	3,71457	3,74105	3,74002	3,72378	3,73170	3,72436
Elif	3,13030	3,11491	3,12485	3,12432	3,12160	3,11841	3,12097
Noisesquare	5,33370	5,30909	5,30951	5,30395	5,30963	5,30969	5,30961
Omaha	5,96855	5,96855	5,96857	6,09212	5,96857	6,09271	5,96857
Sena	3,37877	3,35098	3,38256	3,37351	3,36576	3,36624	3,36528
Sensin	3,57070	3,53142	3,56599	3,56311	3,56132	3,56069	3,55246
Sinan	3,31807	3,29095	3,32643	3,33338	3,29333	3,29011	3,29175
Średnia	4,43446	4,42221	4,45014	4,45715	4,44407	4,45300	4,43124

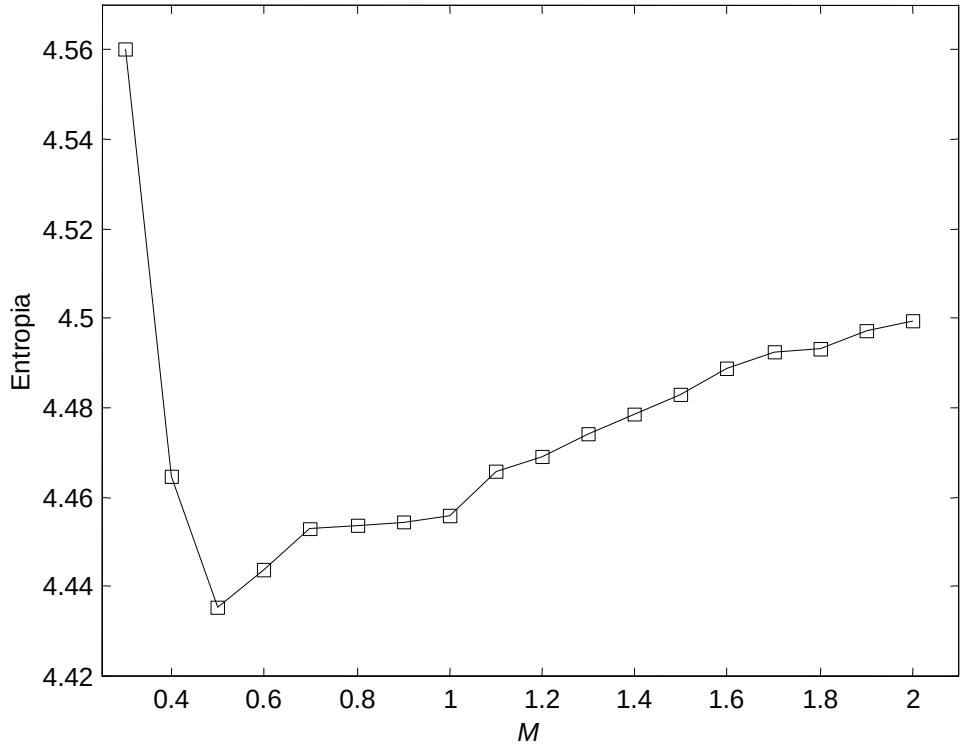
Pierwsze badanie wykazało, że dla małych rzędów predykcji czas wykonania wszystkich operacji w ramach jednej epoki jest na tyle znaczący, że zalecane użycie 50 epok wymaga (przy $r = 8$, $N_{cr} = 200$) aż 11 min obliczeń. Aby zwiększyć efektywność proponowanej tu metody, wprowadzono dodatkową zmianę w procesie inicjalizacji. Oprócz 30 stałych predyktorów podanych w tab. 2.1 dla każdego kodowanego obrazu wygenerowano także metodą MMSE statyczne modele predykcyjne o rzędach od $r = 3$ do $r = 8$. Udało się dzięki temu uzyskać pewien wzrost efektywności, lecz żadna z dwóch funkcji przystosowania (kolumny szósta i siódma w tab. 2.6) nie pozwoliła otrzymać średniej wartości entropii tak dobrej, jak ta, którą uzyskano metodą opisaną w podrozdziale 2.4.2. Jednak główną zaletą zaprojektowanej tu i zoptymalizowanej metody wykorzystującej algorytm genetyczny jest to, że ma ona złożoność wielomianową, w przybliżeniu liniowo zależną od liczby epok czy rzędu predykcji, przy czym czas wyznaczania modelu predykcyjnego nie zależy od zawartości danych obrazowych, ale jedynie liniowo od liczby pikseli obrazu. Po zastosowaniu więc dwukrotnie większej liczby chromosomów ($N_{cr} = 400$) i zwiększeniu liczby epok do 50 uzyskano średnią wartość entropii niższą (przy zachowaniu rzędu predykcji $r = 8$, patrz ostatnia kolumna tab. 2.6) niż w przypadku iteracyjnego przeszukiwania wybiórczego opisanego w podrozdziale 2.4.2. Nadal jest to jednak rezultat gorszy od otrzymanego w podrozdziale 2.4.3, co pokazuje, że trudno jest zaprojektować wysokoefektywny algorytm genetyczny, który pozwalałby uzyskiwać oczekiwaną przez użytkownika odpowiednio niską wartość entropii.

Dla tych samych parametrów wykonano także testy wydajności proponowanego tu algorytmu genetycznego dla rzędu $r = 30$, co nie byłoby możliwe w przypadku algorytmów przeszukiwania o złożoności wykładniczo zależnej od rzędu predykcji. Wyniki porównano z metodą statycznej predykcji liniowej, czyli z modelami predykcyjnymi rzędu $r = 30$ uzyskanymi metodą MMSE (patrz tab. 2.7). Nawet przy tak wysokim rzędzie predykcji daje się zauważyć znaczną przewagę efektywności uzyskanej dzięki użyciu algorytmu genetycznego. Należy przypuszczać, iż poprawę tę zawdzięczamy wykorzystaniu innej potęgi M w funkcji celu. Uzasadnienie, że MMSE jako funkcja celu nie przekłada się na najmniejszą wartość entropii, zaprezentowano w pracy [133]. Analizę wpływu parametru M na średnią wartość entropii przedstawiono na rys. 2.3 (pomiary wykonane przy $r = 9$). Z badań wynika, iż najlepszą wartością przy wykorzystaniu algorytmów genetycznych jest $M = 0{,}5$.

Aby zweryfikować tę zależność, podobne badanie (przy $r = 9$) przeprowadzono dla metody przeszukiwania wybiórczego opisanego w podrozdziale 2.4.3, w którym funkcję celu w postaci wartości entropii zastąpiono minimalizacją funkcji ze wzoru (2.9). Również i w tym przypadku otrzymano podobny wykres. Po uważnym przyjrzeniu się rezultatom dla poszczególnych obrazów okazało się, że wyniki entropii dla dwóch z nich (Frog, Omaha) odbiegają od ogólnej tendencji, bowiem dla potęg $M \leq 0{,}5$ entropia znacząco spada (o kilka dziesiątych bitu na piksel). W przypadku pozostałych 43 obrazów zmiany wartości entropii w badanym zakresie $M \in \langle 0{,}3;\ 2{,}0\rangle$ wynoszą średnio kilka setnych bitu na piksel. Po odrzuceniu tych dwóch obrazów (traktując je podobnie do pomiarowych błędów grubych), które znacząco zakłócają uśredniony wynik, ponownie wyznaczono średnie wartości entropii dla 43 obrazów, co przedstawiono na rys. 2.4. Z tych badań wynika, że najlepsze i porównywalne rezultaty możemy otrzymać w przedziale $M \in \langle 0{,}6;\ 0{,}9\rangle$, stąd za docelową i kompromisową wartość w tego typu badaniach można uznać potęgę $M = 0{,}75$.

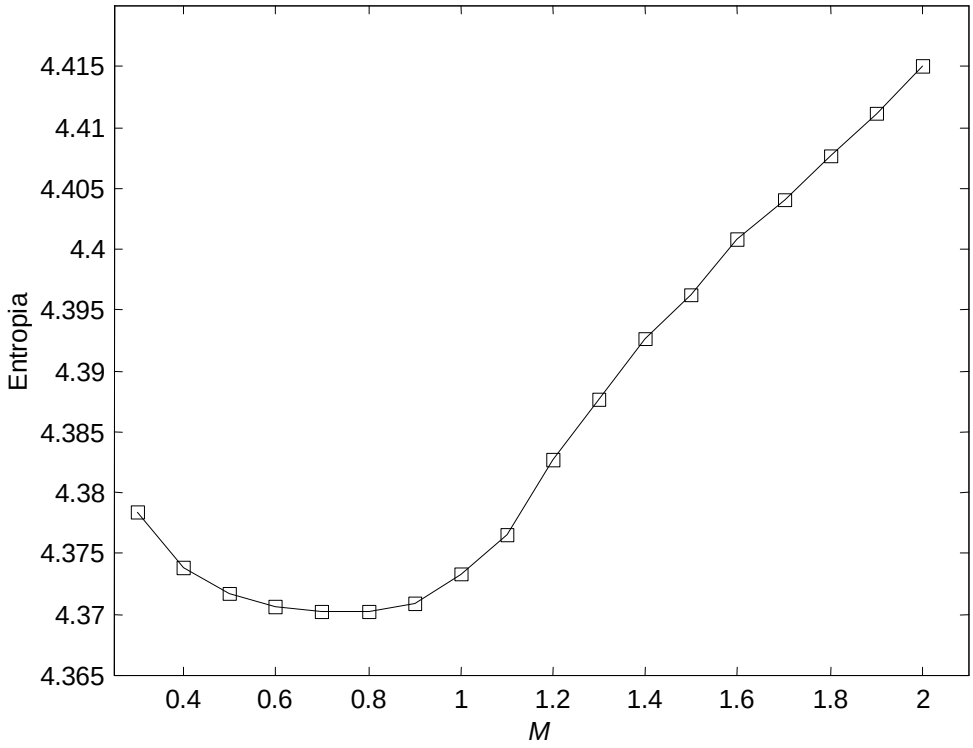
Tab. 2.7. Pomiar entropii dla predyktorów 30. rzędu

Obrazy	Metoda z podrozdziału 2.3.2	MMSE (r = 30)	Algorytm genetyczny + MMSE (r = 30)	Metoda z selekcją	Metoda z podrozdziału 2.4.3
Aerial	5,29816	5,18142	5,17547	5,17547	5,15729
Airfield	5,19447	5,12229	5,12775	5,12229	5,10387
Airplane	4,22887	4,21886	4,14987	4,14987	4,13186
Baboon	6,20065	6,14199	6,14782	6,14199	6,13762
Barbara	5,48057	5,09690	5,16234	5,09690	5,06890
Boat	5,12724	4,84578	4,89074	4,84578	4,82635
Bridge	3,77570	3,73908	3,74295	3,73908	3,73128
Couple	4,55002	4,51227	4,51665	4,51227	4,49184
Crowd	4,38319	4,20102	4,19964	4,19964	4,17375
Elaine	4,93586	4,62805	4,81892	4,62805	4,62237
Finger	5,54495	5,21621	5,25686	5,21621	5,21782
Frog	4,76723	5,14604	4,79835	4,76723	4,76723
Goldhill	4,90045	4,84216	4,87233	4,84216	4,83352
Harbour	5,10351	5,09612	5,11233	5,09612	5,04635
Lax	5,88633	5,86476	5,85876	5,85876	5,84876
Lennagrey	4,46329	4,37445	4,35314	4,35314	4,34322
Lena_{TMW}	4,79786	4,72228	4,70578	4,70578	4,69536
Man	4,79933	4,75344	4,74547	4,74547	4,73288
Peppers	4,69326	4,64119	4,58519	4,58519	4,57697
Sailboat	5,05211	4,96597	4,96813	4,96597	4,92044
Seismic	2,80187	2,22744	2,38488	2,22744	2,24783
Shapes	1,35035	2,50455	1,35119	1,35035	1,35035
Tank	4,03404	4,01065	4,00551	4,00551	4,00112
Truck	4,35726	4,29074	4,28709	4,28709	4,28357
Woman1	4,36638	4,34065	4,32235	4,32235	4,31349
Woman2	3,52173	3,34085	3,34258	3,34085	3,33542
Balloon	3,12768	3,08732	3,04302	3,04302	2,99551
Barb	5,25347	4,85869	4,91215	4,85869	4,82066
Barb2	5,16264	5,00317	5,01502	5,00317	4,94947
Board	4,14415	3,92325	3,96744	3,92325	3,85788
Boats	4,40189	4,23539	4,28600	4,23539	4,19947
Girl	4,20951	3,97733	4,00727	3,97733	3,92722
Gold	4,74809	4,68649	4,71248	4,68649	4,67339
Hotel	4,85425	4,75498	4,76125	4,75498	4,70562
Zelda	4,04126	3,86282	3,86194	3,86194	3,80517
Bridge256	5,91336	5,88212	5,88754	5,88212	5,87500
Camera	4,81700	4,92325	4,81977	4,81700	4,80574
Couple256	4,09306	4,10085	4,09263	4,09263	4,05495
Earth	3,73941	3,74357	3,72276	3,72276	3,71004
Elif	3,34146	2,99400	3,10075	2,99400	2,95939
Noisesquare	5,31577	5,22948	5,23975	5,22948	5,22570
Omaha	5,96855	6,50998	5,96857	5,96855	5,96855
Sena	3,56751	3,23808	3,31267	3,23808	3,20916
Sensin	3,84714	3,48248	3,54852	3,48248	3,38026
Sinan	3,56938	3,15185	3,25455	3,15185	3,15537
Średnia	4,52734	4,43712	4,40880	4,38231	4,36084



Rys. 2.3. Zależność średniej entropii od potęgi *M*

Analizując wyniki uzyskane klasyczną metodą MMSE dla $r = 30$ oraz metodą wykorzystującą proponowany w tym podrozdziale algorytm genetyczny, można zauważyć, że w wielu przypadkach entropia była niższa, gdy użyto wyłącznie MMSE.



Rys. 2.4. Zależność średniej entropii od potęgi *M*

Stąd prosty wniosek, aby ostateczną metodę zaprojektować jako selektywną, wybierającą najlepsze rozwiązanie po uwzględnieniu wyników obu tych metod, a także wyniku działania prostej metody doboru współczynników opisanej w podrozdziale 2.3.2 (metoda ta okazała się najlepsza w czterech przypadkach na 45 testowanych obrazów). W ten sposób uzyskujemy algorytm z selekcją (oparty na rozwiązaniach klasycznych i algorytmie genetycznym), który daje jeszcze lepsze rezultaty, co pokazuje przedostatnia kolumna tab. 2.7. Nadal jednak są to średnio gorsze wyniki niż uzyskane metodą z podrozdziału 2.4.3, zaprezentowane w ostatniej kolumnie tab. 2.7, co pokazuje, że zaprojektowanie wysoce efektywnej metody z wykorzystaniem algorytmów genetycznych jest niezwykle trudnym zadaniem.

2.6. Metody modelowania predykcyjnego z podziałem blokowym

Wykorzystując zasadę mówiącą o różnorodności obszarów w ramach pojedynczego obrazu, można dokonać jego podziału na bloki (np. o wielkości 8×8 lub 16×16 pikseli). Każdemu blokowi przypisuje się indywidualny model predykcyjny. Jednym z pierwszych tego typu rozwiązań była metoda zaprezentowana w artykule [81], w której każdemu blokowi 8×8 pikseli przypisywano jeden z ośmiu stałych modeli – ten, który dawał najmniejszy błąd bezwzględny. Informacja nagłówkowa skojarzona z tym blokiem wymagała trzech bitów, za pomocą których identyfikowany był numer modelu predykcyjnego.

Kolejne rozwinięcia wprowadzały minimalizację błędu średniokwadratowego jako metodę wyznaczania najlepszego zestawu współczynników predykcji. Wiązało się to jednak z bardzo dużą informacją nagłówkową, gdyż zapis współczynników predykcji wymagał znacznej liczby bitów. Aby zmniejszyć wielkość nagłówka, bloki o podobnych cechach zaczęto grupować w klastry, kojarzone ze wspólnym modelem predykcyjnym [39]. Dzięki wykorzystaniu technik kwantyzacji wektorowej (a także klasteryzacji rozmytej [4]) tworzono zoptymalizowane zestawy np. 16 modeli predykcyjnych, dzięki czemu nawet przy dużym rzędzie predykcji ogólna wielkość nagłówka nie wpływała znacząco na średnią bitową. W pracy [78] wykorzystano technikę łączenia sąsiednich bloków, należących do tej samej kategorii (skojarzonych z tym samym predyktorem), w grupy, tworząc większe bloki. Następnie mapa tych bloków, o różnych wielkościach, była zapisywana z użyciem efektywnej techniki kodowania drzew czwórkowych.

Bardzo łatwo można udowodnić, że istnieją możliwości uzyskiwania współczynników predykcji pozwalających otrzymać niższe wartości entropii rzędu zerowego niż z użyciem metody MMSE (patrz tab. 2.3), co wykorzystano choćby w publikacji [78]. W pracy [42] zaproponowano minimalizację błędu absolutnego MMAE (ang. *Minimum Mean Absolute Error*), która pozwoliła w przypadku podziału blokowego uzyskać lepsze rezultaty w porównaniu z metodą klasyczną MMSE. Ze względu na wysoką złożoność algorytmu MMAE, wykazującą się długim czasem kodowania przy dużych zbiorach danych, metody tej nie wykorzystano w badaniach opisywanych w niniejszej pracy. Szersze uzasadnienie nieoptymalnego wpływu MMSE na wartość entropii zaprezentowano w pracy [133].

Interesującym rozwiązaniem jest też technika MEE [137], w której jako funkcję celu zastosowano wartość entropii, jednak brak jest praktycznej implementacji z wykorzystaniem tego pomysłu, która uzyskałaby wysoką efektywność.

W podrozdziale 2.5.3 przedstawiono rezultaty badań otrzymane techniką wybiórczego poszukiwania współczynników predykcji metodą opisaną w podrozdziale 2.4.3. Wynika z nich, że jeszcze lepszą miarą od MAE może być odległość Minkowskiego, z potęgą $M = 0{,}75$ (dla MAE oraz MSE potęga M wynosi, odpowiednio, 1 oraz 2). Może to być istotna sugestia przy projektowaniu metod kontekstowych (opis idei podziału kontekstowego zostanie wprowadzony w rozdziale 3) wykorzystujących wiele modeli predykcyjnych (w takiej sytuacji bowiem metody pomiaru entropii warunkowych w poszczególnych kontekstach mogą być nieefektywne i trudne w implementacji), przy czym należałoby powtórzyć w takim przypadku eksperyment z doborem najlepszej wartości M.

Jeśli porównać tego typu metody z podziałem blokowym (stosujące modele predykcji w przód) z metodami kontekstowymi, to mogą one być dość zbliżone co do zasady, co pokazuje metoda opisana w podrozdziale 4.4, w której z poszczególnymi kontekstami skojarzono odrębne (i zapisane w nagłówku pliku) zestawy współczynników predykcji. Jednak metody podziału kontekstowego główną przewagę nad metodami z podziałem blokowym uzyskują w rozwiązaniach, w których mamy do czynienia z adaptacyjnym procesem wyznaczania współczynników predykcji na bieżąco (patrz rozdział 5 i kolejne, w których opisano modele predykcji wstecz). Wynika to z faktu, iż przełączanie między kontekstami może następować w sposób nieregularny (każdy sąsiedni piksel może być skojarzony z innym kontekstem), natomiast w podziale blokowym mamy regularne podziały na kwadratowe bloki, i to nie zawsze w tych miejscach, gdzie występuje dość jednorodny obszar obrazu. Ponadto w przypadku podziału blokowego niezbędna jest dodatkowa informacja nagłówkowa dotycząca cech bloku (np. numer klastra skojarzonego z blokiem).

Na podstawie powyższych argumentów zdecydowano, że dalsze prace będą skoncentrowane na metodach z adaptacją wstecz, wykorzystujących podział kontekstowy i pozwalających przy tym uniknąć kłopotliwego bagażu w postaci dużego nagłówka pliku, co jest charakterystyczne dla rozwiązań z podziałem blokowym.

3. Adaptacyjna predykcja z przełączanym modelem

3.1. Stałe predyktory nieliniowe

3.1.1. Adaptacyjny predyktor medianowy

Kontekstem nazywamy zbiór cech charakteryzujących typ najbliższego otoczenia kodowanego piksela. Używając podziału kontekstowego przez detekcję różnych typów sąsiedztwa (klas o różnych cechach), możemy dokonywać indywidualizacji modeli predykcyjnych dobrze dobranych do cech otoczenia, co prowadzi do wzrostu efektywności kompresji.

Predykcję nieliniową modeli omówionych w dalszej części tego podrozdziału od typowego predyktora liniowego odróżnia między innymi występowanie schematu przełączania między zestawem kilku modeli predykcyjnych skojarzonych z poszczególnymi kontekstami (inne przykłady nieliniowości zaprezentowane zostaną między innymi w rozdziale 7).

Z każdym kontekstem jest skojarzony indywidualny predyktor (stały, statyczny lub adaptacyjnie zmienny). Tego rodzaju predyktory nieliniowe nazywa się czasem predyktorami z kodowaniem wielokanałowym [114] (ang. *multichannel predictive coders*). Pełną nieliniowością (nawet przt braku przełączania kontekstowego) charakteryzują się np. omówione w rozdziale 7 modele predykcyjne oparte na sieciach neuronowych.

W tym podrozdziale zostaną przedstawione proste metody przełączania skojarzone ze stosunkowo prostymi predyktorami, które charakteryzują się niską złożonością implementacyjną, dzięki czemu można je wykorzystywać w rozwiązaniach zarówno sprzętowych, jak i programowych.

Zaproponowany w algorytmie JPEG-LS adaptacyjny predyktor medianowy MAP (ang. *Median Adaptive Predictor*), oparty na technice przełączania kontekstów MED (ang. *Median Edge Detector*), jest określany mianem predyktora nieliniowego [27, 83, 136]. Jest to zestaw trzech prostych predyktorów z przełączaniem kontekstowym. Konteksty są wyznaczane na podstawie trzech sąsiednich pikseli: $P(1)$, $P(2)$, $P(3)$. Wartość przewidywana jest obliczana zgodnie z następującą zasadą:

$$\hat{x}_{MED} = \begin{cases} \min(P(1), P(2)) \text{ dla } P(3) \geq \max(P(1), P(2)) \\ \max(P(1), P(2)) \text{ dla } P(3) \leq \min(P(1), P(2)) \end{cases} \tag{3.1}$$

oraz $\hat{x}_{MED} = P(1) + P(2) - P(3)$ w pozostałych przypadkach. Zasadę tę można opisać także jako:

$$\hat{x}_{MED} = \min(P(1), P(2), P(3)) + \max(P(1), P(2), P(3)) - P(3). \tag{3.2}$$

Jeśli powyższą zasadę zamienimy na postać algorytmu przełączającego konteksty, to otrzymamy metodę wyboru jednego z trzech kontekstów, z których każdy jest skojarzony ze stałym predyktorem:

```
if (P(1) > P(2)) {
        max = P(1);
        min = P(2);
}
else {
        max = P(2);
        min = P(1);
}
if (P(3) >= max) kontekst = 1;
else if (P(3) <= min) kontekst = 2;
else kontekst = 3;
```

Dla kontekstu numer 1 $\hat{x}_{MED} = \min(P(1), P(2))$, dla kontekstu numer 2 $\hat{x}_{MED} = \max(P(1), P(2))$, a dla kontekstu numer 3 $\hat{x}_{MED} = P(1) + P(2) - P(3)$. Powstało kilka rozwinięć tego pomysłu [36, 50], jednak uzyskana poprawa nie była znaczna w porównaniu z metodami o większej liczbie kontekstów, których zasady działania zostaną zaprezentowane w kolejnych podrozdziałach. Na przykład algorytm opisany w pracy [50] polega na poszerzeniu liczby kontekstów do siedmiu dzięki wprowadzeniu dodatkowych progów (t_1 i t_2) odnoszących się do podziału na małe i duże gradienty (np. pionowy $P(2) - P(4)$) w najbliższym sąsiedztwie kodowanego piksela:

```
if (P(1) > P(2)) {
        max = P(1);
        min = P(2);
}
else {
        max = P(2);
        min = P(1);
}
if (P(3) >= max) {
        if (P(1)<=P(2)) kontekst = 1;
        else if ( (P(2) - P(4)>=t1) && (P(3) - P(1)>=t2) ) kontekst = 2;
        else kontekst = 3;
}
else if (P(3) <= min) {
        if (P(1)>=P(2)) kontekst = 4;
        else if ( (P(4) - P(2)>=t1) && (P(1) - P(3)>=t2) ) kontekst = 5;
        else kontekst = 6;
}
else kontekst = 7;
```

Dla kontekstów numer 1 i 4 $\hat{x}_{\mathrm{MED}_+} = P(1)$, dla kontekstów numer 2 i 5 $\hat{x}_{MED_+} = 0{,}5 \cdot (P(2) + P(4))$, a dla kontekstów numer 3 i 6 $\hat{x}_{MED_+} = P(2)$. Podobnie jak w pierwotnej wersji wartość przewidywana ostatniego, w tym przypadku siódmego kontekstu, $\hat{x}_{MED_+} = P(1) + P(2) - P(3)$.

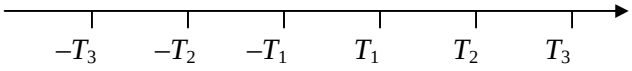
3.1.2. Metoda predykcji z podziałem kontekstowym GAP

Metoda predykcji z podziałem kontekstowym GAP (ang. *Gradient-Adjusted Predictor*) o stałych współczynnikach każdego modelu została zaproponowana jako wstępna metoda predykcji w algorytmie CALIC [141]. W odróżnieniu od opisanej w poprzednim podrozdziale metody MED, która wykorzystuje trzy sąsiednie piksele do wyznaczania wartości przewidywanej i doboru jednego z trzech kontekstów, metoda GAP korzysta z siedmiu sąsiednich pikseli do wyznaczania wartości przewidywanej i pozwala na wybór jednego z siedmiu kontekstów. Są to piksele o numerach {1, 2, 3, 4, 5, 6, 9}. Decyzja o wyborze odpowiedniego kontekstu jest podejmowana na podstawie liczby $d_{GAP} = d_h - d_v$, gdzie d_h i d_v zdefiniowano jako poziomy wychylenia kierunkowego (gradienty) sąsiedztwa [141]:

$$\begin{aligned} d_h &= \left|P(1) - P(5)\right| + \left|P(2) - P(3)\right| + \left|P(4) - P(2)\right|, \\ d_v &= \left|P(1) - P(3)\right| + \left|P(2) - P(6)\right| + \left|P(4) - P(9)\right|. \end{aligned} \tag{3.3}$$

Zakres wartości d_{GAP} jest podzielony na siedem przedziałów na podstawie trzech wartości progowych wyznaczonych eksperymentalnie przez autorów metody (kwantyzacja skalarna o sześciu progach, co pokazano na rys. 3.1). Dla obrazów o 256 odcieniach szarości wartości bezwzględne progów to, odpowiednio, $T_1 = 8$, $T_2 = 32$, $T_3 = 80$ (badania wykonane w pracy [128] doprowadziły do zmiany tych progów do wartości $T_1 = 6$, $T_2 = 25$, $T_3 = 78$). Poniżej przedstawiono w postaci pseudokodu algorytm wyznaczania numeru kontekstu [141]:

```
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;
}
```



Rys. 3.1. Siedmiopoziomowy kwantyzator skalarny

Każdemu z kontekstów przyporządkowano indywidualny stały predyktor liniowy wykorzystujący od jednego do czterech sąsiednich pikseli. Tabela 3.1 zawiera współczynniki predykcji b_j dla każdego z siedmiu kontekstów, przy czym wartość przewidywaną wyznacza się zgodnie ze wzorem (1.2).

Zmodyfikowaną wersję metody GAP zaprezentowano w publikacji [132]. Charakteryzuje się ona nieco wyższą efektywnością i w niniejszej pracy będzie oznaczana jako GAP+. Każdemu z siedmiu kontekstów przyporządkowano indywidualny stały predyktor liniowy

wykorzystujący od jednego do pięciu spośród sześciu sąsiednich pikseli. Tabela 3.2 zawiera współczynniki predykcji dla każdego z siedmiu kontekstów.

Tab. 3.1. Zestaw współczynników predykcji odpowiadających poszczególnym kontekstom metody GAP

Współczynnik\kontekst	1	2	3	4	5	6	7
b_1	1/2	5/8	3/4	3/8	1/4	1	0
b_2	1/2	3/8	1/4	5/8	3/4	0	1
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

Tab. 3.2. 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

Istnieją też inne metody oparte na GAP, w tym wykorzystujące podział kontekstowy do zwiększenia efektywności technik opartych na algorytmach genetycznych [67, 96].

3.1.3. Metoda predykcji z wagami gradientowymi

Metoda GBSW (ang. *Gradient Based Selection and Weighting pixel predictor*), zaprezentowana w pracy [58], opiera się na gradientach kierunkowych wyznaczanych w podobny sposób, jak w metodzie GAP. Obliczane są cztery wartości:

$$\begin{aligned} d_w &= (2|P(1)-P(5)|+2|P(2)-P(3)|+2|P(3)-P(7)|+2|P(2)-P(4)| \\ &\quad +|P(6)-P(8)|+|P(6)-P(9)|)/10, \\ d_n &= (2|P(6)-P(2)|+2|P(1)-P(3)|+2|P(3)-P(8)|+2|P(4)-P(9)| \\ &\quad +|P(5)-P(7)|+|P(7)-P(11)|)/10, \\ d_{nw} &= (2|P(1)-P(7)|+2|P(2)-P(8)|+|P(3)-P(11)|+|P(4)-P(6)|)/6, \\ d_{ne} &= (2|P(5)-P(3)|+2|P(2)-P(9)|+|P(1)-P(2)|+|P(3)-P(6)|)/6. \end{aligned} \tag{3.4}$$

Z tymi wartościami są skojarzone, odpowiednio, predyktory $P(1)$, $P(2)$, $P(3)$ i $P(4)$. Następnie wyznacza się spośród tych czterech gradientów dwa o najmniejszej wartości, które stają się współczynnikami wagowymi modelu predykcyjnego będącego kombinacją liniową dwóch spośród czterech najbliższych sąsiadów. Wagi są kojarzone z predyktorami w taki sposób,

aby uzyskać najlepszą wartość przewidywaną (metodą na krzyż). Na przykład jeśli dwiema najmniejszymi wartościami są d_w oraz d_n, to wartość predykcji jest wyznaczana następująco:

$$\hat{x} = \frac{d_w \cdot P(2) + d_n \cdot P(1)}{d_w + d_n}. \tag{3.5}$$

Metodę tę można rozwinąć przez dodanie piątego gradientu d_{ave}, będącego średnią arytmetyczną czterech opisanych wzorem (3.4), a skojarzony z nim piąty predyktor jest wyznaczany jako Plane3 (patrz tab. 2.1). W sytuacji, gdy mianownik we wzorze (3.5) wynosi 0, wówczas wartość przewidywaną uzyskuje się z modelu Plane3. Tak usprawniona metoda GBSW$_+$ okazała się znacznie efektywniejsza od omówionych wcześniej odmian MED i GAP, co można zauważyć w tab. 3.3.

3.1.4. Metoda predykcji oparta na logice rozmytej

W tym podrozdziale zostanie omówiona metoda predykcji oparta na regułach logiki rozmytej FLBP (ang. *Fuzzy Logic-Based Predictor*) [111]. Choć w metodzie tej, podobnie jak w omówionej w poprzednim podrozdziale metodzie GBSW, nie użyto przełączania kontekstowego, ale cechą wspólną obu technik jest wykorzystywanie gradientów kierunkowych i skojarzonych z nimi prostych modeli predykcyjnych. W tej metodzie są wyznaczane cztery wartości:

$$\begin{aligned}
d_{90} &= \frac{\left|P(5)+P(7)+P(11)-P(1)-P(3)-(8)\right|}{3} + \left|\frac{P(1)+P(3)+(8)}{3} - \frac{P(2)+P(6)}{2}\right|,\\
d_{0} &= \frac{\left|P(6)+P(8)+P(11)-P(2)-P(3)-(7)\right|}{3} + \left|\frac{P(2)+P(3)+(7)}{3} - \frac{P(1)+P(2)}{2}\right|,\\
d_{45} &= \left|\frac{P(7)+P(8)}{2} - \frac{P(3)+P(5)+P(6)}{3}\right| + \left|\frac{P(3)+P(5)+P(6)}{3} - \frac{P(1)+P(2)}{2}\right|,\\
d_{135} &= \left|\frac{P(2)+P(8)-P(3)-P(11)}{2}\right| + \left|\frac{P(3)+P(11)-P(1)-P(7)}{2}\right|.
\end{aligned} \tag{3.6}$$

Z tymi wartościami są skojarzone, odpowiednio, predyktory $P(1)$, $P(2)$, $P(3)$ i $P(4)$. Autorzy po długich wyprowadzeniach opartych na właściwościach zbiorów rozmytych przedstawiają w uproszczonej postaci wartość przewidywaną jako:

$$\hat{x} = \frac{d_0^4 \cdot P_0 + d_{90}^4 \cdot P_{90} + d_{45}^4 \cdot P_{45} + d_{135}^4 \cdot P_{135} + \theta \cdot P_U}{d_0^4 + d_{90}^4 + d_{45}^4 + d_{135}^4 + \theta}, \tag{3.7}$$

gdzie (po dobraniu przez autora niniejszej pracy eksperymentalnych wag, innych niż sugerowane w pracy [111]) cztery podstawowe modele predykcyjne wyznacza się następująco:

$$\begin{aligned} P_0 &= P(1) + 0{,}45(P(1) - P(5)) = 1{,}45P(1) - 0{,}45P(5), \\ P_{90} &= P(2) + 0{,}45(P(2) - P(6)) = 1{,}45P(2) - 0{,}45P(6), \\ P_{45} &= P(4) + 0{,}1(P(4) - P(12)) = 1{,}1P(4) - 0{,}1P(12), \\ P_{135} &= P(3) + 0{,}1(P(3) - P(11)) = 1{,}1P(3) - 0{,}1P(11). \end{aligned} \tag{3.8}$$

Dodatkowy model P_U zaproponowano w pierwotnej wersji jako średnią arytmetyczną powyższych czterech modeli, jednak wersja FLBP$_+$ zoptymalizowana przez autora niniejszej pracy zakłada, że predyktorem P_U zostanie model Plane3 (patrz tab. 2.1). Wagę modelu P_U dobrano eksperymentalnie na poziomie $\theta = 50\,000$. Jak widać, wartość przewidywana jest kombinacją liniową pięciu prostych, odpowiednio uzasadnionych teoretycznie subpredyktorów, przy czym wagi gradientowe zostały podniesione do czwartej potęgi. Wyniki uzyskane tą metodą, mimo wprowadzenia do niej usprawnień, są jedynie w niewielkim stopniu lepsze od otrzymywanych metodą GAP$_+$ (patrz tab. 3.3).

3.1.5. Propozycja metody wielokontekstowej

Obok wcześniej wymienionych metod MED, GAP, GAP$_+$ pojawiają się wciąż nowe propozycje o różnej liczbie kontekstów, można tu wymienić choćby prace z lat 2001–2006 [19, 36, 47, 50, 58, 114], jak i zdecydowanie nowsze [12, 24, 51]. W tym podrozdziale zaproponowano autorską metodę wykorzystania stałych predyktorów z przełączaniem kontekstów SFP (ang. *Switching Fixed Predictor*) opartą na pomyśle zaczerpniętym z pracy [38]. Jest ona pewnym uogólnieniem wcześniej omówionych propozycji i wykorzystuje dużą liczbę 512 kontekstów.

Ze względu na specyfikę zagadnienia zasada wyznaczania numerów kontekstu wykorzystywanych w omawianej tu metodzie zostanie przedstawiona w podrozdziale 4.2.1, przy czym wartość obliczana ze wzoru (4.3) jest kwantyzowana do $L = 2$ przedziałów z progiem równym 300.

Dla każdego z 512 kontekstów dobiera się (na podstawie zbioru uczącego, czyli pewnej bazy obrazów) indywidualny stały predyktor, minimalizując wartość błędu średniokwadratowego (MSE) [98] lub błędu bezwzględnego (MAE) [81]. Badania wykonano z użyciem obu miar, ponieważ minimalizacja wartości wariancji nie gwarantuje uzyskania modelu dającego najmniejszą średnią entropię rzędu zerowego [133]. Proponowana metoda jest suboptymalna, gdyż dla danego kontekstu dobiera się współczynniki predykcji jedynie ze zbioru 30 stałych predyktorów przedstawionych w tab. 2.1. W przypadku takiego doboru predyktorów lepszym rozwiązaniem okazała się selekcja predyktorów dających najmniejszą średnią wartość bezwzględną błędów predykcji w danym kontekście (MMAE).

Innym sposobem doboru indywidualnych stałych predyktorów (dla każdego z 512 kontekstów) jest wyznaczenie ich jako modeli statycznej predykcji, uzyskanych metodą MMSE (patrz podrozdział 2.2) na podstawie zbioru uczącego składającego z 45 obrazów (innych niż testowe). Zakładamy przy tym dokładność zapisu każdego współczynnika z uży-

ciem dziewięciu bitów części ułamkowej (do tego jeden bit znaku i jeden bit wartości całkowitej przy założeniu, iż współczynnik predykcji mieści się w przedziale od –1,999 do 1,999). Predyktory z tak wyznaczonymi współczynnikami stają się wówczas predyktorami stałymi, przy czym najlepsze rezultaty otrzymano dla rzędu $r = 4$ (metoda SFP-MMSE). Podobne rozwiązanie zaprezentowano w pracy [38], gdzie zaproponowano podział na 1024 konteksty i zapis współczynników na 10 bitach, a także w pracy [63], gdzie wykorzystano podział na 81 kontekstów i modele czwartego rzędu.

W SFP-MMSE uwzględniono też fakt, iż dla kontekstów najrzadziej występujących (poniżej 50 wystąpień podczas całego pomiaru) nie wyznacza się współczynników predykcji statycznej metodą MMSE (zbyt mała, niereprezentatywna grupa), a w takich sytuacjach jest używany predyktor Pirsha (w pracy [38] predyktor JPEG7).

W tabeli 3.3 przedstawiono wyniki pomiarów otrzymanych technikami SFP i SFP--MMSE, porównując je z kilkoma innymi metodami. Średnia uzyskana dla SFP, wynosząca 4,51594, jest rezultatem lepszym o 0,04224 bitu niż w przypadku zastosowania metody GAP_+.

Tab. 3.3. Porównanie wartości entropii różnych metod predykcyjnych

Obrazy	MED	MED_+	GAP_+	$FLBP_+$	SFP	$GBSW_+$	SFP--MMSE
Aerial	5,31868	5,31935	5,24967	5,26167	5,17998	5,20980	5,15877
Airfield	5,27145	5,27586	5,19937	5,23066	5,19110	5,23124	5,13778
Airplane	4,20351	4,18823	4,11734	4,19029	4,10897	4,22330	4,09689
Baboon	6,27482	6,28554	6,27328	6,19545	6,21740	6,16699	6,15692
Barbara	5,47998	5,46855	5,31285	5,07014	5,30201	5,08683	5,25235
Boat	5,10148	5,09825	4,98112	5,01490	5,02999	4,92471	5,02141
Bridge	3,77344	3,77335	3,76268	3,74341	3,74821	3,78725	3,70279
Couple	4,42937	4,42803	4,44654	4,42234	4,41417	4,43724	4,44374
Crowd	4,38610	4,36012	4,28135	4,38213	4,25298	4,33028	4,17651
Elaine	5,33756	5,33669	5,22085	5,11250	5,21767	5,01396	5,13662
Finger	5,64770	5,63975	5,56779	5,76125	5,60937	5,75338	5,49806
Frog	4,98944	4,99216	5,38566	5,30572	5,32159	5,25838	5,32275
Goldhill	4,87672	4,87665	4,86288	4,88707	4,86758	4,88998	4,83452
Harbour	4,99902	5,00587	5,07907	5,02512	5,01915	4,98079	5,03616
Lax	5,97575	5,98674	6,00103	5,94132	5,93673	5,88456	5,90344
Lennagrey	4,54677	4,52208	4,35782	4,40748	4,37583	4,35267	4,32789
$Lena_{TMW}$	4,89160	4,86930	4,72601	4,74865	4,74657	4,70654	4,68681
Man	4,81103	4,80533	4,70206	4,71786	4,69973	4,67962	4,66246
Peppers	4,94226	4,93268	4,77589	4,72983	4,77082	4,58786	4,70787
Sailboat	5,16648	5,16416	5,08202	5,10737	5,06793	5,02295	5,00197
Seismic	2,95392	2,95392	3,53394	3,21478	3,16960	3,18339	3,24242
Shapes	1,44798	1,44834	2,07669	2,19335	1,72139	1,57608	2,00087
Tank	4,10503	4,10362	4,02874	4,05305	4,03457	4,12887	3,98714
Truck	4,41386	4,41322	4,38331	4,39848	4,37057	4,46475	4,30848
Woman1	4,41667	4,41207	4,28785	4,25560	4,28664	4,19618	4,27757
Woman2	3,56804	3,55791	3,45634	3,48838	3,45200	3,51662	3,41055

Tab. 3.3. Porównanie wartości entropii różnych metod predykcyjnych (cd.)

Obrazy	MED	MED_+	GAP_+	$FLBP_+$	SFP	$GBSW_+$	SFP--MMSE
Balloon	3,12014	3,11139	2,99754	3,01889	2,96806	2,99869	2,98449
Barb	5,20357	5,20317	5,01219	4,91767	5,07967	4,93747	5,06526
Barb2	5,18095	5,17198	5,00262	5,04993	5,08242	4,98304	5,08742
Board	3,94724	3,93466	3,89214	3,91134	3,82831	3,76410	3,89409
Boats	4,30702	4,29491	4,22891	4,30633	4,24625	4,22130	4,28872
Girl	4,20682	4,18920	4,04393	4,08623	4,03455	3,95029	4,06129
Gold	4,71607	4,71525	4,69884	4,71859	4,69708	4,71654	4,66670
Hotel	4,73183	4,72792	4,67599	4,66952	4,61892	4,56691	4,63532
Zelda	4,11251	4,10784	3,93568	4,02397	3,99744	3,92202	3,98571
Bridge256	5,88975	5,89737	5,91929	5,95676	5,88833	5,90959	5,84487
Camera	4,73620	4,73364	4,72876	4,76860	4,66701	4,70075	4,68003
Couple256	3,98819	3,98797	4,01192	4,02020	3,99652	3,98553	3,99422
Earth	3,61430	3,61473	3,74950	3,80003	3,70731	3,76374	3,70685
Elif	3,34149	3,33213	3,40746	3,41073	3,31146	3,45333	3,26318
Noisesquare	5,72510	5,72588	5,71826	5,63236	5,68551	5,56294	5,63399
Omaha	6,00196	6,01848	6,51890	6,51851	6,22472	6,31907	6,35743
Sena	3,64079	3,63118	3,69707	3,66720	3,62242	3,77456	3,57268
Sensin	3,93245	3,90942	4,01042	3,99628	3,86203	4,03850	3,80674
Sinan	3,65990	3,63195	3,71670	3,71234	3,58690	3,73688	3,52957
Średnia	4,56411	4,55904	4,55818	4,55654	4,51594	4,50888	4,50114

Najlepszy rezultat (H = 4,50114 bitu na piksel) uzyskano metodą SFP-MMSE dla predykcji czwartego rzędu. Otrzymany zysk wyznaczania predyktorów statycznych metodą MMSE, w porównaniu z doborem najlepszego w danym kontekście spośród 30 stałych predyktorów (z użyciem kryterium MMAE), nie jest duży i wynosi zaledwie 0,0148 bitu na piksel. Różnica ta może w praktyce okazać się jeszcze mniejsza ze względu na zbyt dokładne dopasowanie poszczególnych predyktorów do wzorcowych obrazów uczących, co jest cechą charakterystyczną metody MMSE.

Ponadto wykorzystując zestaw 512 predyktorów czwartego rzędu, o 11-bitowym koszcie przypadającym na jeden współczynnik, wprowadzamy znaczny wzrost zapotrzebowania na pamięć, czego należy unikać w przypadku rozwiązań sprzętowych.

Pewnym kompromisem między liczbą kontekstów a efektywnością może być wyznaczenie metodą MMSE siedmiu stałych zestawów współczynników predykcji z przełączaniem kontekstowym zgodnym z zasadą GAP, co zaprezentowano w pracy [112].

3.2. Podział kontekstowy stosowany w złożonych metodach predykcyjnych

3.2.1. Zasada efektywnego doboru kontekstu głównego

W tym podrozdziale zostaną omówione podstawowe zasady konstruowania autorskich reguł podziału kontekstowego używanych w opracowanych przez autora nowych wersjach

metod predykcyjnych opartych na podstawowych technikach statycznej i adaptacyjnej predykcji liniowej. Opisy samych metod zamieszczono w kolejnych rozdziałach tej pracy.

Wzrost efektywności wielu metod jest możliwy dzięki wprowadzeniu podziału kontekstowego, co pokazano w podrozdziale 3.1. Dla odróżnienia dwóch kategorii kontekstów techniki dzielenia na małą liczbę kontekstów zostaną określone mianem metod z doborem kontekstu głównego. Natomiast metody wielokontekstowe zostaną opisane w rozdziale 4, gdyż związane są one głównie z omawianym tam usuwaniem skumulowanego błędu predykcji.

Numer kontekstu głównego to liczba określająca odrębny predyktor liniowy lub nieliniowy. W badaniach przedstawianych w tej pracy każdemu z tych kontekstów będzie przypisywany własny predyktor liniowy wyznaczany metodą MMSE lub jedną z metod adaptacyjnej korekcji wag opisanych w kolejnych rozdziałach.

W pracach [121, 128], wykorzystując predykcję statyczną (metoda MMSE) z przełączaniem kontekstowym, przeanalizowano kilka znanych metod decyzyjnych (patrz podrozdział 3.1) podziału na konteksty główne, takich jak MED [135] (trzy konteksty), GAP [141] (siedem kontekstów) czy propozycja własna korzystająca z 66 kontekstów [121]. Najsłabiej wypadł podział oparty na MED. Metoda z 66 kontekstami, wzorowana na pracy [38], pozwoliła uzyskać niewiele lepszy rezultat od metody siedmiokontekstowego podziału GAP, co wynika z potrzeby umieszczania w nagłówku pliku 66 zestawów współczynników predykcji.

Wraz ze wzrostem liczby kontekstów powinna zmniejszać się całkowita średnia bitowa kodowanego obrazu. Jednak wiele zależy od sposobu podziału na poszczególne konteksty. Poszukując kompromisu między złożonością obliczeniową a efektywnością, w poniżej omówionych badaniach zdecydowano się na trzy konteksty główne wraz z kilkoma uzupełniającymi. Podstawowymi kryteriami podziału na trzy konteksty są pomiar i klasyfikacja wariancji wagowej otoczenia (najbliższego sąsiedztwa złożonego z m najbliższych pikseli położonych po lewej stronie kodowanego piksela $P(0)$ lub powyżej niego). Wariancja wagowa, której wartości dzieli się na trzy przedziały (kwantyzacja skalarna z dwoma poziomami), uwzględnia odległości pikseli $P(j)$ branych pod uwagę przy obliczeniach. Wagę $\overline{d}_j$ wyznacza się jako odwrotność odległości euklidesowej między pikselami $P(j)$ a $P(0)$:

$$\overline{d}_j = \frac{1}{\sqrt{(\Delta x_j)^2 + (\Delta y_j)^2}}, \tag{3.9}$$

gdzie Δx_j oraz Δy_j to odległości (pozioma i pionowa) piksela $P(j)$ względem $P(0)$. Średnia wagowa z m pikseli najbliższego sąsiedztwa ma postać:

$$\tilde{p} = \frac{1}{\delta} \sum_{j=1}^{m} \overline{d}_j \cdot P(j), \tag{3.10}$$

gdzie:

$$\delta = \sum_{j=1}^{m} \overline{d}_j \,. \tag{3.11}$$

Wariancję wagową wyliczamy ze wzoru:

$$\tilde{\sigma}^2 = \frac{1}{\delta}\sum_{j=1}^{m} \bar{d}_j \cdot (P(j) - \tilde{p})^2 . \tag{3.12}$$

Podobnie jak w publikacji [127], w badaniach omawianych w niniejszej pracy zaproponowano podział wartości wariancji wagowych otoczenia $\tilde{\sigma}^2$ na trzy przedziały (mała, średnia oraz duża wariancja) z progami $\beta_1 \cdot \bar{\sigma}^2$ oraz $\beta_2 \cdot \bar{\sigma}^2$ (gdzie $\bar{\sigma}^2$ jest średnią arytmetyczną wszystkich wariancji wagowych $\tilde{\sigma}^2$ wyznaczonych dla kodowanego obrazu), przyjmując za otoczenie *m* najbliższych sąsiadów (pocieniowany obszar na rys. 1.1 przedstawia otoczenie o wielkości *m* = 10 pikseli). Poniżej podano eksperymentalnie dobrane parametry dla czterech metod podziału kontekstowego omawianych w tym rozdziale:

– dla metod: statycznej, ALCM$_+$, RLS$_+$, AdNN$_+$ (opisanych, odpowiednio, w podrozdziałach 4.4, 5.3, 6.1, 7.3) $\beta_1 = 0{,}15$, $\beta_2 = 1{,}3$, $m = 10$;

– dla metody CoBALP (opisanej w podrozdziale 5.4) $\beta_1 = 0{,}05$, $\beta_2 = 0{,}7$, $m = 30$.

3.2.2. Kontekst uzupełniający w metodzie statycznej predykcji liniowej

W przypadku opisanej w podrozdziale 4.4 statycznej metody predykcji z przełączaniem kontekstowym wraz ze wzrostem liczby kontekstów, a co za tym idzie, zwiększeniem liczby predyktorów powinna się zmniejszać całkowita średnia bitowa kodowanego obrazu. Przy wzroście liczby modeli predykcyjnych dużą rolę odgrywa odpowiedni sposób zapisu współczynników predykcji, średnia bitowa bowiem jest zależna od średniej liczby bitów potrzebnej do zakodowania piksela, ale także i od średniej bitowej nagłówka. Im więcej użytych predyktorów i im większe ich rzędy, tym średnia bitowa nagłówka staje się coraz istotniejszą przeszkodą wzrostu skuteczności takiej metody. Dobrym rozwiązaniem wydaje się zapis współczynników predykcji za pomocą od 11 do 14 bitów (1 bit znaku, 1 bit części całkowitej oraz od 9 do 12 bitów części ułamkowej). Rozwiązaniem kompromisowym użytym w tej pracy dla predyktorów wyznaczanych metodą MMSE jest zapis współczynników za pomocą 11 bitów.

Obok podziału na trzy konteksty główne (zgodnie z opisem w podrozdziale 3.2.1) w metodzie statycznej predykcji liniowej jako czwarty kontekst dołączono detekcję elementów krawędziowych, zaproponowaną w pracy [55]. W przypadku wykrycia w najbliższym otoczeniu kodowanego piksela krawędzi (fragmenty obrazu o dwóch odległych grupach kolorystycznych), jest stosowany czwarty kontekst skojarzony ze statycznym predyktorem rzędu *r* = 4. Ze względu na specyfikę pozostałych trzech kontekstów większe rzędy kontekstu krawędziowego nie dawały lepszych rezultatów. Wynika to z dość małej liczby wystąpień pikseli, dla których jest wyznaczany czwarty kontekst, a co za tym idzie, koszt zapisu współczynników predykcji jest dość duży w porównaniu z zyskiem otrzymywanym dzięki predyktorowi dobrze dopasowanemu do otoczenia krawędziowego.

Alternatywą pozwalającą na zwiększenie rzędu predykcji mogłaby być metoda predykcji adaptacyjnej, wymagałoby to jednak zwiększenia złożoności obliczeniowej po stronie dekodera.

Algorytm detekcji krawędzi (wyznaczania czwartego kontekstu) polega na obliczeniu wariancji σ^2 z 12 najbliższych pikseli sąsiedztwa. Następnie dzielimy te piksele na dwie grupy, dolną o wartościach mniejszych od średniej arytmetycznej wartości pikseli i górną złożoną z pozostałych pikseli tego otoczenia. Dla obu tych grup pikseli są wyznaczane ich wariancje: dolna σ_l^2 oraz górna σ_h^2, a następnie jest sprawdzana para warunków istnienia sytuacji krawędziowej [55]:

$$\sigma^2 \geq \xi_1 \text{ oraz } \frac{\sigma^2}{0.01 + \sigma_l^2 + \sigma_h^2} \geq \xi_2, \tag{3.13}$$

gdzie wartości progowe to, odpowiednio, ξ_1 = 125 i ξ_2 = 15. Algorytm wyznaczania czwartego kontekstu nie jest wykluczający względem pozostałych trzech kontekstów, dlatego najpierw sprawdza się, czy otoczenie można zaklasyfikować do kontekstu pierwszego lub drugiego (o małej lub średniej wariancji wagowej). Jeśli nie, to sprawdza się, czy są spełnione warunki (3.13) istnienia czwartego kontekstu, w przeciwnym razie uzyskujemy trzeci kontekst.

Badanie efektywności czterokontekstowej metody, wykorzystującej statyczną predykcję liniową, zostanie zaprezentowane w podrozdziale 4.4.

3.2.3. Konteksty uzupełniające w metodzie ALCM

W przypadku metody ALCM opisanej w podrozdziale 5.3 po wyznaczeniu numeru kontekstu głównego obliczamy wartości gradientów d_h i d_v zgodnie ze wzorem (3.3). Gdy uzyskamy trzeci kontekst (o wysokiej wariancji), testujemy, czy występuje w najbliższym otoczeniu duża rozbieżność między gradientem poziomym d_h a pionowym d_v:

if (d_h > 2*d_v) kontekst = 4;
else if (d_v > 2.5*d_h) kontekst = 5;

Dodatkowy, szósty kontekst (tzw. gładki) jest wyznaczany, gdy najbliższe cztery piksele sąsiedztwa mają identyczną wartość.

3.2.4. Konteksty uzupełniające w metodzie CoBALP

W przypadku metody CoBALP opisanej w podrozdziale 5.4 po wyznaczeniu numeru kontekstu głównego obliczamy wartości gradientów d_h i d_v na podstawie wzoru (3.3). Gdy

uzyskamy trzeci kontekst (o wysokiej wariancji), testujemy, czy występuje duża rozbieżność w najbliższym otoczeniu między gradientem poziomym d_h a pionowym d_v:

if (d_h > 2*d_v) kontekst = 4;
else if (d_v > 1.5*d_h) kontekst = 5;

Dodatkowo dla obrazów o rozmiarach większych od 256×256 wykonujemy kolejne porównania, zwiększając liczbę kontekstów do siedmiu. Gdy uzyskamy drugi kontekst główny (o średniej wariancji), testujemy, czy występuje duża rozbieżność otoczenia między gradientem poziomym d_h a pionowym d_v:

if (d_h > 1.7*d_v) kontekst = 6;
else if (d_v > 1.7*d_h) kontekst = 7;

3.2.5. Konteksty uzupełniające w metodzie RLS

W przypadku metody RLS opisanej w podrozdziale 6.1 po wyznaczeniu numeru kontekstu głównego obliczamy wartości gradientów d_h i d_v na podstawie wzoru (3.3). Gdy uzyskamy trzeci kontekst (o wysokiej wariancji), testujemy, czy występuje duże odchylenie w najbliższym otoczeniu między gradientem poziomym d_h a pionowym d_v:

if (d_h > 1.6*d_v) kontekst = 4;
else if (d_v > 1.5*d_h) kontekst = 5;

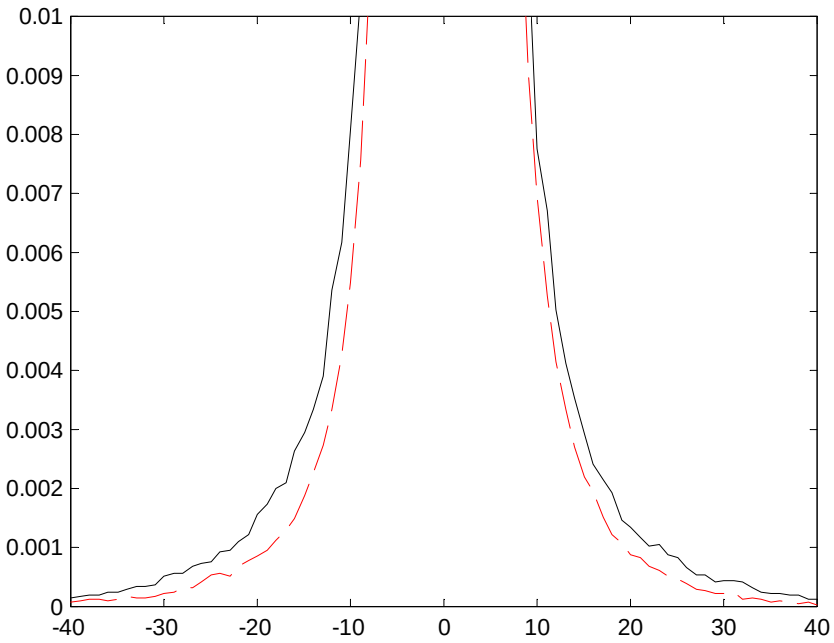
Dodatkowo dla obrazów o rozmiarach większych od 256×256 wykonujemy kolejne porównania, zwiększając liczbę kontekstów do siedmiu. Gdy uzyskamy drugi kontekst główny (o średniej wariancji), testujemy, czy występuje duże odchylenie otoczenia między gradientem poziomym d_h a pionowym d_v:

if (d_h > 1.5*d_v) kontekst = 6;
else if (d_v > 1.5*d_h) kontekst = 7;

4. Skumulowany błąd predykcji

4.1. Metody korekcji skumulowanego błędu predykcji

Metody predykcji mogą w wielu przypadkach wprowadzać w wyznaczanych błędach predykcji składowe stałe, których charakter zależy od cech danego kontekstu. Z tego powodu w wielu rozwiązaniach proponuje się wykorzystanie adaptacyjnej metody usuwania składowej stałej (ang. *bias cancelation*), zwanej także korekcją błędu predykcji skojarzonego z odpowiednim kontekstem (ang. *context-based error correction*), co pozwala uzyskać lepsze rezultaty modelowania predykcyjnego. Fragment rozkładu prawdopodobieństwa dla obrazu Lennagrey po kodowaniu z użyciem techniki modelowania MED przedstawia rys. 4.1. Linią ciągłą oznaczono rozkład powstały bez usuwania składowej stałej, a linią przerywaną rozkład uzyskany po modelowaniu z użyciem metody korekcji skumulowanego błędu predykcji. W tym drugim przypadku daje się zauważyć większe skupienie błędów predykcji wśród wartości zbliżonych do zera.



Rys. 4.1. Rozkład prawdopodobieństwa dla obrazu Lennagrey po kodowaniu różnicowym bez usuwania składowej stałej (linia ciągła) oraz z użyciem metody korekcji skumulowanego błędu predykcji (linia przerywana)

W podrozdziale 4.1 będą zaprezentowane dwie podstawowe metody korekcji skumulowanego błędu predykcji, a w podrozdziale 4.2 cztery metody podziału wielokontekstowego. Kolejne podrozdziały tego rozdziału przedstawiają przykłady wykorzystania podziałów wielokontekstowych, pokazując między innymi skuteczność metod usuwania składowej stałej.

Adaptacyjna metoda usuwania składowej stałej jest stosowana zarówno w algorytmie CALIC, jak i w JPEG-LS. Dla każdego kontekstu odnotowuje się liczbę jego wystąpień oraz skumulowaną sumę błędów i na ich podstawie koryguje się aktualnie wyznaczany błąd predykcji [141]. Różnice między metodami są nieznaczne, wersja opracowana na potrzeby JPEG-LS jest lepiej przygotowana pod względem realizacji sprzętowej, gdyż nie wymaga użycia operacji mnożenia i dzielenia. Ze względu na specyfikę metody kodowania błędów predykcji jako liczb ze znakiem autor postanowił w tej pracy dokonać pewnych modyfikacji oryginalnych rozwiązań użytych w CALIC i JPEG-LS.

Poniżej przedstawiono oba algorytmy, gdzie i oznacza numer kontekstu, $S[i]$ to aktualna wartość skumulowana błędu predykcji e w danym kontekście, $N[i]$ jest licznikiem wystąpień danego kontekstu, a $C[i]$ oznacza aktualną wartość korekcji, którą należy dodać do wartości przewidywanej jako jej wartość korygującą:

$$\dot{x} = \hat{x} + C[i]\,. \tag{4.1}$$

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

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

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

Algorytm metody wzorowanej na JPEG-LS:

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

if (abs(e) < 32) {
        S[i] = S[i] + e;
        N[i] = N[i] + 1;

        if (S[i] ≤ -N[i]) {
                C[i] = C[i] - 1;
                S[i] = S[i] + N[i];
                if (S[i] ≤ -N[i]) S[i] = -N[i] + 1;
        }
        else if (S[i] > 0) {
                C[i] = C[i] + 1;
                S[i] = S[i] - N[i];
                if (S[i] > 0) S[i] = 0;
        }
}
```

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

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

Wzór (1.3) przybiera teraz postać:

$$e = x - \dot{x} . \tag{4.2}$$

4.2. Typy podziałów wielokontekstowych

4.2.1. Pierwsza metoda

Główna idea podziału wielokontekstowego pochodzi z algorytmu CALIC [141], w którym oprócz metody wstępnej predykcji GAP (z podziałem na siedem kontekstów głównych – patrz podrozdział 3.1.2) wykorzystuje się podział na 576 kontekstów, służących do korekcji błędu predykcji skojarzonego z odpowiednim kontekstem. Każdy kontekst charakteryzuje się indywidualnymi cechami najbliższego otoczenia aktualnie kodowanego piksela, uwzględnia wzajemne zależności występujące między sąsiednimi pikselami, a często także ich poziom wariancji. W podrozdziale 4.2 zostaną zaprezentowane cztery autorskie metody podziału wielokontekstowego, które w kilku przypadkach są oparte na pomysłach znanych z literatury.

Najprostszym sposobem uzyskania podziału kontekstowego (użytym w CALIC) jest wyznaczenie średniej z l stałych predyktorów, np. dla $l = 8$ są wykorzystywane predyktory $P(1)$, $P(2)$, $P(3)$, $P(4)$, $P(5)$, $P(6)$, $GradNorth = 2P(2) - P(6)$ oraz $GradWest = 2P(1) - P(5)$. Następnie porównuje się wartość każdego z nich ze średnią z tych ośmiu predyktorów. Jeśli wartość i-tego predyktora jest większa od średniej, to znacznik bitowy κ_i ustawiany jest na 1, w przeciwnym razie na 0. Ze znaczników tworzymy l-bitową liczbę $\kappa_{l-1}... \kappa_3\kappa_2\kappa_1\kappa_0$, która stanowi numer kontekstu. W przypadku $l = 8$ mamy 256 kontekstów otoczenia. Dodatkowo można wyznaczyć miarę poziomu odchylenia od średniej, obliczając wariancję z wartości tych l predyktorów. Wartość ta może zostać skwantyzowana do L przedziałów, co łącznie daje $L \cdot 2^l$ kontekstów.

W pierwszej metodzie wykorzystano $l = 8$ powyżej opisanych predyktorów i $L = 4$ poziomy kwantyzacji wariancji. Daje to łącznie 1024 konteksty, przy czym średnią arytmetyczną z ośmiu predyktorów zastąpiono wartością przewidywaną $\hat{x}$ skojarzoną z odpowiednią metodą predykcji, w której ma być zastosowana korekcja skumulowanego błędu predykcji. Wartość wariancji (pomnożonej przez 8) możemy wyznaczyć ze wzoru:

$$\hat{\sigma}^2 = (\hat{x} - GradNorth)^2 + (\hat{x} - GradWest)^2 + \sum_{i=1}^{6} (\hat{x} - P(i))^2 . \tag{4.3}$$

Aby uzyskać podział wariancji na cztery poziomy kwantyzacji, wyznaczono eksperymentalnie trzy progi wartości $\hat{\sigma}^2$ wynoszące, odpowiednio, 400, 2500, 8000 (co odpowiada progom rów-

nym, odpowiednio, 0,18, 1,14 oraz 3,65 wartości średniej wariancji zbioru uczącego niezwiązanego ze zbiorem obrazów testowych). Podobną ideę podziału zaprezentowano w pracy [38].

4.2.2. Druga metoda

Drugi z proponowanych sposobów przypomina podział kontekstowy wykorzystany w metodzie JPEG-LS [136]. Każdą z trzech poniżej przedstawionych różnic d_1, d_2, d_3 sąsiednich pikseli dzielimy na sześć przedziałów (różnica ujemna bardzo duża, ujemna duża, ujemna mała, dodatnia mała, dodatnia duża, dodatnia bardzo duża) z użyciem wartości progowych $\{-q_2, -q_1, 0, q_1, q_2\}$, co daje łącznie $6^3 = 216$ kontekstów. Oprócz tych trzech najistotniejszych różnic wykorzystuje się różnicę d_4 z dalszego sąsiedztwa, która jest klasyfikowana dwustanowo (zapis na jednym bicie małej lub dużej różnicy bezwzględnej pikseli $P(1)$ i $P(5)$), z wykorzystaniem progu q_3. Dodatkowo bierze się pod uwagę bit znaku błędów $e(1)$ i $e(2)$. Te trzy dodatkowe bity dają $L = 8$ różnych stanów. Łącznie w drugiej metodzie uzyskujemy $8 \cdot 6^3 = 1728$ kontekstów. Poniżej przedstawiono algorytm wyznaczania numeru kontekstu:

```
d1 = P(1) – P(3);
d2 = P(2) – P(3);
d3 = P(4) – P(2);
d4 = abs(P(1) – P(5));
kontekst = 0;

if (d1<0) {
        if (d1>-q1) kontekst = 2;
        else if (d1>-q2) kontekst = 1;
}
else {
        if (d1<q1) kontekst = 3;
        else if (d1<q2) kontekst = 4;
                else kontekst = 5;
}
if (d2<0) {
        if (d2>-q1) kontekst += 2*6;
        else if (d2>-q2) kontekst += 1*6;
}
else {
        if (d2<q1) kontekst += 3*6;
        else if (d2<q2) kontekst += 4*6;
                else kontekst += 5*6;
}
if (d3<0) {
        if (d3>-q1) kontekst += 2*36;
        else if (d3>-q2) kontekst += 1*36;
}
else {
        if (d3<q1) kontekst += 3*36;
        else if (d3<q2) kontekst += 4*36;
                else kontekst += 5*36;
```

```
}
if (d4>q3) kontekst += 216;
if (e(1)>0) kontekst += 432;
if (e(2)>0) kontekst += 864;
```

Eksperymentalnie dobrane progi q_i wynoszą, odpowiednio, {5, 18, 20}.

4.2.3. Trzecia metoda

Pomysł kolejnej metody podziału kontekstowego, który zaczerpnięto z pracy [55], został częściowo oparty na uproszczonej wersji kwantyzacji wektorowej. Wstępna analiza pozwoliła określić stałą liczbę 16 wektorów (centroidów), z których każdy jest siedmioelementowym zbiorem skojarzonym z wektorem zawierającym wartości czterech błędów i trzech pikseli z najbliższego otoczenia: **V** = {*e*(1), *e*(2), *e*(3), *e*(4), *P*(1), *P*(2), *P*(4)}. Poniżej przedstawiono sposób inicjalizacji centroidów:

```
for (j=0; j<16; j++) {
        count[j] = 1;
        for (i=0; i<4; i++) centroid[j][i] = ((j>>i) & 1) * 2 - 1;
        for (i=4; i<7; i++) centroid[j][i] = (j<<4);
}
```

Wykorzystując uproszczoną metodę adaptacyjnego wyznaczania centroidów, dokonuje się porównania aktualnego wektora **V** z każdym z 16 centroidów w celu znalezienia minimalnej odległości euklidesowej. Numer *j* najlepiej pasującego centroidu jest czterobitową liczbą, która staje się częścią 10-bitowego numeru kontekstu (jest to metoda o 1024 kontekstach). Adaptacja tak wyznaczonego *j*-tego centroidu przebiega według następującej zasady:

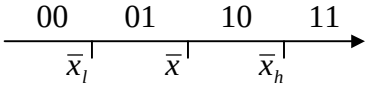
```
for (i=0; i<7; i++) {
        centroid[j][i] = (count[j] * centroid[j][i] + V[i]) / (count[j] + 1));
}
count[j]++;
```

Kolejne cztery bity decydujące o numerze kontekstu są wyznaczane jako binarna decyzja o tym, czy piksel z najbliższego sąsiedztwa (każdemu z pikseli *P*(1), *P*(2), *P*(3), *P*(4) odpowiada jeden bit numeru kontekstu) ma wartość zbliżoną do wartości przewidywanej $\hat{x}$ (odpowiedni bit kontekstu równy 0), czy też różni się od niej o wartość przekraczającą pewien próg (eksperymentalnie ustalony na 7, czyli gdy $|\hat{x} - P(i)| \geq 7$). Wtedy odpowiedni bit kontekstu jest ustawiany na 1.

Ostatnie dwa bity, służące do wyznaczenia liczby będącej numerem kontekstu, otrzymujemy, porównując $\hat{x}$ z wartościami dwóch najbliższych sąsiadów *P*(1) i *P*(2). Jeśli $P(i) < \hat{x}$, to bit ma wartość 0, w przeciwnym razie wartość 1. W ten sposób uzyskujemy 10 bitów tworzących numer kontekstu.

4.2.4. Czwarta metoda

Wnikliwa analiza najbliższego sąsiedztwa prowadzi do kolejnego ujęcia klasyfikacji kontekstów. Podobnie jak w trzeciej metodzie i w tej sytuacji rozpatruje się wyłącznie cztery najbliższe piksele: $P(1)$, $P(2)$, $P(3)$, $P(4)$ i wyznacza 10-bitowy numer kontekstu. Najpierw oblicza się ich średnią arytmetyczną $\bar{x}$, później piksele te są dzielone na dwie grupy – pierwszą o wartościach znajdujących się poniżej średniej oraz drugą składającą się z pozostałych pikseli. Dla każdej z tych grup wyznacza się ich własne średnie $\bar{x}_l$ oraz $\bar{x}_h$. Następnie dokonuje się kwantyzacji wartości każdego z czterech pikseli $P(i)$ przez skojarzenie go z jednym z czterech przedziałów (wykorzystując trzy progi kwantyzacji: średnią arytmetyczną wszystkich pikseli $\bar{x}$ oraz średnie $\bar{x}_l$ i $\bar{x}_h$). Tak zaprojektowany kwantyzator czterostanowy, który przedstawiono na rys. 4.2, zwraca dwubitową liczbę dla każdego z czterech sąsiadów $P(i)$, co łącznie daje osiem bitów stanowiących część numeru kontekstu.



Rys. 4.2. Czterostanowy kwantyzator skalarny

Dodatkowe dwa bity uzupełniające 10-bitowy numer kontekstu wyznacza się, korzystając z czterostanowego kwantyzatora dzielącego zakres różnicy $\bar{x}_h - \bar{x}_l$ na cztery przedziały z eksperymentalnie dobranymi progami {4, 12, 30}. Wartość tej różnicy odwzorowuje pośrednio poziom wariancji analizowanych pikseli $P(1)$, $P(2)$, $P(3)$, $P(4)$.

4.3. Mieszana metoda korekcji skumulowanego błędu predykcji

W autorskim rozwiązaniu proponowanym w tej pracy zastosowano średnią ważoną obu metod korekcji (patrz podrozdział 4.1), które skojarzono z każdym z czterech sposobów wyznaczania kontekstu. Łącznie wykorzystuje się zatem średnią ważoną z ośmiu wartości korekcji wyznaczanych dla wartości przewidywanej $\hat{x}$. W przeprowadzonych eksperymentach dobrano stałe wagi, lecz istnieje możliwość indywidualnego ich doboru dla każdego z kodowanych obrazów, co wiąże się jednak ze znacznym wzrostem złożoności obliczeniowej kodera. Jeśli konteksty poszczególnych czterech metod omówionych w podrozdziale 4.2 oznaczymy jako $\{k_1, k_2, k_3, k_4\}$, poszczególne wartości korekcji wyliczane metodą CALIC jako $C_C[k_i]$, a wartości korekcji wyznaczane metodą JPEG-LS jako $C_J[k_i]$, to ostateczną wartość korekcji mieszanej C_{mix} ustala się jako średnią ważoną:

$$C_{\text{mix}} = 0{,}2C_C[k_1] + 0{,}15(C_C[k_2] + C_C[k_3] + C_J[k_1] + C_J[k_3]) + \\ + 0{,}1C_C[k_4] + 0{,}05(C_J[k_2] + C_J[k_4]). \tag{4.4}$$

Wówczas, po uwzględnieniu autorskiej mieszanej metody korekcji, otrzymujemy wartość przewidywaną o postaci:

$$\dot{x} = \hat{x} + C_{\text{mix}} . \tag{4.5}$$

Wysoka skuteczność tej metody wynika z zasad mieszania wartości korekcji obciążonych pewnym poziomem niepewności, które pojedynczo mogą w niektórych sytuacjach pogarszać ostateczny poziom błędu predykcji (chybione przewidywanie wartości korekcji). Zapobiega temu mieszanie (średnia ważona) powodujące w dużym stopniu znoszenie się owych niepewnych wartości korekcji. Jednocześnie metoda ta zmniejsza szanse na wystąpienie efektu niesymetryczności rozkładu, który może objawiać się pojawieniem w danym kontekście *i* rozbieżności między pozycją w histogramie, wskazywaną jako średnia arytmetyczna *C*[*i*] błędów predykcji, a faktyczną pozycją wskazującą w histogramie maksymalne prawdopodobieństwo (histogram rozkładu Laplace'a ma jedno maksimum), co zasygnalizowano w pracy [114]. Badania skuteczności tej metody korekcji wykazują, że średnio aż w 70% przypadków zastosowanie korekcji daje wartość zmodyfikowanego błędu predykcji mniejszą (lepszą) od wartości błędu predykcji bez użycia korekcji lub równą tej wartości.

Warto zauważyć, że część metod omówionych w podrozdziale 4.2 wykorzystuje do wyznaczania numeru kontekstu wartość przewidywaną kodowanego piksela. Im dokładniejsza metoda przewidywania, tym celniejsze wydaje się dopasowanie do aktualnego kontekstu. Po skorygowaniu wartości przewidywanej o wartość składowej stałej otrzymujemy kolejną wartość przewidywaną, co może posłużyć do ponownego, jeszcze precyzyjniejszego wyznaczenia kontekstu. Można to określić mianem kaskadowego wyznaczania składowej stałej. Taki trzystopniowy proces wykorzystano w pracy [110]. Podobna idea zostanie użyta (w połączeniu z omówioną tu techniką mieszanej korekcji skumulowanego błędu predykcji) w metodzie Blend-24 (patrz podrozdział 9.8), w której również występuje trzystopniowy proces wyznaczania ostatecznej wartości przewidywanej.

4.4. Metoda statyczna z podziałem kontekstowym

Metodę statyczną z podziałem kontekstowym opisano w tym miejscu, aby dokładniej zobrazować możliwości wzrostu efektywności dzięki wykorzystaniu podziału kontekstowego oraz korekcji skumulowanego błędu predykcji. W podrozdziałach 3.2.1 i 3.2.2 zaprezentowano sposób wyznaczania jednego z trzech kontekstów głównych i jednego uzupełniającego. W metodzie statycznej dla każdego z nich oblicza się metodą MMSE indywidualny zestaw współczynników predykcji, dobierając rzędy poszczególnych predyktorów w zależności od rozmiarów obrazu: przy rozdzielczości 256×256 r = 29, przy rozdzielczości 512×512 r = 46, a przy rozdzielczości 720×576 r = 48. Dla wszystkich wielkości obrazów rząd predykcji w czwartym kontekście ustalono na r = 4.

W tabeli 4.1 zaprezentowano wyniki pomiarów entropii rzędu zerowego i średniej bitowej dla poszczególnych ustawień kodera (badanie dla 45 standardowych obrazów testowych), opublikowane w pracy [122]. Na początku są podane wartości dla kodera korzystającego z jednego oraz trzech predyktorów statycznych (zgodnie z opisem zawartym w podrozdziale 3.2.1).

Tab. 4.1. Wyniki entropii i średnich bitowych dla 45 standardowych obrazów testowych

Metoda	Entropia	Średnia bitowa
1 predyktor	4,42601	4,16603
3 predyktory	4,33600	4,08275
3 predyktory z metodą korekcji nr 1	4,27731	4,05319
3 predyktory z metodą korekcji nr 2	4,31012	4,07319
3 predyktory z metodą korekcji nr 3	4,29894	4,06354
3 predyktory z metodą korekcji nr 4	4,31376	4,07617
3 predyktory z mieszaną metodą korekcji	4,25925	4,03033
4 predyktory z mieszaną metodą korekcji	4,25443	4,02778

Kolejne badania przeprowadzono, aktywując pojedynczo metody korekcji (z adaptacją według algorytmu CALIC) oparte na poszczególnych czterech metodach wyznaczania kontekstów opisanych w podrozdziale 4.2. Dwa końcowe pomiary wykonano z aktywną opcją mieszanej korekcji, zaproponowaną w podrozdziale 4.3, przy czym w drugim przypadku aktywna była także detekcja kontekstu krawędziowego (czwarty predyktor statyczny). Wyniki zawarte w tej tabeli ukazują poziom istotności zarówno wprowadzenia podziału na konteksty główne, jak i użycia mieszanej metody korekcji skumulowanego błędu predykcji.

Wyniki prezentowane w kolejnych rozdziałach będą uwzględniały użycie korekcji mieszanej, a także rozbudowaną (w porównaniu z omówioną w pracy [122]) wersję kodera arytmetycznego, którą zaprezentowano w rozdziale 8. Dzięki zwiększeniu efektywności tego kodera uzyskano spadek średniej bitowej z 4,02778 do 4,01522. Tabela 4.2 przedstawia porównanie podstawowej statycznej metody predykcji (z jednym zestawem współczynników predykcji) z metodą czterokontekstową.

Tab. 4.2. Wyniki pomiarów dla 45 obrazów testowych – statyczne metody podstawowa i czterokontekstowa

Obrazy	Entropia (1 kontekst)	Średnia (1 kontekst)	Entropia (4 konteksty)	Średnia (4 konteksty)
Aerial	5,03012	4,62645	5,01352	4,61207
Airfield	5,02493	4,86789	5,00198	4,84984
Airplane	3,99270	3,65373	3,94208	3,60678
Baboon	6,07428	5,78803	6,05775	5,77475
Barbara	4,78822	4,42192	4,71414	4,35007
Boat	4,68213	4,50143	4,64393	4,46870
Bridge	3,67609	3,40238	3,67319	3,40294
Couple	4,33727	4,13578	4,32223	4,12689
Crowd	4,09447	3,65221	4,07617	3,62647
Elaine	4,53371	4,38324	4,50049	4,35730
Finger	5,18094	5,17865	5,17882	5,18034
Frog	5,09806	5,02217	5,07509	5,00950
Goldhill	4,72074	4,52411	4,70529	4,51170
Harbour	4,90377	4,35149	4,88452	4,33705
Lax	5,76931	5,57224	5,75009	5,55564
Lennagrey	4,23992	4,02362	4,20267	3,99126

Tab. 4.2. Wyniki pomiarów dla 45 obrazów testowych – statyczne metody podstawowa i czterokontekstowa (cd.)

Obrazy	Entropia (1 kontekst)	Średnia (1 kontekst)	Entropia (4 konteksty)	Średnia (4 konteksty)
$Lena_{TMW}$	4,59995	4,41089	4,55912	4,37727
Man	4,64974	4,32373	4,63773	4,31444
Peppers	4,48412	4,30183	4,43669	4,26812
Sailboat	4,78699	4,48114	4,75293	4,44104
Seismic	2,11172	2,10477	2,10825	2,11259
Shapes	2,14057	1,65710	1,87919	1,46812
Tank	3,95541	3,79158	3,94686	3,78834
Truck	4,22868	4,03247	4,22237	4,03085
Woman1	4,19594	3,92957	4,16843	3,91056
Woman2	3,29604	3,04764	3,27820	3,04036
Balloon	2,84380	2,73730	2,77498	2,68496
Barb	4,53708	4,23651	4,45642	4,14692
Barb2	4,80449	4,44634	4,73136	4,36433
Board	3,62093	3,44942	3,54365	3,38744
Boats	3,97261	3,70549	3,90834	3,64393
Girl	3,77064	3,62801	3,71902	3,58097
Gold	4,55985	4,30204	4,54148	4,28340
Hotel	4,46672	4,19057	4,40877	4,13583
Zelda	3,72667	3,63188	3,65790	3,57529
Bridge256	5,81932	5,61526	5,80913	5,61925
Camera	4,68925	4,19048	4,63307	4,15546
Couple256	3,93645	3,55283	3,92203	3,54767
Earth	3,70365	2,86818	3,69186	2,86913
Elif	2,89840	2,74648	2,85368	2,70694
Noisesquare	5,22308	5,23497	5,20979	5,23996
Omaha	6,35053	6,10573	6,33711	6,09644
Sena	3,12075	2,95294	3,08683	2,92368
Sensin	3,41364	3,27753	3,36460	3,23108
Sinan	3,07800	2,99599	3,05357	2,97920
Średnia	4,29182	4,04564	4,25412	4,01522

Wykorzystanie wyznaczania modeli predykcyjnych metodą MMSE dla większej liczby kontekstów zaprezentowano w pracy [128], a następnie pomysłu tego użyto także w badaniach [113]. Jednak uzyskane rezultaty nie były konkurencyjne do opisanej tu metody czterokontekstowej, dlatego nie prowadzono dalszych badań z użyciem dużej liczby kontekstów.

4.5. Dobór współczynników predykcji metodą minimalizacji średniej bitowej

W podrozdziałach 2.4 oraz 2.5 zaprezentowano różne autorskie metody doboru współczynników predykcji, uzasadniając, iż metoda MMSE stosowana do wyznaczania współczynników predykcji nie gwarantuje najniższej wartości entropii. Zastąpienie MMSE metodą

poszukiwania współczynników predykcji wiąże się z dużą częstością wyznaczania entropii, co ogranicza praktyczne jej zastosowanie jedynie do predyktorów niewielkiego rzędu. W tym podrozdziale zostanie przedstawiona autorska propozycja o małej liczbie działań wymagających wyznaczania entropii, możliwa do wykorzystania dla dużych wartości *r*. Dzięki temu można dokonywać pomiarów nie tylko entropii, lecz pełnego pomiaru średniej bitowej, uwzględniając zarówno użycie mieszanej metody korekcji skumulowanego błędu predykcji, jak i kodera arytmetycznego bez obawy o zbyt długi czas kodowania.

Metoda wyznaczania czterech zestawów współczynników predykcji, przedstawiona w podrozdziale 4.4, może zostać wykorzystana także do obliczania jednego, wspólnego dla całego obrazu zestawu współczynników statycznej predykcji liniowej. Badania wykazały, iż lepsze rezultaty można uzyskać, ignorując dane o pikselach, których sąsiedztwo zaklasyfikowano jako kontekst krawędziowy (czwarty), dlatego wartości te nie są uwzględniane przy obliczaniu macierzy **R**. Dla pozostałych pikseli przynależących do jednego z trzech pierwszych kontekstów wyznacza się metodą MMSE zestawy współczynników predykcji $\mathbf{B}_{\text{stat1}}$, $\mathbf{B}_{\text{stat2}}$ oraz $\mathbf{B}_{\text{stat3}}$, odpowiadające poszczególnym klasom, zgodnie z zasadą opisaną w podrozdziale 4.4. Mając dane N_{stat1}, N_{stat2} oraz N_{stat3} o liczebności każdego kontekstu, możemy wyznaczyć nowy zestaw $\mathbf{B}_{\text{stat}}$ współczynników predykcji jako kombinację liniową obliczonych uprzednio trzech modeli predykcyjnych:

$$\mathbf{B}_{\text{stat}} = \frac{N_{\text{stat1}} \cdot \mathbf{B}_{\text{stat1}} + N_{\text{stat2}} \cdot \mathbf{B}_{\text{stat2}} + N_{\text{stat3}} \cdot \mathbf{B}_{\text{stat3}}}{N_{\text{stat1}} + N_{\text{stat2}} + N_{\text{stat3}}}. \tag{4.6}$$

Metoda ta uzyskuje lepsze dopasowanie do każdego z trzech rejonów o różnym poziomie wariancji otoczenia, a wagi każdego modelu predykcyjnego są proporcjonalne do liczby pikseli przynależnych do odpowiadających im kontekstów.

Kontekst krawędziowy (czwarty) jest na ogół dość mało liczny (poniżej 1% pikseli całego obrazu), z tego powodu model utworzony dla tej grupy pikseli nie jest brany pod uwagę (wyniki eksperymentalne potwierdzają słuszność pomijania tego czwartego modelu we wzorze (4.6)). W tabeli 4.3 znajdują się wyniki entropii i średniej bitowej dla obrazów zakodowanych statyczną metodą z jednym predyktorem $\mathbf{B}_{\text{stat}}$ (patrz lewa strona tabeli – pierwsza metoda). Porównując dane podane w tab. 4.2, można zauważyć, iż średnia wartość entropii z 45 pomiarów wynosi 4,30081 i jest wyższa o 0,00899 bitu od średniej uzyskanej klasyczną (bezkontekstową) metodą MMSE.

Wzrosła również średnia wartość MSE z 7,15142 do 7,39901 (wzrost o 3,46%), a także średnia wartość MAE z 4,58732 do 4,65254 (wzrost o 1,42%). Jednak wartość średniej bitowej zmniejszyła się z 4,04564 do 4,03637, co pokazuje, że nawet kryterium minimalizacji MSE, MAE czy entropii nie gwarantuje uzyskania najlepszej średniej bitowej.

Tab. 4.3. Wyniki pomiarów dla 45 obrazów testowych – statyczne metody z minimalizacją średniej bitowej

Obrazy	Entropia (metoda I)	Średnia (metoda I)	Entropia (metoda II)	Średnia (metoda II)
Aerial	5,02686	4,61665	5,02686	4,61665
Airfield	5,03917	4,86601	5,03264	4,86252
Airplane	4,00944	3,63971	3,99487	3,63592
Baboon	6,07510	5,78297	6,07510	5,78297
Barbara	4,84901	4,42480	4,81387	4,41168
Boat	4,71383	4,50477	4,69529	4,49561
Bridge	3,67426	3,40106	3,67206	3,39871
Couple	4,35257	4,13955	4,34313	4,13021
Crowd	4,08445	3,63261	4,07954	3,63032
Elaine	4,53438	4,37470	4,52897	4,37173
Finger	5,18178	5,17920	5,18094	5,17865
Frog	5,10355	5,01869	5,10355	5,01869
Goldhill	4,72652	4,52216	4,72082	4,51691
Harbour	4,91087	4,35290	4,90468	4,35074
Lax	5,78078	5,57360	5,76931	5,57224
Lennagrey	4,26407	4,01559	4,23992	4,00865
$Lena_{TMW}$	4,62407	4,40456	4,60511	4,39562
Man	4,65501	4,32155	4,64551	4,31594
Peppers	4,48842	4,28761	4,47807	4,28444
Sailboat	4,81557	4,48001	4,80264	4,46976
Seismic	2,11612	2,11220	2,11386	2,10385
Shapes	1,98377	1,52494	1,70398	1,33807
Tank	3,95659	3,78708	3,95659	3,78708
Truck	4,22791	4,02769	4,22791	4,02769
Woman1	4,20438	3,91897	4,19547	3,91770
Woman2	3,30258	3,04460	3,29339	3,04352
Balloon	2,86515	2,71686	2,82598	2,71474
Barb	4,58645	4,22665	4,54715	4,20560
Barb2	4,83096	4,43707	4,80671	4,43281
Board	3,68301	3,45414	3,62357	3,43382
Boats	4,03191	3,71216	3,99024	3,69386
Girl	3,78424	3,61936	3,76207	3,60684
Gold	4,56197	4,29457	4,55715	4,29194
Hotel	4,50718	4,19236	4,47801	4,17951
Zelda	3,72297	3,60016	3,70805	3,59943
Bridge256	5,82453	5,62019	5,81932	5,61526
Camera	4,71263	4,20174	4,68925	4,19048
Couple256	3,94856	3,55144	3,93112	3,54703
Earth	3,70794	2,86984	3,69593	2,86722
Elif	2,91143	2,71481	2,91143	2,71481
Noisesquare	5,22279	5,23143	5,22279	5,23143
Omaha	6,36039	6,10931	6,35053	6,10573
Sena	3,14894	2,93738	3,14894	2,93738
Sensin	3,33557	3,20299	3,33557	3,20299
Sinan	3,08899	2,98997	3,08899	2,98997
Średnia	4,30081	4,03637	4,28215	4,02726

Możemy rozwinąć zaprezentowaną powyżej propozycję celem dalszego zmniejszenia średniej bitowej. Po wyznaczeniu wektora $\mathbf{B}_{stat}$ zgodnie ze wzorem (4.6) obliczamy wartości błędów predykcji *e* dla każdego piksela, po czym dokonujemy reklasyfikacji trzech kontekstów głównych (nadal omijając piksele skojarzone z kontekstem krawędziowym) według następującej zasady:

if ($e > T_{stat}$) kontekst = 1;
else if ($e < -T_{stat}$) kontekst = 2;
else kontekst = 3;

Podział polega zatem na kwantyzacji wartości błędu predykcji *e* na trzy klasy: z małą, dużą dodatnią i dużą ujemną wartością *e*. Eksperymentalnie próg kwantyzacji T_{stat} ustalono na 11, choć dla każdego obrazu można wyznaczyć indywidualną wartość tego parametru. Proces reklasyfikacji wykonujemy dwukrotnie, przy czym po każdej iteracji wyznaczamy ponownie $\mathbf{B}_{stat1}$, $\mathbf{B}_{stat2}$, $\mathbf{B}_{stat3}$ oraz zgodnie ze wzorem (4.6) nowy zestaw współczynników predykcji $\mathbf{B}_{stat}$, dla którego jest mierzona wartość średniej bitowej. Dodatkowo dokonujemy czwartego pomiaru średniej bitowej, wykorzystując klasyczną (bezkontekstową) metodę MMSE i na podstawie wszystkich pomiarów wybieramy ostateczny zestaw współczynników $\mathbf{B}_{stat}$, dla którego uzyskano najmniejszą średnią bitową (patrz prawa strona tab. 4.3 – druga metoda). Przy pomiarach wykorzystano rzędy predykcji zgodnie z opisem zawartym w podrozdziale 4.4. Również i w tym przypadku nastąpił wzrost średniej wartości MSE (o 1,93%) i średniej wartości MAE (o 0,31%) w porównaniu z klasyczną metodą MMSE, natomiast średnia bitowa zmniejszyła się o 0,01838 do wartości 4,02726 bitu. W porównaniu z metodami proponowanymi w podrozdziale 2.4 zaletą tej propozycji jest wysoka efektywność przy krótkim czasie wyznaczenia współczynników predykcji (wraz z kodowaniem jest to zaledwie kilkanaście sekund).

5. Metody prostej adaptacji modelu predykcji

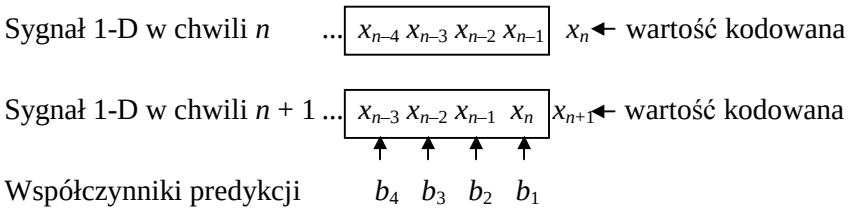
5.1. Wprowadzenie

Oprócz stałych i statycznych predyktorów liniowych istnieją także ich adaptacyjne odmiany. W podrozdziale 2.6 przedstawiono ideę zastosowania statycznej predykcji liniowej z podziałem blokowym, którą można uznać za metodę z adaptacją w przód. W tym i w kolejnym rozdziale zostaną omówione podstawowe metody z adaptacją wstecz, czyli oparte na pikselach dostępnych zarówno koderowi, jak i dekoderowi.

Główną zaletą adaptacji współczynników predykcji jest możliwość dostosowywania się współczynników b_j do zmienności sygnału w czasie. Drugą zaletą, charakterystyczną jedynie dla adaptacji wstecz, jest brak potrzeby wstępnej analizy całego sygnału, która jest wymagana przy użyciu modeli statycznych. W analizie tej np. w przypadku MMSE należy wyznaczyć i rozwiązać jeden lub wiele układów równań z r niewiadomymi. Podstawy teoretyczne dotyczące adaptacji metod opisanych zarówno w rozdziale 5, jak i 6 szczegółowo omówiono w pracy [153].

W tym rozdziale zostaną omówione trzy metody szybkiej adaptacji wstecz współczynników predykcji liniowej. We wszystkich metodach adaptacja jest wykonywana po zakodowaniu każdego kolejnego piksela, co sprawia, że istotną sprawą jest znalezienie efektywnych metod adaptacji przy zachowaniu jak najmniejszej złożoności obliczeniowej.

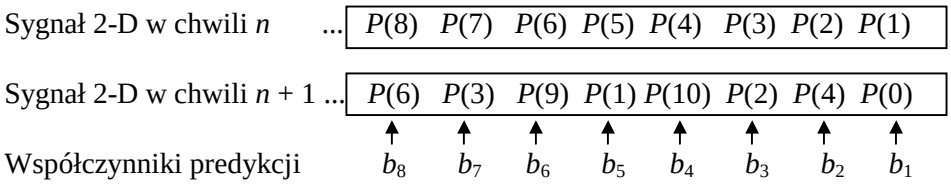
Podstawowe metody dokonują adaptacji w sposób iteracyjny. W takim podejściu zmiana między dwiema kolejnymi iteracjami pozwala najczęściej na wykorzystanie okna przesuwnego w odniesieniu do r sąsiednich (kolejno kodowanych/dekodowanych) próbek sygnału. Dwie kolejne chwile dla sygnału jednowymiarowego (1-D) przedstawiono na rys. 5.1. W obramowaniu znajdują się próbki x_{n-j} sąsiedztwa dla rzędu predykcji $r = 4$, a pod spodem odpowiadające im współczynniki predykcji.



Rys. 5.1. Interpretacja otoczenia wartości kodowanej dla sygnału jednowymiarowego w dwóch kolejnych chwilach

Dwuwymiarowość danych obrazu jest dużym problemem w każdej z metod adaptacyjnych. Przy rastrowym skanowaniu obrazu prowadzi to do sytuacji, w której aktualne wartości współczynników predykcji mogą być lepiej dostosowane do cech otoczenia piksela znajdującego się na lewo o 100 pozycji niż do piksela znajdującego się np. pięć wierszy powyżej aktualnie kodowanego $P(0)$. Drugim problemem jest sposób odwoływania się do sąsiedztwa

przedstawiony na rys. 1.1, niepozwalający na wykorzystanie zalet okna przesuwnego, co jest charakterystyczne tylko dla sygnału 1-D. Aby wyjaśnić zalety okna przesuwnego, wystarczy porównać rys. 5.1 z rys. 5.2. Sposób odwołań do sąsiedztwa sygnału dwuwymiarowego (2-D) zaprezentowano na rys. 5.2, na którym przedstawiono okna dla dwóch kolejnych chwil (n oraz $n + 1$), zawierające numery pikseli zgodne z rys. 1.1 i odpowiadające im współczynniki predykcji b_j dla rzędu $r = 8$. Widać tu w chwili ($n + 1$) duże nieuporządkowanie kolejności występujących obok siebie pikseli $P(j)$.



Rys. 5.2. Interpretacja otoczenia wartości kodowanej dla sygnału dwuwymiarowego w dwóch kolejnych chwilach

5.2. Podstawowa metoda adaptacji – LMS

Najpowszechniejszą z metod adaptacji współczynników predykcji jest LMS (ang. *Least Mean Square*). Należy ona do najprostszych, lecz jednocześnie najmniej efektywnych, gdyż w praktyce dla obrazów zbyt wolno jest zbieżna do uzyskania rezultatów otrzymywanych w technice MMSE z adaptacją w przód. Adaptacja wstecz współczynników predykcji w metodzie LMS z ogranicznikiem zapobiegającym wpadaniu w oscylacje odbywa się według następującej zasady:

$$b_j(n+1) = b_j(n) + \mu \cdot \bar{e} \cdot P(j), \tag{5.1}$$

gdzie:

$$\bar{e} = \operatorname{sgn}(e) \cdot \min\{|e|, \varphi\} \tag{5.2}$$

jest wartością błędu predykcji z ogranicznikiem φ.

Kluczową rolę odgrywa tu współczynnik uczenia μ, od którego zależy szybkość adaptacji współczynników b_j do aktualnych cech sygnału. W przypadku obrazu skanowanie rastrowe rozważane w tej pracy może powodować szybkie zmiany cech sygnału. Choć sytuacja podobnej zmienności powtarza się w każdym kolejnym wierszu obrazu, to w celu zachowania stabilności predyktora (należy zabezpieczyć się przed wpadaniem w oscylacje) wartość μ powinna być mała, często przyjmuje się ją na poziomie od 2^{-23} do 2^{-22} [16]. Odpowiedni dobór tej wartości ma duży wpływ na końcową średnią bitową kodowanego obrazu. Zakładając niską złożoność projektowanego kodera LMS, nie można pozwolić na wielokrotne kodowanie obrazu celem odpowiedniego doboru parametru μ. Problemem jest zatem ustalenie zarówno uniwersalnej wartości współczynnika uczenia, jak i rzędu predykcji.

Wyciągając wnioski z własnych badań i opierając się na literaturze związanej z bezstratnym kodowaniem obrazów, w której wykorzystano metodę LMS [25, 107], udało się w tej pracy zaprezentować usprawnioną metodę adaptacji. Istotną rolę odgrywa tu stan początkowy, czyli sposób inicjalizacji wektora współczynników predykcji.

Na podstawie analiz można zaproponować predyktor rzędu $r = 6$, o współczynnikach **B** = [0,62, 0,625, –0,125, 0,125, –0,125, –0,125]. W przypadku użycia wyższych wartości r, każdy kolejny współczynnik ma wyzerowaną wartość początkową. W badaniach stosowano rząd predykcji $r = 24$ oraz ogranicznik $\varphi = 8$. Współczynnik szybkości adaptacji został uzmienniony w następujący sposób:

$$\mu_j = \frac{0{,}088}{\vartheta_j(n) + 400}, \tag{5.3}$$

przy czym wartość $\vartheta_j(n)$ jest aktualizowana zgodnie z poniższą zależnością [29]:

$$\vartheta_j(n+1) = \gamma \cdot \vartheta_j(n) + (1-\gamma) \cdot P^2(j), \tag{5.4}$$

gdzie wartość γ jest współczynnikiem zapominania ustawionym na 0,95. Początkowe wartości: $\vartheta_j(0) = 0$, dla każdego j od 1 do r. Jest to iteracyjna wersja metody NLMS przystosowanej do sygnałów jednowymiarowych. Z tego względu nawet zaproponowane modyfikacje nie pozwalają uzyskać zbyt dobrych rezultatów dla sygnałów 2-D. Choć otrzymane wartości są nieco lepsze niż w przypadku kodera CALIC, to wynika to w dużym stopniu z zastosowania wydajnych metod korekcji błędu predykcji oraz adaptacyjnego kodera arytmetycznego.

W tabeli 5.1 przedstawiono porównanie entropii oraz średnich bitowych (uzyskanych z użyciem kodera arytmetycznego opisanego w rozdziale 8) dla trzech metod predykcyjnych. Pierwsza z nich wykorzystuje opisany w tym podrozdziale stały predyktor szóstego rzędu, stosowany do inicjalizacji współczynników w metodzie LMS. Drugi model predykcyjny, GAP_+, został zaczerpnięty z pracy [132]. Jest to udoskonalona wersja nieliniowego predyktora GAP używanego w algorytmie CALIC (szczegóły zaprezentowano w podrozdziale 3.1.2). Ostatnim modelem jest udoskonalona metoda LMS opisana w tym rozdziale.

Istnieje też możliwość zastosowania LMS, jeśli dane potraktuje się jako dwuwymiarowe [75]. Wymaga to jednak metody dostrajania współczynników (po każdym zakodowanym pikselu) na podstawie obszaru uczącego, znajdującego się w najbliższym sąsiedztwie, i to w sposób iteracyjny. Zapewnia to dobre dostosowanie się predyktora do lokalnych cech obrazu, lecz wiąże się z dużym wzrostem złożoności obliczeniowej (nawet kilkaset operacji adaptacji niezbędnych do zakodowania/zdekodowania jednego piksela). Ponadto wyniki zaprezentowane w pracy [75] są bardzo zbliżone do uzyskanych metodą LMS przedstawioną w tym podrozdziale.

Tab. 5.1. Wyniki pomiarów dla 45 obrazów testowych – metoda ze stałym predyktorem ($r = 6$), metoda GAP_+ oraz LMS

Obrazy	Entropia stały pred.	Średnia stały pred.	Entropia GAP_+	Średnia GAP_+	Entropia LMS	Średnia LMS
Aerial	5,09527	4,69309	5,09764	4,70139	5,06019	4,67131
Airfield	5,09220	4,93405	5,09002	4,93266	5,05892	4,90318
Airplane	4,04486	3,66544	4,02611	3,66076	4,01665	3,65675
Baboon	6,12482	5,82512	6,13406	5,83980	6,06936	5,79326
Barbara	5,06820	4,56168	5,06812	4,57981	4,81873	4,42936
Boat	4,85320	4,62828	4,81307	4,60501	4,73652	4,53410
Bridge	3,69380	3,43323	3,68348	3,41808	3,68383	3,41594
Couple	4,39845	4,18562	4,33410	4,14396	4,35219	4,14808
Crowd	4,17296	3,69046	4,16877	3,70566	4,13734	3,69401
Elaine	4,89213	4,72359	4,89146	4,72439	4,70716	4,55872
Finger	5,34932	5,29675	5,45609	5,40688	5,23502	5,22771
Frog	5,13202	5,05743	5,14289	5,06289	5,11218	5,03686
Goldhill	4,79535	4,58159	4,78012	4,57509	4,74830	4,54361
Harbour	5,00913	4,44659	4,89118	4,36786	4,89191	4,36259
Lax	5,80605	5,60580	5,80656	5,60647	5,77848	5,58369
Lennagrey	4,27655	4,03939	4,26725	4,03303	4,26426	4,03656
$Lena_{TMW}$	4,62894	4,42392	4,62220	4,41877	4,61985	4,42319
Man	4,66435	4,34086	4,63495	4,31748	4,66724	4,34747
Peppers	4,57634	4,38826	4,53107	4,34945	4,51229	4,32602
Sailboat	4,92000	4,62918	4,90884	4,61355	4,85398	4,56811
Seismic	2,83159	2,70126	2,83183	2,73737	2,50796	2,43831
Shapes	1,98425	1,51937	1,72802	1,37669	1,93593	1,54893
Tank	3,98461	3,82049	3,96355	3,80701	3,97615	3,81277
Truck	4,25511	4,06622	4,26365	4,07219	4,24527	4,05632
Woman1	4,22669	3,95134	4,18075	3,92339	4,19139	3,93123
Woman2	3,33965	3,08049	3,36172	3,09330	3,32833	3,06709
Balloon	2,90693	2,77103	2,87982	2,76134	2,88778	2,76766
Barb	4,83848	4,34765	4,76423	4,32652	4,54884	4,21111
Barb2	4,90841	4,48401	4,85390	4,45889	4,81456	4,44245
Board	3,78817	3,55702	3,68146	3,49265	3,65337	3,45577
Boats	4,15108	3,80020	4,06908	3,75490	4,02567	3,72290
Girl	3,90981	3,72256	3,87662	3,70830	3,82787	3,66602
Gold	4,62630	4,34849	4,60963	4,34009	4,58174	4,31549
Hotel	4,54846	4,25023	4,50220	4,21403	4,48957	4,20217
Zelda	3,84692	3,72054	3,80928	3,69696	3,75192	3,64189
Bridge256	5,86484	5,67424	5,83822	5,64427	5,82050	5,62912
Camera	4,69584	4,21588	4,60173	4,14163	4,67040	4,18649
Couple256	3,98521	3,57669	3,91304	3,53334	3,96525	3,58479
Earth	3,69051	2,87404	3,70133	2,87286	3,70681	2,87570
Elif	3,16778	2,91302	3,16020	2,92062	3,12652	2,89494
Noisesquare	5,37756	5,37311	5,37662	5,37584	5,29600	5,31194
Omaha	6,33825	6,09454	6,37080	6,14447	6,34207	6,11322
Sena	3,41894	3,11467	3,41609	3,12593	3,35396	3,08517
Sensin	3,63629	3,40344	3,66282	3,44241	3,58900	3,38808
Sinan	3,36040	3,15071	3,40613	3,18225	3,34999	3,15248
Średnia	4,40613	4,12626	4,38157	4,11578	4,34025	4,08361

Podział na zestaw kilku niezależnych predyktorów LMS z zastosowaniem przełączania kontekstów głównych (patrz podrozdział 3.2.1) nie prowadzi do wzrostu efektywności metody LMS. Aby otrzymać lepszą efektywność przy zachowaniu niskiej złożoności, należy wykorzystać jedną z kilku metod lepiej dostosowanych do sygnałów 2-D, co zostanie opisane w kolejnych podrozdziałach. Istotnym zastosowaniem metody LMS jest jej użycie jako dodatkowego etapu dalszego modelowania błędów predykcji, co zostanie szerzej zaprezentowane w podrozdziale 6.5.

5.3. Metoda adaptacji ALCM i jej udoskonalenie

Wśród metod o niskiej złożoności obliczeniowej, zaproponowanych w odpowiedzi na ogłoszone w latach 90. XX wieku wezwanie organizacji JPEG, znalazła się adaptacyjna metoda predykcji ALCM (ang. *Activity Level Classification Model*) [99]. Ma ona niższą, choć zbliżoną złożoność obliczeniową w porównaniu z LMS. Oryginalna metoda operowała na modelach predykcji liniowej piątego lub szóstego rzędu. Spośród tych kilku współczynników w każdej iteracji adaptacji ulegały (o wartość stałą μ = 1/256) tylko wybrane dwa z nich. Mimo swej prostoty metoda dawała dobre rezultaty, z tego też względu w tej pracy zaproponowano jej rozbudowę, zachowując jednocześnie niską złożoność obliczeniową.

Przy projektowanej modyfikacji ALCM$_+$ wykorzystano podział na sześć kontekstów (trzy główne i trzy uzupełniające), z których każdy ma swój własny zestaw współczynników predykcji. Zasady podziału omówiono w podrozdziale 3.2.

Autorski algorytm adaptacji współczynników predykcji jest następujący: Najpierw wyznaczana jest wartość szybkości adaptacji:

$$\mu = \frac{1}{5 \cdot 2^{12} \cdot m} \cdot \sum_{i=1}^{m} \left| P(0) - P(i) \right|, \tag{5.5}$$

gdzie m określa wielkość otoczenia najbliższego sąsiedztwa kodowanego piksela $P(0)$, które ma wpływ na poziom adaptacji. Następnie są wyznaczane wartości: maksymalna $P_{\max} = P(p)$ oraz minimalna $P_{\min} = P(q)$ spośród najbliższych r pikseli sąsiedztwa. W przypadku kilku pikseli mających tę samą wartość minimalną (maksymalną) wybierany jest ten, który znajduje się najbliżej kodowanego piksela $P(0)$. W ostatnim kroku jest wykonywana adaptacja dwóch współczynników predykcji zgodnie z poniższą zasadą:

if $\hat{x} < P(0)$ **{**
$b_p(n+1) = b_p(n) + \mu$
$b_q(n+1) = b_q(n) - \mu$
}
else if $\hat{x} > P(0)$ **{**
$b_p(n+1) = b_p(n) - \mu$
$b_q(n+1) = b_q(n) + \mu$
}

Wyniki badań metody ALCM+ zaprezentowano w pracy [124]. Najlepsze rezultaty otrzymano przy rzędzie r = {14, 37, 15} dla kontekstów głównych (dla, odpowiednio, obrazów 256 × 256, 512 × 512 oraz 720 × 576) i r = 6 dla kontekstu gładkiego oraz odpowiednio dużej wartości m = 54 (im wyższa wartość m, tym niższa średnia bitowa, choć większe wartości wnoszą już tylko marginalną poprawę).

Z porównania wartości uzyskanych z 45 pomiarów wynika, że metoda ALCM+ dała średnią niższą o 0,028 bitu na piksel w porównaniu z LMS opisaną w podrozdziale 5.2 (patrz tab. 5.3).

5.4. Metoda adaptacji CoBALP i jej udoskonalenie

Kolejną metodą adaptacji współczynników predykcji jest metoda CoBALP (ang. *Context-Based Adaptive Linear Prediction*) zaproponowana w pracy [107]. Dokonano w niej podziału na 199 kontekstów głównych. W niniejszej pracy po szczegółowej analizie wykazano, iż użycie zaledwie siedmiu (lub pięciu dla mniejszych obrazów – patrz podrozdział 3.2) kontekstów, w połączeniu z mieszaną metodą korekcji składowej stałej opisaną w podrozdziale 4.3, daje znacznie lepsze rezultaty. Wynika to między innymi z faktu, iż początkowo zachodzi potrzeba wstępnego dostrojenia współczynników predykcji skojarzonych z poszczególnymi kontekstami, co przy dużej liczbie 199 kontekstów wymaga wielu pikseli obrazu, taki niekorzystny efekt nazywamy rozrzedzeniem kontekstu [93].

W porównaniu z LMS główną ideą CoBALP, gwarantującą wysoką efektywność, jest zastąpienie wartości pikseli $P(j)$ ich różnicami $D(j)$, których zestawienie znajduje się w tab. 5.2. Wówczas wartość przewidywaną wyznaczamy jako:

$$\hat{x} = P(2) + \sum_{j=1}^{r} b_j \cdot D(j) . \tag{5.6}$$

Piksel $P(2)$ wybrano jako jeden z dwóch najbliższych sąsiadów kodowanego $P(0)$, a jest on, w odróżnieniu od $P(1)$, położony bardziej centralnie w stosunku do pozostałych pikseli branych pod uwagę w zestawie różnic $D(j)$.

Adaptacja współczynników b_j odbywa się następująco:

$$b_j(n+1) = b_j(n) + \mu_j \cdot \bar{e} \cdot D(j) \tag{5.7}$$

przy ograniczniku φ = 7 wykorzystywanym do wyznaczenia wartości $\bar{e}$ (patrz wzór (5.2)). Początkowo wartości b_j są wyzerowane. Wartość μ_j jest zależna od współczynnika skalującego ζ_j oraz zmiennej c_j (ustawianej początkowo na 0):

$$\mu_j = \frac{\zeta_j}{1 + c_j} \cdot 10^{-6} . \tag{5.8}$$

Zmienna c_j jest wyznaczana iteracyjnie [107]:

$$c_j(n+1) = \frac{7}{8} c_j(n) + \frac{1}{8} \left| D(j) \right| . \tag{5.9}$$

W pierwotnej wersji CoBALP ustalono rząd predykcji na $r = 9$. Podczas badań omawianych w tej pracy, chcąc uzyskać wyższą efektywność, zwiększono rząd do $r = 46$, eksperymentalnie dobierając różnice $D(j)$ oraz odpowiadające im współczynniki skalujące ζ_j. Zestaw tych par zawiera tab. 5.2.

Tab. 5.2. Różnice $D(j)$ oraz odpowiadające im współczynniki skalujące ζ_j

j	$D(j)$	ζ_j	j	$D(j)$	ζ_j	j	$D(j)$	ζ_j
1	$P(1) - P(3)$	315	17	$P(3) - P(11)$	80	33	$P(19) - P(31)$	45
2	$P(3) - P(2)$	110	18	$P(14) - P(17)$	45	34	$P(20) - P(32)$	55
3	$P(2) - P(4)$	250	19	$P(8) - P(16)$	90	35	$P(21) - P(33)$	70
4	$P(1) - P(5)$	240	20	$P(6) - P(9)$	130	36	$P(28) - P(34)$	50
5	$P(2) - P(6)$	180	21	$P(11) - P(19)$	55	37	$P(23) - P(35)$	60
6	$P(3) - P(8)$	130	22	$P(11) - P(20)$	40	38	$P(24) - P(38)$	80
7	$P(3) - P(7)$	100	23	$P(12) - P(21)$	70	39	$P(31) - P(36)$	40
8	$P(4) - P(9)$	140	24	$P(12) - P(22)$	70	40	$P(32) - P(37)$	55
9	$P(4) - P(10)$	90	25	$P(13) - P(23)$	60	41	$P(30) - P(39)$	15
10	$P(2) - P(8)$	100	26	$P(14) - P(24)$	80	42	$P(34) - P(40)$	90
11	$P(6) - P(14)$	100	27	$P(15) - P(25)$	23	43	$P(35) - P(41)$	23
12	$P(4) - P(12)$	100	28	$P(18) - P(28)$	45	44	$P(26) - P(42)$	25
13	$P(5) - P(13)$	100	29	$P(16) - P(26)$	50	45	$P(41) - P(45)$	20
14	$P(7) - P(15)$	55	30	$P(24) - P(27)$	40	46	$P(32) - P(46)$	33
15	$P(10) - P(18)$	80	31	$P(19) - P(29)$	50			
16	$P(1) - P(2)$	260	32	$P(22) - P(30)$	55			

W tabeli 5.3 przedstawiono pomiary średnich bitowych, które potwierdzają wysoką skuteczność proponowanego udoskonalenia metody CoBALP$_+$. Wyniki badań opublikowano w pracy [126].

5.5. Metoda mieszana M-LMS

Spośród przeanalizowanych w tym rozdziale metod predykcyjnych o niskiej złożoności obliczeniowej możemy wybrać dwie najefektywniejsze (modyfikacje CoBALP$_+$ i ALCM$_+$) i na ich podstawie zbudować autorski mieszany model predykcyjny M-LMS (ang. *Mixed-LMS*). Najpierw wyznacza się wartości przewidywane uzyskane obiema metodami ($\hat{x}_{CoBALP}$ oraz $\hat{x}_{ALCM}$), następnie, dodając ich indywidualne mieszane modele korekcji skumulowanych błędów predykcji (patrz podrozdział 4.3), wyznaczamy wartość błędu predykcji $\hat{x}_{M-LMS}$ jako kombinację liniową:

$$\hat{x}_{M-LMS} = 0{,}8\dot{x}_{CoBALP} + 0{,}2\dot{x}_{ALCM} . \tag{5.10}$$

Również do wyniku tego predyktora jest dodawana jego wartość otrzymana z mieszanego modelu korekcji skumulowanych błędów predykcji. Okazuje się bowiem, że taki zabieg przynosi dalszy wzrost efektywności kompresji.

Wyniki pomiarów entropii i średniej bitowej dla metody M-LMS znajdują się w tab. 5.3.

Tab. 5.3. Porównanie trzech metod kompresji wykorzystujących adaptacyjną predykcję liniową

Obrazy	Entropia $ALCM_+$	Średnia $ALCM_+$	Entropia $CoBALP_+$	Średnia $CoBALP_+$	Entropia M-LMS	Średnia M-LMS
Aerial	5,01705	4,63277	4,94600	4,56295	4,94052	4,55689
Airfield	5,02973	4,87826	5,00088	4,84890	4,99550	4,84460
Airplane	4,03063	3,65980	3,94134	3,60685	3,93950	3,60447
Baboon	6,05762	5,78122	6,01859	5,74457	6,01343	5,73973
Barbara	4,76602	4,38830	4,53281	4,23264	4,53546	4,23130
Boat	4,72575	4,52413	4,62539	4,44499	4,62206	4,44107
Bridge	3,67162	3,41131	3,64716	3,38065	3,64516	3,37970
Couple	4,32070	4,12664	4,27957	4,09456	4,27389	4,08941
Crowd	4,15005	3,69306	4,05818	3,62898	4,05116	3,61370
Elaine	4,60409	4,45188	4,51278	4,36867	4,51127	4,36670
Finger	5,26174	5,24570	5,19259	5,18899	5,18963	5,18538
Frog	5,11543	5,04271	5,07358	5,00470	5,07285	5,00380
Goldhill	4,72869	4,53108	4,67109	4,48556	4,66856	4,48313
Harbour	4,86233	4,34345	4,80008	4,28636	4,79529	4,28205
Lax	5,76355	5,57027	5,73116	5,54170	5,72787	5,53814
Lennagrey	4,23865	4,01204	4,16983	3,96143	4,16702	3,95950
$Lena_{TMW}$	4,58975	4,39377	4,52752	4,34689	4,52433	4,34484
Man	4,64526	4,33125	4,59386	4,28496	4,59005	4,28155
Peppers	4,47210	4,29803	4,41327	4,24364	4,40992	4,24125
Sailboat	4,80685	4,49939	4,75211	4,43330	4,74809	4,43161
Seismic	2,39094	2,34002	2,30818	2,27248	2,30432	2,26852
Shapes	1,73764	1,38852	1,72229	1,38116	1,67766	1,33041
Tank	3,96640	3,80768	3,93756	3,78188	3,93524	3,78076
Truck	4,24247	4,05597	4,22362	4,03563	4,21901	4,03247
Woman1	4,15776	3,90636	4,09704	3,86274	4,09362	3,85925
Woman2	3,31666	3,06096	3,27975	3,03606	3,27705	3,03202
Balloon	2,84686	2,73480	2,76590	2,67658	2,76110	2,67151
Barb	4,51692	4,17450	4,30918	4,04322	4,30749	4,03728
Barb2	4,74331	4,37212	4,65694	4,31603	4,64881	4,30465
Board	3,60604	3,41807	3,51919	3,36666	3,51051	3,35675
Boats	3,98940	3,68441	3,89465	3,62666	3,88840	3,61967
Girl	3,80439	3,64006	3,69089	3,55466	3,68565	3,54735
Gold	4,56406	4,29627	4,51150	4,26559	4,50758	4,25767
Hotel	4,48394	4,19945	4,35927	4,10069	4,36211	4,10310
Zelda	3,69888	3,59905	3,65214	3,56420	3,64646	3,55894
Bridge256	5,80905	5,62498	5,78416	5,59659	5,77878	5,59116
Camera	4,64940	4,18185	4,55713	4,10118	4,55819	4,10316
Couple256	3,93298	3,55341	3,89920	3,53563	3,88613	3,51720
Earth	3,69057	2,86731	3,68073	2,85466	3,67869	2,85359
Elif	3,07159	2,85257	2,95556	2,75404	2,95045	2,75090
Noisesquare	5,29979	5,31001	5,24560	5,25801	5,23726	5,25024
Omaha	6,31774	6,07864	6,31296	6,07581	6,30783	6,06900
Sena	3,32896	3,06602	3,18860	2,96280	3,18620	2,96016
Sensin	3,55073	3,36267	3,42097	3,26114	3,41800	3,26025
Sinan	3,27388	3,10806	3,17048	3,03317	3,16341	3,02954
Średnia	4,30773	4,05553	4,23635	4,00019	4,23137	3,99476

5.6. Analiza prostych metod adaptacyjnych

W tym podrozdziale zostanie przedstawiona analiza efektywności i czasu kodowania metodami adaptacyjnymi omówionymi w tym rozdziale. Wyniki pomiarów entropii i średnich bitowych przedstawia tab. 5.3. Czas kodowania obrazu Lennagrey o wymiarach 512×512 pikseli (kod w języku C bez optymalizacji – proces kodowania z wykorzystaniem procesora Pentium4 2,8 GHz) metodą $ALCM_+$ wyniósł 2,46 s, natomiast metodą $CoBALP_+$ 2,66 s.

Oprócz opublikowania w pracy [107] kodera CoBALP T. Strutz zaprezentował jego efektywniejszą wersję (program WaveConvert Version 1.2 z 2002 roku), która wymagając dwukrotnie dłuższego czasu kodowania niż zaproponowana tu wersja $CoBALP_+$, pozwoliła na otrzymanie średniej bitowej z dziewięciu pomiarów równej 3,865 (patrz kolumna $CoBALP_{max}$ w tab. 5.4), co daje efektywność zaledwie zbliżoną do kodera CALIC.

Metodę M-LMS przebadaną w podrozdziale 5.5 porównano w tab. 5.4 z metodą CALIC oraz z kilkoma znanymi z literatury metodami kompresji wykorzystującymi adaptacyjną predykcję liniową. Ustępuje ona wyłącznie metodom zaklasyfikowanym (zgodnie z opisem zawartym w podrozdziale 1.2) jako te o nieakceptowalnym czasie kodowania, wynoszącym kilkadziesiąt minut (patrz tab. 9.9). Średnia z dziewięciu pomiarów dla proponowanej tu metody M-LMS jest natomiast porównywalna ze średnią uzyskaną dla metody GLICBAWLS [85], choć z użyciem M-LMS niższe średnie bitowe otrzymano aż w dwóch trzecich przypadków. Przy porównywaniu czasu kodowania korzystniej wypada M-LMS – kodowanie obrazu Lennagrey o wymiarach 512×512 pikseli trwało zaledwie 3 s, a w przypadku GLICBAWLS aż 21,2 s.

Tab. 5.4. Pomiar średnich bitowych standardowych obrazów testowych

Obrazy	$CoBALP_{max}$ [107]	CALIC [149]	LAT-RLMS [75]	$ALCM_+$	$CoBALP_+$	M-LMS	GLICBAWLS [85]
Balloon	2,853	2,78	2,75	2,735	2,677	2,672	2,640
Barb	4,176	4,31	4,15	4,175	4,043	4,037	3,916
Barb2	4,440	4,46	4,45	4,372	4,316	4,305	4,318
Board	3,492	3,51	3,48	3,418	3,367	3,357	3,392
Boats	3,780	3,78	3,74	3,684	3,627	3,620	3,628
Girl	3,696	3,72	3,68	3,640	3,555	3,547	3,565
Gold	4,382	4,35	4,34	4,296	4,266	4,258	4,276
Hotel	4,219	4,18	4,21	4,199	4,101	4,103	4,177
Zelda	3,749	3,69	3,61	3,599	3,564	3,559	3,537
Średnia	3,865	3,864	3,823	3,791	3,724	3,717	3,717

Szczegółowa analiza i porównanie metod adaptacyjnych omówionych w tym rozdziale z statyczną predykcją liniową zaprezentowaną w podrozdziale 4.4 pozwalają określić dalszy kierunek badań nad zwiększaniem efektywności bezstratnej kompresji obrazów. Dla metod $CoBALP_+$ oraz M-LMS uzyskano niższą średnią niż dla metody statycznej przy zbliżonym czasie kodowania. W kolejnym rozdziale zostaną omówione złożone metody adaptacyjne pozwalające uzyskać dalszą poprawę stopnia kompresji.

6. Złożone metody adaptacji współczynników predykcji

6.1. Metoda RLS

W tym rozdziale zostaną przedstawione złożone metody adaptacji współczynników predykcji liniowej z uwzględnieniem analizy ich złożoności obliczeniowej. Główną wadą takich metod adaptacyjnych jest fakt, iż wysoka złożoność obliczeniowa dotyczy także dekodera, są to bowiem metody symetryczne czasowo.

Metoda RLS (ang. *Recursive Least Square*) pozwala na minimalizację błędu średniokwadratowego z uwzględnieniem lokalnych zmian sygnału. Adaptacja współczynników predykcji wymaga operacji na macierzy **K**, którą inicjalizujemy jako macierz jednostkową **I** pomnożoną przez stałą $\varepsilon << 1$. W proponowanym rozwiązaniu, oznaczanym dalej jako RLS$_+$, zastosowano podział kontekstowy omówiony w podrozdziale 3.2.1, zatem dla każdego z kontekstów istnieje osobna macierz **K** o wymiarach $r \times r$. Każda z nich jest inicjalizowana jako:

$$\mathbf{K} = 0{,}00048 \cdot \mathrm{diag}(\overline{\mathbf{D}}), \tag{6.1}$$

gdzie wektor $\overline{\mathbf{D}} = [\overline{d}_1, \overline{d}_2, \ldots, \overline{d}_r]^T$ składa się z elementów $\overline{d}_j$ wyznaczanych zgodnie ze wzorem (3.9), a funkcja $\mathrm{diag}(\overline{\mathbf{D}})$ definiuje macierz kwadratową złożoną z zer, z wyjątkiem głównej przekątnej, na której umieszczono wartości wektora $\overline{\mathbf{D}}$. Podobnie jak w przypadku ALCM$_+$ oraz CoBALP$_+$, także i w tej metodzie uzyskano znaczną poprawę efektywności dzięki zastosowaniu podziału kontekstowego (patrz podrozdział 3.2). W każdym z kontekstów wartości początkowe wektora współczynników predykcji ustala się jako **B** = [0,62, 0,625, –0,125, 0,125, –0,125, –0,125, 0, ... , 0]. Poniżej przedstawiono algorytm adaptacji współczynników predykcji obliczanych po zakodowaniu każdego kolejnego piksela [14]. W pierwszym kroku wyznacza się wektor **U**:

$$\mathbf{U} = \mathbf{K}(n) \cdot \mathbf{X}, \tag{6.2}$$

gdzie wektor $\mathbf{X} = [P(1), P(2), \ldots, P(r)]^T$, po czym jest obliczany współczynnik uczenia μ:

$$\mu = \frac{1}{1{,}25 + \mathbf{X}^T \cdot \mathbf{U}}. \tag{6.3}$$

Następnie dokonywana jest adaptacja macierzy **K** dla kolejnej chwili:

$$\mathbf{K}(n+1) = 1{,}0005 \cdot (\mathbf{K}(n) - \mu \cdot \mathbf{U} \cdot \mathbf{U}^T). \tag{6.4}$$

Wartość tej macierzy jest podstawiana do wzoru na adaptację wektora współczynników predykcji $\mathbf{B} = [b_1, b_2, \ldots, b_r]^T$:

$$\mathbf{B}(n+1) = \mathbf{B}(n) + \bar{e} \cdot \mathbf{K}(n+1) \cdot \mathbf{X}, \tag{6.5}$$

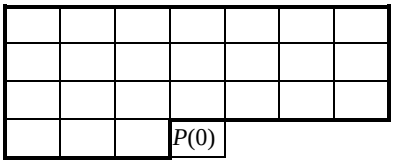
gdzie $\bar{e}$ jest wyznaczane zgodnie ze wzorem (5.2) przy ograniczniku $\varphi = 14$. Współczynnik 1,0005 dobrano eksperymentalnie, wykorzystując doświadczenia z pracy [14].

Najlepsze rezultaty uzyskano dla rzędu $r = 46$ [125], dla którego czas kodowania obrazu Lennagrey wynosi 12,42 s. Porównanie podstawowej metody RLS z jej rozwinięciem RLS_+ wykorzystującym podział kontekstowy przedstawiono w tab. 6.3.

6.2. Metoda OLS

Opisana w podrozdziale 2.2 metoda statycznej predykcji liniowej, oparta na kryterium MMSE, wymaga rozwiązania układu równań z r niewiadomymi b_j. Do tego celu wykorzystuje się macierz **R** oraz wektor **P**, wyznaczane, odpowiednio, ze wzorów (2.2) i (2.3). Statyczność tej metody oznaczała niezmienność wektora **B** współczynników predykcji w całym obszarze kodowanego obrazu. Wiadomo jednak, że ze względu na zmienny charakter danych w różnych częściach obrazu, metody z adaptacją współczynników predykcji umożliwiają uzyskiwanie lepszego stopnia kompresji. W metodzie OLS (ang. *Ordinary Least Square*) [149, 150] proponowanej w tym podrozdziale współczynniki predykcji są wyznaczane adaptacyjnie, indywidualnie dla każdego kodowanego piksela, z wykorzystaniem minimalizacji błędu średniokwadratowego w pewnym ograniczonym obszarze.

W metodzie OLS przyjmuje się założenie lokalnej stacjonarności w pewnym obszarze Q, nazywanym oknem treningowym, dla którego wyznacza się metodą MMSE wektor **B**, rozwiązując równanie macierzowe (2.1). Obszar Q składa się z pikseli leżących w W wierszach powyżej $P(0)$, ale oddalonych nie więcej niż o W pikseli w lewą lub prawą stronę. Uzupełnieniem obszaru Q jest W pikseli leżących w wierszu piksela $P(0)$ po jego lewej stronie. Obszar Q składa się zatem z $2W(W + 1)$ pikseli i tylko do tego obszaru należy ograniczyć dane niezbędne do wyznaczenia macierzy **R** i wektora **P**. Przykładowy obszar Q, o wysokości $W = 3$, przedstawiono na rys. 6.1. W sytuacji, gdy kodowany piksel leży blisko lewej lub prawej krawędzi obrazu, może się zdarzyć, że część obszaru Q znajdzie się na zewnątrz obrazu. Aby temu zapobiec, W górnych wierszy obszaru Q przesuwa się w taki sposób, aby ta część o stałej szerokości $2(W + 1)$ nie wystawała poza lewy i prawy brzeg obrazu. W początkowych wierszach obrazu oraz w sytuacji, gdy macierz **R** jest osobliwa, jest stosowany stały predyktor liniowy o $r = 6$ i współczynnikach **B** = [0,62, 0,625, –0,125, 0,125, –0,125, –0,125].



Rys. 6.1. Obszar treningowy Q o wysokości $W = 3$

Należy zwrócić uwagę na możliwość występowania źle uwarunkowanych układów równań, których wynikiem może okazać się model predykcyjny o sumie współczynników mocno odbiegającej od jedności lub o współczynnikach nienależących do przedziału (–2; 2) charakterystycznego dla danych obrazowych. Pomocne staje się rozwiązanie polegające na kontroli sumy wartości bezwzględnych współczynników predykcji. Gdy suma ta jest większa od dwukrotności rzędu predykcji, można zastosować domyślny model predykcyjny szóstego rzędu o współczynnikach **B** = [0,62, 0,625, –0,125, 0,125, –0,125, –0,125].

Kolejną próbą poprawienia efektywności i uniknięcia źle uwarunkowanego równania macierzowego może być uwzględnienie dodatkowej stałej, którą wprowadza się, modyfikując równanie (2.1) do postaci [43, 85]:

$$\mathbf{B} = (\mathbf{R} + u_{\text{bias}} \cdot \mathbf{I})^{-1} \cdot \mathbf{P}, \tag{6.6}$$

gdzie eksperymentalnie w przypadku OLS dobrano $u_{\text{bias}} = 800$.

Metodę OLS charakteryzuje wysoka złożoność obliczeniowa związana z wyznaczaniem dla każdego kodowanego piksela wektora **P**, macierzy **R** i jej odwrotności niezbędnej do rozwiązania równania (2.1). Szczegółowa analiza złożoności obliczeniowej znajduje się w podrozdziale 6.7. W literaturze opisywane są próby zmniejszenia złożoności obliczeniowej przez szacowanie modeli wyższego rzędu z użyciem rzędu $r - 1$, a także przez obliczanie modelu predykcyjnego jedynie w sytuacjach wykrycia obszaru krawędziowego lub o dużej zmienności [48], a nie dla każdego kolejnego piksela, jak to jest w przypadku OLS.

Metoda OLS daje nieco niższą średnią bitową dla 45 obrazów testowych od RLS. W odróżnieniu od metody RLS_+ w przypadku metody OLS mamy do czynienia z tylko jednym zestawem współczynników predykcji, gdzie przy $W = 12$ i $r = 14$ otrzymano średnią 4,00039 bitu na piksel (patrz tab. 6.3). Natomiast dzięki przełączaniu kontekstów w metodzie RLS_+ możliwe było uzyskanie niższej średniej równej 3,98151 bitu. W tabeli 6.2 przedstawiono porównanie średnich bitowych dla proponowanej tu wersji metody OLS ($W = 10$, $r = 14$) i dla metody opisanej w pracy [148] ($W = 15$, $r = 18$), której wynik okazał się gorszy o ponad 0,021 bitu (dla dziewięciu podstawowych obrazów testowych).

6.3. Metoda WLS

Metoda OLS opisana w poprzednim podrozdziale uwzględnia zmienność cech danych w czasie kodowania (fragmenty obrazu można zaklasyfikować np. jako gładkie, krawędziowe, teksturowane), lecz przyjmuje stacjonarność tych danych w najbliższym otoczeniu, definiując okno treningowe Q, na podstawie którego są wyznaczane macierz **R** i wektor **P**. W pracy [143] zasugerowano, że metodę OLS można rozbudować o dwa dodatkowe kryteria, pozwalające zwiększyć jej wydajność. Pierwszym z nich jest możliwość przewidywania najlepszego rzędu liniowego modelu predykcji dla każdego z kodowanych pikseli z osobna, a także próba doboru najlepszych pikseli sąsiedztwa dla danego rzędu predykcji (w sensie poziomu korelacji występującego w ramach okna treningowego Q). W praktyce dobór taki jest bardzo czasochłonny

i trudny do zautomatyzowania, jednak wprowadzając pewne uproszczenia, skutecznie rozwiązuje ten problem metoda AVE-WLS, która zostanie opisana w podrozdziale 6.4. Drugie kryterium poprawy wykorzystuje możliwości selektywnego wyboru danych z okna treningowego Q (stosując kryterium podobieństwa najbliższego sąsiedztwa aktualnie kodowanego piksela do sąsiedztw innych pikseli znajdujących się w oknie treningowym Q) zamiast używania wszystkich pikseli znajdujących się w tym oknie. Także i w tym przypadku możliwe jest mniej uciążliwe rozwiązanie będące kompromisem między wyborem selektywnym a pełnym. Polega ono na wprowadzeniu wag zależnych od cech najbliższego sąsiedztwa.

Rozszerzenie metody OLS przez wprowadzenie dodatkowych wag jest określane mianem metody WLS (ang. *Weighted Least Square*). W metodzie tej również należy rozwiązać równanie macierzowe (6.6), wymagana jest jednak pewna modyfikacja wzorów (2.2) i (2.3). Wersja WLS zaproponowana w pracy [85] uwzględnia zmniejszającą się wykładniczo zawartość informacyjną pikseli wraz ze wzrostem ich odległości od aktualnie kodowanego $P(0)$, czego nie uwzględniono w pracy [143]. Choć metoda GLICBAWLS, podobnie jak OLS, dla każdego kodowanego piksela rozwiązuje równanie macierzowe (6.6), to nie stosuje jednak zasady okna treningowego Q. W pracy [148] zaproponowano odmianę WLS z ograniczeniem do okna treningowego. W obu propozycjach wzory (2.2) i (2.3) zostały rozszerzone o parametr wagowy ψ i przyjmują postać:

$$\mathbf{R}(j,i) = \sum_{y \in Q} \sum_{z \in Q} \psi_{(y,z)} \cdot P_{(y,z)}(i) \cdot P_{(y,z)}(j), \tag{6.7}$$

$$\mathbf{P}(j) = \sum_{y \in Q} \sum_{z \in Q} \psi_{(y,z)} \cdot P_{(y,z)}(0) \cdot P_{(y,z)}(j), \tag{6.8}$$

gdzie indeks (y, z) wskazuje na aktualne umiejscowienie piksela $P(0)$ wewnątrz obszaru Q (y – numer wiersza, z – numer kolumny).

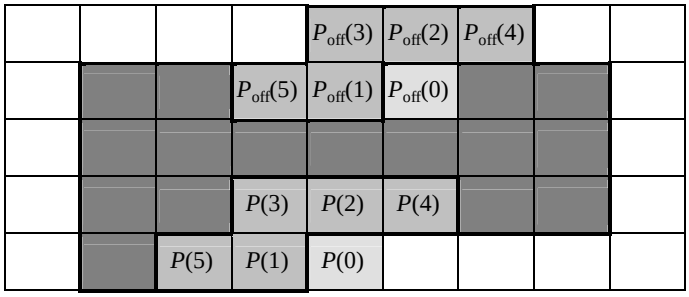
Kluczową rolę odgrywa tu zasada wyznaczania parametru wagowego ψ. W pracy [148] wartość ta jest odwrotnie proporcjonalna do kwadratu odległości euklidesowej między wzorcem złożonym z m pikseli $P(i)$ najbliższego sąsiedztwa aktualnie kodowanego piksela $P(0)$ a wzorcem $\mathbf{Y}_{\text{off}}$ o takim samym kształcie, składającym się z pikseli od $P_{\text{off}}(1)$ do $P_{\text{off}}(m)$, skojarzonym z pikselem $P_{\text{off}}(0)$, należącym do okna treningowego Q:

$$\psi = \frac{1}{1 + \sum_{j=1}^{m} (P(j) - P_{\text{off}}(j))^2}. \tag{6.9}$$

Przykład dla $m = 5$ oraz $W = 3$ przedstawia rys. 6.2. W tej pracy parametr ψ w metodzie WLS został udoskonalony i jest wyznaczany ze wzoru:

$$\psi = \frac{1}{350 + \sum_{j=1}^{m} \left(\overline{d}_j \cdot (P(j) - P_{\text{off}}(j))\right)^2}. \tag{6.10}$$

Wprowadzenie takiego parametru wagowego uniemożliwia korzystanie z uproszczeń w aktualizacji macierzy **R** i wektora **P** występujących w OLS. Obliczenia macierzy i wektora muszą być wyznaczane ponownie w całości dla każdego kodowanego piksela. Z tego względu kodowanie metodą WLS jest znacznie bardziej czasochłonne niż metodą OLS. Szczegółowa analiza złożoności obliczeniowej znajduje się w podrozdziale 6.7. Ponadto uzyskanie wysokiej efektywności metody WLS wymaga większej wartości W niż stosowana w metodzie OLS.



Rys. 6.2. Przykład obrazujący umiejscowienie wzorców w obszarze Q najbliższego sąsiedztwa piksela $P(0)$ dla $m = 5$ oraz $W = 3$

W badaniach przedstawionych w tab. 6.3 użyto wartości $W = 14$ i rzędu predykcji $r = 18$. Średnia bitowa dla 45 obrazów testowych wyniosła 3,95356 bitu (dla OLS było to 4,00039 bitu).

6.4. Metoda AVE-WLS

Wśród metod adaptacyjnych zaprezentowanych w tym rozdziale najlepszą efektywność uzyskała WLS (patrz tab. 6.3). W pracy [29] postawiono tezę, iż w metodzie OLS dla każdego z kodowanych pikseli może istnieć indywidualny optymalny rząd predykcji. Nie sposób jednak precyzyjnie przewidzieć, jaki rząd byłby optymalny, a wyznaczenie jednej najlepszej wartości r dla całego obrazu jest tylko rozwiązaniem kompromisowym i wymaga wielokrotnego kodowania całego obrazu. Pewne próby rozwiązania tego problemu podjęto w pracy [143]. Można zaproponować inne, stosunkowo proste rozwiązanie, polegające na wyznaczaniu predyktorów dla kolejnych rzędów predykcji, które to predyktory później zostaną połączone w jeden model predykcyjny. W praktyce oznacza to rozwiązanie wielu równań macierzowych (6.6) z poszczególnymi rzędami od r_{min} do r_{max}. Macierz $\mathbf{R}_{max}$ w proponowanej tu metodzie AVE-WLS, będącej rozwinięciem metody WLS opisanej w podrozdziale 6.3, jest wyznaczana ze wzoru (6.7) dla najwyższego rzędu, czyli wartości r_{max}. Pozostałe mniejsze macierze $\mathbf{R}_j$ można uzyskać jako podmacierze będące fragmentem $\mathbf{R}_{max}$ (to samo dotyczy wektora $\mathbf{P}_{max}$ i jego podwektorów $\mathbf{P}_j$).

Najprostszą metodą wspólnego wykorzystania wielu predyktorów WLS jest (dla odpowiednio dużej wartości W [29]) wyznaczanie ich średniej arytmetycznej (stąd nazwa AVE-WLS), co sprowadza się do obliczania średniej ze wszystkich uzyskanych zestawów współczynników predykcji od $\mathbf{B}_{min}$ do $\mathbf{B}_{max}$:

$$\mathbf{B}_{\text{ave}} = \frac{1}{r_{\max} - r_{\min} + 1} \cdot \sum_{j=r_{\min}}^{r_{\max}} \mathbf{B}_j \ . \tag{6.11}$$

Wektory $\mathbf{B}_j$ o niższych od $r_{\max}$ rzędach uzupełnia się wcześniej o zerowe współczynniki celem uzyskania wektorów o rozmiarze $r_{\max} \times 1$. Parametr wagowy ψ w metodzie AVE-WLS został rozbudowany i uwzględnia także pomysły z prac [85, 148]. Zaprojektowano dwie jego autorskie wersje dla metody AVE-WLS1:

$$\psi_{(y,z)}^{(1)} = \frac{0{,}25 + 0{,}8^{\sqrt{(\Delta y)^2 + (\Delta z)^2}}}{350 + \sum_{j=1}^{m} \left(\bar{d}_j \cdot \left(P(j) - P_{\text{off}}(j) \right) \right)^2} \tag{6.12}$$

oraz dla metody AVE-WLS2 (wzorowanej na [66, 101]):

$$\psi_{(y,z)}^{(2)} = \left(0{,}25 + 0{,}8^{\sqrt{(\Delta y)^2 + (\Delta z)^2}} \right) \cdot \mathrm{e}^{-0{,}000089 \sum_{j=1}^{m} \left(\bar{d}_j \cdot \left(P(j) - P_{\text{off}}(j) \right) \right)^2}, \tag{6.13}$$

gdzie $\sqrt{(\Delta y)^2 + (\Delta z)^2}$ jest odległością euklidesową między pikselami $P(0)$ i $P_{\text{off}}(0)$ (patrz rys. 6.2). Wartość $u_{\text{bias}} = 0{,}9$ w przypadku AVE-WLS1 oraz $u_{\text{bias}} = 40$ dla AWE-WLS2 (patrz wzór (6.6)).

W badaniach eksperymentalnie dobrano $W = 14$, a także dla obrazów o rozdzielczości 256×256 zakres rzędów predykcji od 4 do 22, dla obrazów o rozdzielczości 512×512 zakres r_j od 4 do 28 oraz dla obrazów o rozdzielczości 720×576 zakres r_j od 6 do 24. W porównaniu z metodą WLS wynik średniej bitowej uzyskanej dla 45 obrazów testowych kodowanych metodą AVE-WLS1 był o 0,01698 bitu mniejszy, wyniósł bowiem 3,93658 bitu na piksel. Mimo iż metoda AVE-WLS2 daje gorszy rezultat o ponad 0,01 bitu na piksel w porównaniu z AVE-WLS1, to jest ważnym składnikiem metody Blend-24, opisanej w rozdziale 9.

6.5. Metoda NLMS

Metoda NLMS (ang. *Normalized LMS*) jest rozszerzeniem podstawowej metody LMS opisanej w podrozdziale 5.2. W projektowanym w tej pracy systemie bezstratnej kompresji obrazów głównym zastosowaniem NLMS jest dalsza minimalizacja błędu średniokwadratowego, który jest wyznaczany przez podstawową metodę predykcyjną jako wartość $\hat{x}$. Zastosowano tu technikę łączenia kaskadowego [45, 101], w której kolejno występujące po sobie bloki mają za zadanie zmniejszyć średni błąd kwadratowy sygnału otrzymanego z wejścia. Podobnie jak w podstawowej wersji LMS, także tu wykorzystuje się wzory (5.1) i (5.2), zmienia się jedynie sposób wyznaczania współczynnika szybkości adaptacji [22]:

$$\mu = \frac{1}{2^7 \cdot \left(10 + \sum_{j=1}^{r_{NLMS}} e^2(j)\right)}, \tag{6.14}$$

gdzie $e(j)$ to j-ty błąd predykcji (najbliższego sąsiedztwa) uzyskany na poprzednim etapie modelowania. Można nieco poprawić efektywność metody, wprowadzając dopasowanie współczynnika szybkości adaptacji do poziomu zmienności obrazu, uwzględniając przy tym malejącą wartość adaptacji dla błędów $e(j)$ o wzrastającym indeksie j, w następujący sposób:

$$\mu_j = \frac{\overline{d}_j}{2^3 \cdot \sqrt{\overline{\sigma}^2} \cdot \left(10 + \sum_{i=1}^{r_{NLMS}} \sqrt{\overline{d}_i} \cdot e^2(i)\right)}, \tag{6.15}$$

gdzie $\overline{\sigma}^2$ jest średnią arytmetyczną wszystkich wariancji wagowych $\tilde{\sigma}^2$ w kodowanym obrazie, wyznaczanych zgodnie ze wzorem (3.12).

W przypadku proponowanej tu autorskiej metody wartości współczynników predykcji ustawia się początkowo na 0, czyli $\mathbf{B}_{NLMS}$ = [0, ... , 0]. Wartość korygująca $\hat{x}_{NLMS}$ modelu NLMS wyznaczana jest ze wzoru:

$$\hat{x}_{NLMS} = \sum_{j=1}^{r_{NLMS}} b_{NLMS(j)} \cdot e(j). \tag{6.16}$$

W wyniku zastosowania NLMS uzyskujemy nowe błędy e_{NLMS}:

$$e_{NLMS}(0) = P(0) - (\hat{x} + \hat{x}_{NLMS}), \tag{6.17}$$

które mogą zostać wykorzystane do poszerzenia o drugi człon modelu korygującego opartego na wzorze (6.16), do postaci:

$$\hat{x}_{NLMS+} = \sum_{j=1}^{r_{NLMS}} b_{NLMS(j)} \cdot e(j) + \sum_{j=1}^{r_{NLMS+}} b_{NLMS+(j)} \cdot e_{NLMS}(j). \tag{6.18}$$

Wówczas metodę tę określamy jako NLMS+, a wektor $\mathbf{B}_{NLMS+}$ również jest inicjalizowany zerami. Adaptacja współczynników predykcji ze wzoru (5.1) przekształca się do postaci:

$$b_{NLMS(j)}(n+1) = b_{NLMS(j)}(n) + \mu_j \cdot \sqrt{\overline{d}_j} \cdot \overline{e} \cdot e(j) \tag{6.19}$$

oraz:

$$b_{NLMS+(j)}(n+1) = b_{NLMS+(j)}(n) + \mu_j \cdot \sqrt{\overline{d}_j} \cdot \overline{e} \cdot e_{NLMS}(j), \tag{6.20}$$

gdzie:

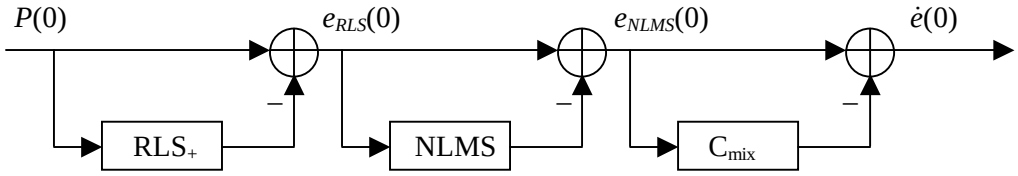
$$\bar{e} = \mathrm{sgn}(P(0) - (\hat{x} + \hat{x}_{NLMS+})) \cdot \min\{|P(0) - (\hat{x} + \hat{x}_{NLMS+})|, \varphi\} \tag{6.21}$$

jest wartością błędu predykcji z ogranicznikiem $\varphi = 14$.

Jako przykład kaskadowej metody można rozpatrzyć metodę RLS-NLMS, w której wejściem dla metody NLMS jest wartość błędu predykcji $e_{RLS}(0)$, wyznaczana metodą RLS+. Jak wynika z rys. 6.3, ostateczny błąd predykcji $\dot{e}(0)$, po uwzględnieniu wartości przewidywanej $\hat{x}_{RLS}$ obliczonej w bloku modelowania RLS+, a także wartości przewidywanej $\hat{x}_{NLMS}$ wyznaczonej w bloku modelowania NLMS oraz wartości C_{mix} uzyskanej w bloku korekcji skumulowanego błędu predykcji, jest wyliczany w następujący sposób:

$$\dot{e}(0) = P(0) - (\hat{x}_{RLS} + \hat{x}_{NLMS} + C_{\mathrm{mix}}) . \tag{6.22}$$

Kaskadowe połączenie pozwala na efektywne wykorzystanie wysokiego rzędu predykcji w metodzie NLMS, co nie było skutecznym rozwiązaniem w klasycznej metodzie LMS. Czas kodowania metodą RLS-NLMS przy r = 46 i r_{NLMS} = 72 wynosi dla obrazu Lennagrey 14,30 s. Podobny rezultat (14,62 s) uzyskano metodą OLS-NLMS+ przy r = 14 i W = 12, r_{NLMS} = 72, r_{NLMS+} = 30. Czas kodowania dodatkowego bloku NLMS to zatem niecałe 2 s, co zwłaszcza przy najbardziej złożonych metodach WLS i AVE-WLS, jest tylko małym dodatkowym nakładem.



Rys. 6.3. Kaskadowe połączenie kolejnych etapów wyznaczania błędu predykcji

6.6. Analiza rezultatów badań

W metodzie AVE-WLS dużą rolę odgrywa ustalenie maksymalnego rzędu predykcji. W przypadku rzędu $r_{\max}$ = 24 czas kodowania dla obrazu Lennagrey wynosi 457 s (co stanowi zaledwie 37,2% czasu kodowania metodą MRP 0.5), a dla rzędu $r_{\max}$ = 28 jest to już 608 s.

W przypadku metody RLS+ dołączenie bloku NLMS powoduje spadek średniej bitowej dla 45 obrazów testowych o zaledwie 0,0087 bitu. Jest to związane z wysoką elastycznością metody RLS+, która w swym schemacie adaptacyjnym zawiera długoterminową pamięć dopasowania do wcześniejszych danych. W przypadku metod OLS i AVE-WLS zyski z dołączenia bloku NLMS są znacznie większe i wynoszą średnio 0,04 bitu na piksel. Wynika to z ograniczania się tych metod do danych zawartych w pewnym oknie Q oraz ze znacznie mniejszych rzędów predykcji w porównaniu z RLS+. Dlatego metoda NLMS dołączona do metod OLS i AVE-WLS pozwala, dzięki wysokiemu rzędowi predykcji, usunąć sporą część nadmiarowości informacyjnej długoterminowej charakterystycznej dla tych metod. Porównanie średnich bito-

wych dla dwóch podstawowych zestawów testowych przedstawiono w tab. 6.1 i 6.2. Rezultaty pomiarów dla pełnego zestawu 45 obrazów testowych umieszczono w tab. 6.3.

Tab. 6.1. Porównanie średnich bitowych – test 1

Obrazy	RLS	RLS_+	RLS--NLMS	OLS	OLS--NMLS	WLS ($W = 14$, $r = 18$)	AVE--WLS	AVE--WLS1--NLMS+
Camera	4,15608	4,09428	4,09128	4,08351	4,07672	4,00397	3,98462	3,97865
Couple256	3,55466	3,52721	3,52756	3,49568	3,49078	3,45171	3,39153	3,39426
Noisesquare	5,23596	5,23657	5,22934	5,29633	5,26511	5,28156	5,26733	5,22803
Airplane	3,63761	3,60119	3,60226	3,61792	3,61018	3,60027	3,59117	3,58464
Baboon	5,76030	5,74580	5,74191	5,67781	5,66262	5,65852	5,64520	5,62794
$Lena_{TMW}$	4,38531	4,34658	4,34098	4,29582	4,29451	4,28695	4,26081	4,26085
Lennagrey	4,00357	3,96113	3,95530	3,90633	3,90498	3,89848	3,87031	3,87060
Peppers	4,28610	4,23821	4,20868	4,23878	4,20462	4,20771	4,19357	4,14323
Shapes	1,63300	1,49713	1,59990	1,50326	1,55516	1,48046	1,22625	1,27879
Balloon	2,72238	2,66175	2,63363	2,68898	2,62515	2,65755	2,63483	2,57008
Barb	4,11039	4,01514	4,00175	3,90540	3,87107	3,79572	3,77526	3,74421
Barb2	4,39051	4,30685	4,30030	4,28548	4,26364	4,21701	4,19244	4,16911
Gold	4,27388	4,25190	4,25119	4,25097	4,23363	4,24146	4,20932	4,19597
Średnia	4,01152	3,96029	3,96031	3,94202	3,92755	3,90626	3,86482	3,84972

Tab. 6.2. Pomiar wartości średnich bitowych – test 2

Obrazy	ALPC [86]	RLS_+ ($r = 46$)	RLS--NLMS ($r = 46$)	OLS [148] ($W = 15$, $r = 18$)	OLS ($W = 10$, $r = 14$)	OLS--NLMS ($W = 10$, $r = 14$)	WLS ($W = 14$, $r = 18$)	AVE--WLS ($W = 14$, $r = \overline{6;24}$)	AVE-WLS1--NLMS+ ($W = 14$, $r = \overline{6;24}$)
Balloon	2,74	2,66175	2,63363	2,690	2,68898	2,62521	2,65755	2,63483	2,57008
Barb	4,00	4,01514	4,00175	3,939	3,90540	3,87119	3,79572	3,77526	3,74421
Barb2	4,41	4,30685	4,30030	4,310	4,28548	4,26348	4,21701	4,19244	4,16911
Board	3,48	3,34925	3,34077	3,388	3,37364	3,34380	3,31780	3,30154	3,26955
Boats	3,77	3,60894	3,60168	3,638	3,61552	3,58220	3,56591	3,55182	3,52415
Girl	3,68	3,54189	3,52923	3,576	3,55453	3,51636	3,48891	3,46675	3,44106
Gold	4,39	4,25190	4,25119	4,273	4,25097	4,23374	4,24146	4,20932	4,19597
Hotel	4,33	4,08938	4,08026	4,162	4,11360	4,09782	4,05384	4,03429	4,01939
Zelda	3,60	3,55991	3,55482	3,549	3,54700	3,51389	3,53730	3,52657	3,50442
Średnia	3,822	3,70945	3,69929	3,725	3,70390	3,67197	3,65283	3,63254	3,60422

6.7. Analiza złożoności obliczeniowej

W badaniach praktycznych mamy do czynienia z ograniczonymi zakresami wartości parametrów, np. rząd predykcji $r \leq 50$. Drugim równie istotnym parametrem jest liczba N pikseli obrazu, która jest dużo większa od wartości r, z tego powodu teoretyczna klasyfikacja złożoności pod względem stopnia wielomianu zmiennej r często nie prowadzi do rozsądnych wniosków analizy czasu kodowania. Dlatego w tym podrozdziale zostaną opisane zależności czasowe wynikające z rzeczywistych pomiarów implementacji poszczególnych metod.

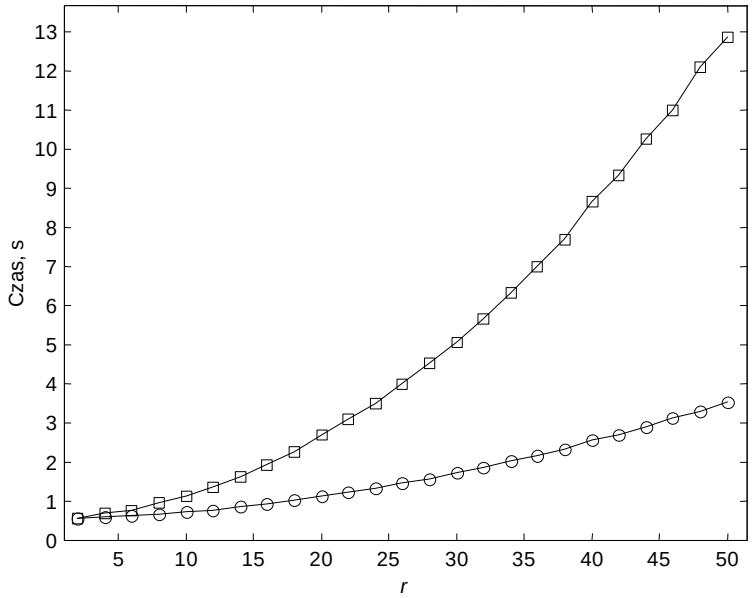
Tab. 6.3. Porównanie średnich bitowych dla metod adaptacyjnych o wysokiej złożoności obliczeniowej

Obrazy	RLS	RLS_+	RLS--NLMS	OLS	OLS--NMLS	WLS (W = 14, r = 18)	AVE--WLS	AVE--WLS1--NLMS+	AVE--WLS2--NLMS+
Aerial	4,59259	4,54737	4,54358	4,54675	4,53278	4,49519	4,43350	4,42222	4,46286
Airfield	4,86728	4,83778	4,83584	4,87426	4,85970	4,83133	4,82420	4,81799	4,83484
Airplane	3,63761	3,60119	3,60226	3,61792	3,61018	3,60027	3,59117	3,58464	3,59538
Baboon	5,76030	5,74580	5,74191	5,67781	5,66262	5,65852	5,64520	5,62794	5,65289
Barbara	4,27758	4,20556	4,15353	4,09783	4,01572	3,99181	3,96945	3,85446	3,84362
Boat	4,47468	4,42259	4,41608	4,44153	4,41212	4,38587	4,37533	4,35211	4,36050
Bridge	3,39543	3,37654	3,38240	3,38208	3,36845	3,35565	3,32985	3,33465	3,35110
Couple	4,12738	4,08686	4,08433	4,07590	4,07108	4,04308	4,02349	4,02063	4,03265
Crowd	3,65636	3,61261	3,61147	3,62790	3,60834	3,56423	3,53243	3,52158	3,53934
Elaine	4,37440	4,34625	4,29670	4,56025	4,30859	4,51941	4,45826	4,23453	4,24530
Finger	5,16906	5,16895	5,16366	5,18627	5,16470	5,17471	5,18117	5,15068	5,18192
Frog	5,01452	5,00262	5,00123	4,99801	4,99507	4,98637	4,97999	4,97787	4,98135
Goldhill	4,48847	4,47317	4,46398	4,48298	4,45806	4,45525	4,43881	4,41642	4,42596
Harbour	4,32497	4,28470	4,28046	4,28040	4,26979	4,21889	4,19584	4,18583	4,20328
Lax	5,56748	5,53833	5,53498	5,53996	5,53712	5,51143	5,50365	5,49716	5,51360
Lennagrey	4,00357	3,96113	3,95530	3,90633	3,90498	3,89848	3,87031	3,87060	3,87504
$Lena_{TMW}$	4,38531	4,34658	4,34098	4,29582	4,29451	4,28695	4,26081	4,26085	4,26799
Man	4,32268	4,27751	4,27704	4,24382	4,24023	4,19013	4,16076	4,16170	4,17187
Peppers	4,28610	4,23821	4,20868	4,23878	4,20462	4,20771	4,19357	4,14323	4,15054
Sailboat	4,47127	4,42867	4,42225	4,51274	4,44783	4,46626	4,45428	4,40392	4,42114
Seismic	2,09696	2,10781	2,05942	2,49323	2,06918	2,26694	2,40345	2,06462	2,05140
Shapes	1,63300	1,49713	1,59990	1,50326	1,55516	1,48046	1,22625	1,27879	1,32149
Tank	3,79409	3,78318	3,78639	3,78978	3,78449	3,77469	3,76426	3,76846	3,77899
Truck	4,03352	4,02525	4,02849	4,04068	4,03496	4,02896	4,02040	4,02332	4,03389
Woman1	3,90237	3,86298	3,85868	3,77962	3,77720	3,75464	3,73325	3,73539	3,74192
Woman2	3,04668	3,03466	3,03358	3,00795	2,99376	2,99169	2,97579	2,97589	2,97345
Balloon	2,72238	2,66175	2,63363	2,68898	2,62515	2,65755	2,63483	2,57008	2,57512
Barb	4,11039	4,01514	4,00175	3,90540	3,87107	3,79572	3,77526	3,74421	3,72764
Barb2	4,39051	4,30685	4,30030	4,28548	4,26364	4,21701	4,19244	4,16911	4,16265
Board	3,42795	3,34925	3,34077	3,37364	3,34341	3,31780	3,30154	3,26955	3,27922
Boats	3,66830	3,60894	3,60168	3,61552	3,58244	3,56591	3,55182	3,52415	3,53261
Girl	3,58771	3,54189	3,52923	3,55453	3,51627	3,48891	3,46675	3,44106	3,46419
Gold	4,27388	4,25190	4,25119	4,25097	4,23363	4,24146	4,20932	4,19597	4,20340
Hotel	4,16591	4,08938	4,08026	4,11360	4,09790	4,05384	4,03429	4,01939	4,02235
Zelda	3,60112	3,55991	3,55482	3,54700	3,51376	3,53730	3,52657	3,50442	3,50155
Bridge256	5,59435	5,59146	5,58870	5,58138	5,57802	5,55884	5,53481	5,53262	5,55450
Camera	4,15608	4,09428	4,09128	4,08351	4,07672	4,00397	3,98462	3,97865	3,98540
Couple256	3,55466	3,52721	3,52756	3,49568	3,49078	3,45171	3,39153	3,39426	3,41743
Earth	2,85475	2,85950	2,85878	2,83740	2,82106	2,82318	2,80005	2,80434	2,81459
Elif	2,68684	2,66173	2,62686	2,81180	2,63326	2,71356	2,74092	2,55397	2,56293
Noisesquare	5,23596	5,23657	5,22934	5,29633	5,26511	5,28156	5,26733	5,22803	5,23444
Omaha	6,09674	6,08286	6,08211	6,07854	6,07855	6,04689	6,04573	6,04683	6,10881
Sena	2,88367	2,86165	2,81906	2,97829	2,82545	2,89403	2,92918	2,74777	2,74855
Sensin	3,14221	3,13383	3,08931	3,27092	3,08870	3,16788	3,22473	2,99467	3,00052
Sinan	2,94273	2,92041	2,88672	3,04655	2,89478	2,95416	2,98889	2,80855	2,81200
Średnia	4,01777	3,98151	3,97281	4,00039	3,95513	3,95356	3,93658	3,89362	3,90489

Badania zrealizowano za pomocą własnego oprogramowania w języku ANSI C (bez szczególnych optymalizacji kodu), wykorzystując procesor Pentium4 2,8 GHz.

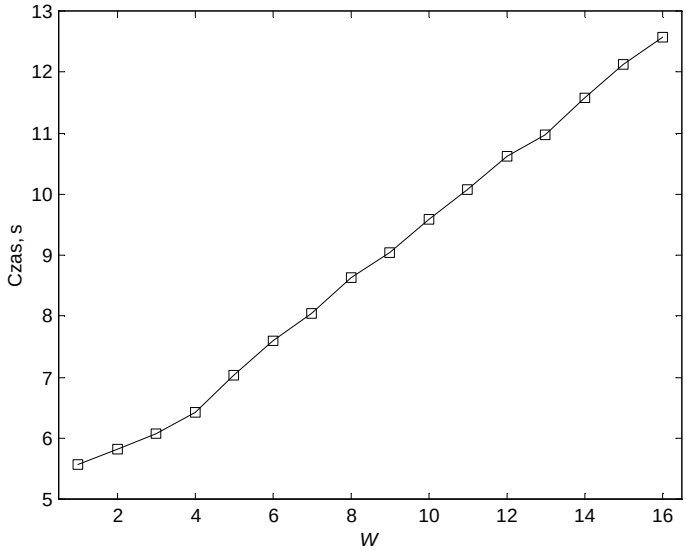
Metoda z predykcją statyczną (patrz podrozdział 4.4) wykorzystuje cztery predyktory przełączane kontekstowo, które są wyznaczane przed etapem kodowania i wymagają przesłania współczynników w nagłówku pliku. Złożoność $O(r^3)$ rozwiązania zaledwie czterech układów równań z r niewiadomymi jest pomijalnie mała w odniesieniu do r^2 operacji przygotowania elementów macierzy **R**, na której są wykonywane obliczenia na podstawie danych z całego obrazu (np. dla obrazu Lennagrey są to $N = 262144$ piksele). Widać zatem, że należy w wielu sytuacjach uwzględniać złożoność przede wszystkim operacji wyznaczającej macierz **R**. Dla wersji statycznej metody obliczania współczynników predykcji jest to $O(N \cdot r^2)$, gdzie N jest znacznie większe od praktycznie stosowanych wartości r^2.

W przypadku metody RLS dla każdego kodowanego piksela jest wykonywana adaptacja współczynników o złożoności $O(r^2)$. Są jednak trzy takie pętle, co sprawia, iż ogólny czas kodowania jest dłuższy niż metodą statycznej predykcji. Rysunek 6.4 przedstawia zależność czasu od rzędu predykcji dla obu metod. Okręgi oznaczają czas dla metody statycznej, a kwadraty dla RLS.



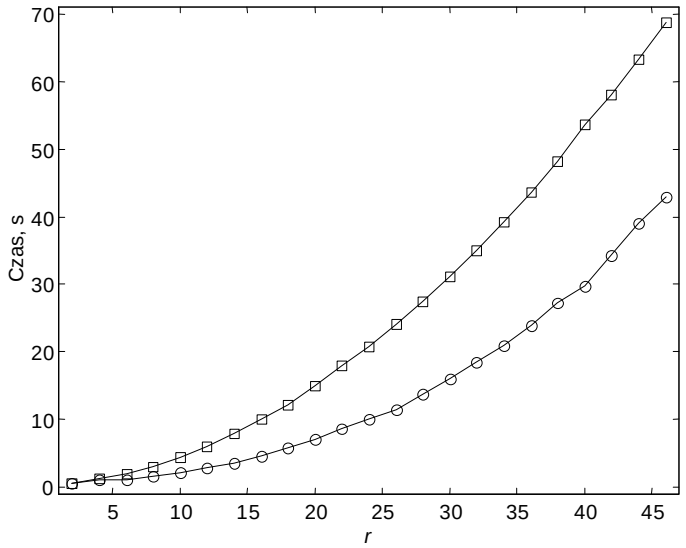
Rys. 6.4. Zależność czasu od rzędu predykcji. Okręgi oznaczają czas dla metody statycznej, a kwadraty dla RLS

Czas kodowania metodą OLS zależy od wysokości W okna danych treningowych Q. Dlatego dokonano dwóch analiz czasowych. Pierwsze badanie dotyczyło zależności czasowej od parametru W przy rzędzie predykcji $r = 14$ – patrz rys. 6.5. Dzięki adaptacyjnej metodzie okna przesuwnego (usuwanie jednej kolumny po lewej stronie i dodawanie jednej kolumny po prawej stronie obszaru Q [140]) udało się uzyskać liniową zależność czasu od wielkości boku W, choć liczba pikseli w oknie jest funkcją kwadratową i wynosi $2W(W + 1)$.



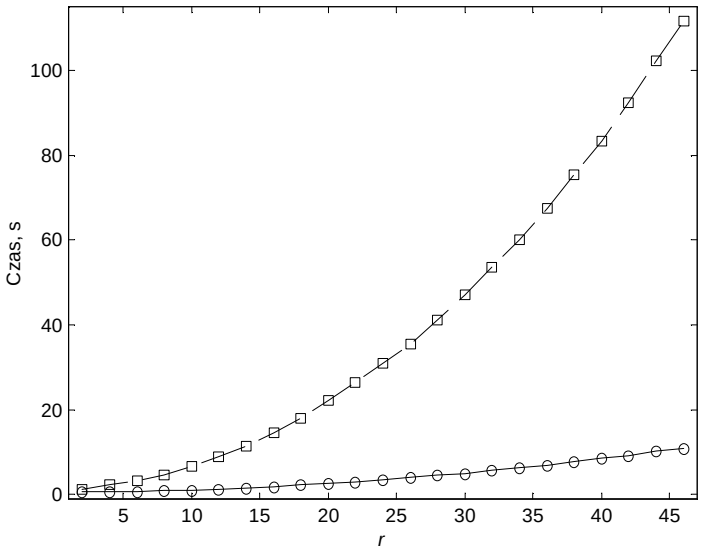
Rys. 6.5. Zależność czasu od wielkości W okna danych treningowych dla metody OLS

Do dalszych badań wybrano wartość $W = 14$, mierząc zależność czasu od rzędu predykcji. Metoda OLS wymaga rozwiązywania układu równań z r niewiadomymi dla każdego piksela, co oznacza złożoność $O(r^3)$. Oprócz tego dla każdego piksela jest też wymagane wyznaczenie elementów macierzy **R**, które dzięki stosowaniu okna przesuwnego cechuje złożoność $O(W \cdot r^2)$. Zmierzono czas obu tych operacji, wyniki umieszczono na rys. 6.6. Kwadraty oznaczają czas przygotowania elementów macierzy, a okręgi czas rozwiązywania układów równań. W metodzie OLS średni czas przygotowania macierzy stanowi około 64,7% całkowitego czasu wyznaczania wektorów **B**.



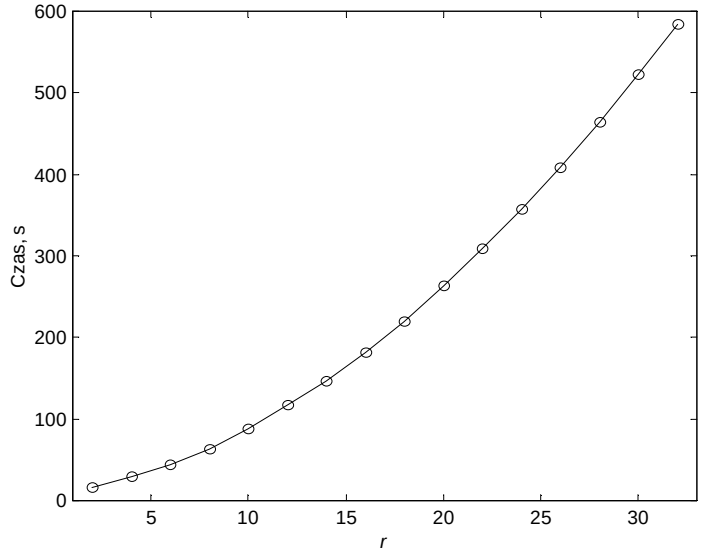
Rys. 6.6. Czas przygotowania macierzy (punkty pomiarowe oznaczone kwadratami) oraz czas rozwiązania układów równań (punkty pomiarowe oznaczone okręgami) metodą OLS przy $W = 14$

Porównanie czasów kodowania metodami OLS (linia przerywana, punkty pomiarowe w postaci kwadratów) i RLS (linia ciągła, punkty pomiarowe w postaci okręgów) przedstawia rys. 6.7.

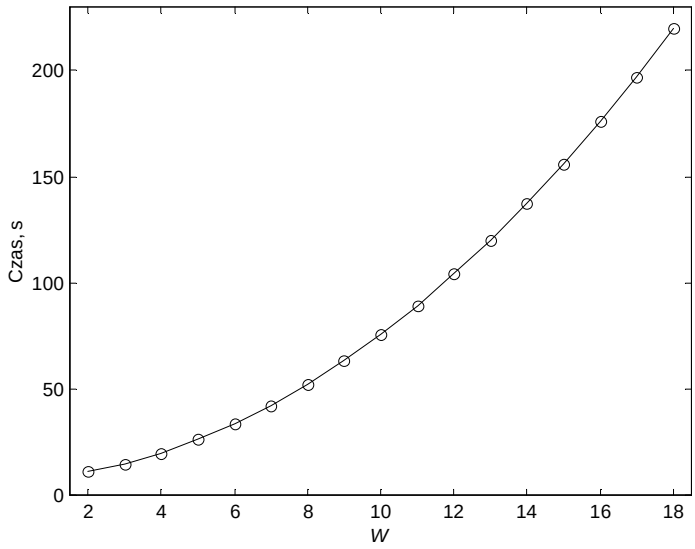


Rys. 6.7. Zależność czasu od rzędu predykcji dla metody RLS (linia ciągła) i metody OLS (linia przerywana)

Metoda WLS wymaga dla każdego kodowanego piksela rozwiązywania układu równań (złożoność $O(r^3)$) z r niewiadomymi. Oprócz tego dla każdego piksela jest też wymagane wyznaczenie elementów macierzy **R**, które cechuje złożoność $O(W^2 \cdot r^2)$. Średni czas przygotowania macierzy **R** metodą WLS jest zdecydowanie dłuższy niż czas rozwiązania układu równań i stanowi około 96,1% całkowitego czasu obliczeń metodą WLS. Wpływ na to mają zarówno rząd predykcji r, jak i wielkość okna treningowego Q. Zależność czasu kodowania metodą WLS od rzędu predykcji (przy $W = 14$) przedstawia rys. 6.8. Natomiast na rys. 6.9 przedstawiono pomiar czasu kodowania metodą WLS w zależności od wartości W przy rzędzie $r = 14$.

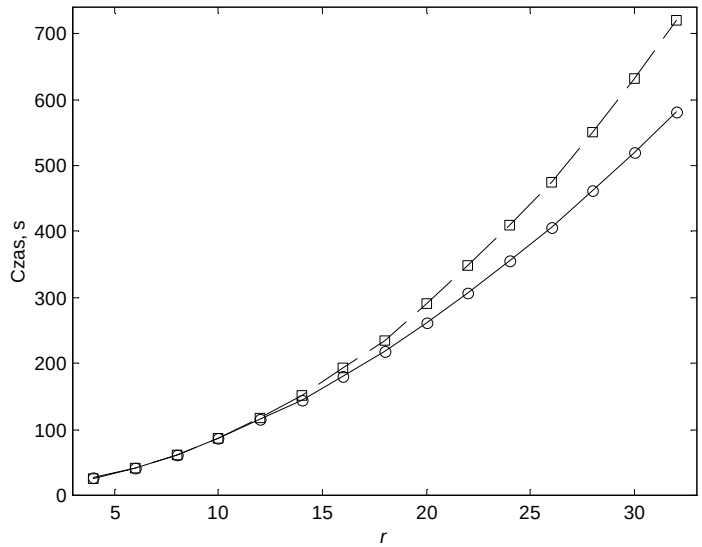


Rys. 6.8. Zależność czasu od rzędu predykcji dla metody WLS przy $W = 14$



Rys. 6.9. Zależność czasu od wielkości W okna danych treningowych dla metody WLS przy $r = 14$

W metodzie AVE-WLS przy $W = 14$ współczynniki modelu predykcji $\mathbf{B}_{ave}$ rzędu r są średnią arytmetyczną współczynników dla wszystkich kolejnych rzędów od czwartego do $r_{max} = 28$ (dla obrazów o wymiarach 512×512). Dla małych wartości r_{max} czas kodowania metodą AVE-WLS jest niewiele dłuższy niż w przypadku podstawowej metody WLS, np. przy $r_{max} = 18$ czas wydłuża się o zaledwie 7%. Porównanie czasów kodowania metodą WLS (linia ciągła, punkty pomiarowe w postaci okręgów) oraz AVE-WLS (linia przerywana, punkty pomiarowe w postaci kwadratów) w zależności od rzędu predykcji przedstawia rys. 6.10.



Rys. 6.10. Zależność czasu od rzędu predykcji dla metody WLS (linia ciągła) oraz AVE-WLS (linia przerywana) przy $W = 14$

7. Zastosowanie sieci neuronowych

7.1. Adaptacyjna sieć neuronowa

Metody adaptacyjne opisane w dwóch poprzednich rozdziałach charakteryzowały się wspólną cechą, czyli adaptacyjnym doborem współczynników predykcji b_j, dzięki któremu po podstawieniu do wzoru (1.2) uzyskiwano wartość przewidywaną $\hat{x}$. Wzór ten jest jednak kombinacją liniową, a metody te są określane w literaturze mianem nieliniowych zazwyczaj ze względu na możliwość przełączania modeli i złożony, nieliniowy charakter adaptacyjnego wyznaczania współczynników. W przypadku sieci neuronowych nie posługujemy się wzorem (1.2), lecz najczęściej różnymi nieliniowymi schematami wyznaczania wartości przewidywanej.

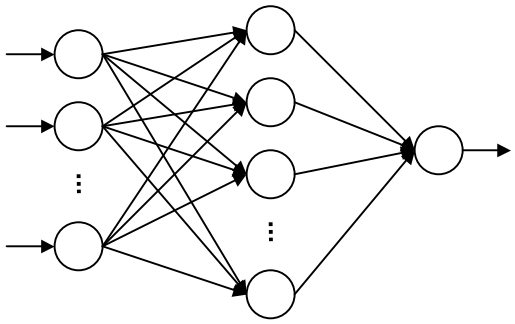
Sieci neuronowe mają szerokie zastosowanie w wielu dziedzinach, pozwalając na uzyskiwanie bardzo dobrych rezultatów przy minimalizacji (maksymalizacji) funkcji celu. Przegląd zastosowań sieci neuronowych do kompresji obrazów przedstawiono w pracach [32, 49]. Opisy tam zawarte odnoszą się głównie do kodowania stratnego. Istnieje niewiele publikacji bezpośrednio dotyczących użycia sieci neuronowych do bezstratnej kompresji obrazów. Stałe nauczenie sieci na podstawie pewnej bazy obrazów nie gwarantuje wysokiej efektywności w przypadku kodowania nowych obrazów. Natomiast metoda statyczna [44, 94], w której wykorzystano uczenie sieci na podstawie kodowanego obrazu, ma poważną wadę, gdyż wymaga przesłania do dekodera bardzo dużej liczby wag sieci. Dlatego w praktycznych zastosowaniach należy uwzględnić metodę adaptacyjną, co zostanie omówione w kolejnych podrozdziałach tego rozdziału. Jednak nie każde podejście adaptacyjne prowadzi do uzyskania wysokiej efektywności, o czym można przekonać się na podstawie wyników metody stosującej komórkowe sieci neuronowe CNN (ang. *Cellular Neural Network*), którą wykorzystano w wielorozdzielczej bezstratnej metodzie kompresji obrazów [109].

W pracach [54, 73, 75] zastosowano adaptacyjną sieć neuronową AdNN (ang. *Adaptive Neural Network*), której konstrukcję oparto na wielowarstwowej sieci perceptronowej MLP (ang. *Multilayer Perceprton Network*). Cecha adaptacyjności oznacza symetryczność czasową kodowania i dekodowania. Sieć jest uczona indywidualnie dla każdego kodowanego piksela na podstawie cech jego otoczenia Q, co prowadzi do wysokiej złożoności obliczeniowej oraz długiego czasu kodowania i dekodowania obrazu. W następnym podrozdziale zostanie przeanalizowany wpływ poszczególnych parametrów sieci na efektywność i czas kompresji.

Model sieci zaproponowany w tej pracy składa się z trzech warstw, co przedstawiono na rys. 7.1. Warstwa wejściowa to zbiór pikseli najbliższego sąsiedztwa kodowanego piksela $P(0)$, zatem ich liczbę możemy określić parametrem r, podobnie jak rząd predykcji (patrz rys. 1.1). Dodatkowo możemy wprowadzić składową stałą równą 1 jako ostatni z neuronów wejściowych, co zasugerowano w pracy [94], wówczas wejścia możemy oznaczyć jako $P(1)$, $P(2)$, ... , $P(r-1)$, 1. Warstwa wyjściowa składa się z jednego neuronu, a warstwa ukryta z n_h neuronów. W procesie inicjalizacji ustala się losowe (lecz identyczne w koderze i dekoderze)

wagi sieci jako liczby z przedziału od 0 do 1. Następnie dla każdego kodowanego piksela $P(0)$ określa się jego obszar treningowy Q, który jest także nazywany obszarem uczącym. Kształt okna jest identyczny jak dla metod OLS i WLS opisanych w rozdziale 6. Na rysunku 6.1 przedstawiono przykład takiego okna o wysokości $W = 3$.

Jedna epoka uczenia sieci polega na przejściu piksela $P_{\text{off}}(0)$ po całym obszarze uczącym Q, podobnie jak to jest przy wyznaczaniu macierzy **R** w metodach OLS i WLS (patrz rys. 6.2). W każdej kolejnej pozycji piksela $P_{\text{off}}(0)$ następuje adaptacja wag sieci. Proces uczenia kończy się po wykonaniu odpowiednio dużej liczby t_e epok (w pracy [54] $t_e = 20$, w pracach [73, 75] $t_e = 25$). Wagi tak ustalone służą do wyznaczenia aktualnej wartości przewidywanej, a także jako wagi początkowe w kolejnym procesie uczenia przy kodowaniu następnego piksela.



Rys. 7.1. Trójwarstwowa sieć neuronowa

Wykorzystując sieci neuronowe, należy pamiętać o dokonaniu normalizacji wartości wejściowych, czyli przeskalowaniu wartości $P(j)$ pikseli sąsiedztwa z pierwotnego przedziału od 0 do 255 na przedział od –0,5 do 0,5. Proces uczenia i kodowania jest następujący [54, 73]: Wartości wyjściowe $S_h(i)$ każdego i-tego neuronu warstwy ukrytej wyznaczamy, wykorzystując funkcję sigmoidalną:

$$S_h(i) = \frac{1}{1+e^{-S_{\text{in}}(i)}} \text{ dla każdego } i = \{1, \ldots, n_h\}, \tag{7.1}$$

gdzie $S_{\text{in}}(i)$ jest średnią ważoną wartości wyjść neuronów z warstwy wejściowej:

$$S_{\text{in}}(i) = \sum_{j=1}^{r} w_{\text{in}}^{(i)}(j) \cdot \left(\frac{P(j)}{255} - 0{,}5 \right) \text{ dla każdego } i = \{1, \ldots, n_h\}, \tag{7.2}$$

a $w_{\text{in}}^{(i)}(j)$ oznacza wagę wejściową i-tego neuronu warstwy ukrytej połączoną z j-tym neuronem warstwy wejściowej (pikselem $P(j)$ lub dla $j = r$ składową stałą równą 1). Kolejnym krokiem jest wyznaczenie wartości warstwy wyjściowej sieci:

$$y_{\text{out}} = \frac{1}{1+e^{-S_{\text{out}}}}, \tag{7.3}$$

gdzie S_{out} jest średnią ważoną wartości $S_h(i)$ wyjść neuronów z warstwy ukrytej:

$$S_{\text{out}} = \sum_{i=1}^{n_h} w_{\text{out}}(i) \cdot S_h(i), \tag{7.4}$$

a $w_{\text{out}}(i)$ oznacza wagę wyjściową i-tego neuronu warstwy ukrytej. Po każdym takim procesie wyznaczenia y_{out} dla kolejnych pikseli z obszaru uczącego następuje adaptacja wag sieci. W tym celu wyznacza się sygnał błędu warstwy wyjściowej:

$$\delta_{\text{out}} = y_{\text{out}} \cdot (1 - y_{\text{out}}) \cdot \left(\frac{P(0)}{255} - y_{\text{out}} \right), \tag{7.5}$$

a także sygnały błędów dla każdego neuronu warstwy ukrytej:

$$\delta_h(i) = S_h(i) \cdot (1 - S_h(i)) \cdot w_{\text{out}}(i) \cdot \delta_{\text{out}}. \tag{7.6}$$

Dla uproszczenia oznaczeń i uniknięcia nadmiernej liczby indeksów w kolejnych wzorach opisujących adaptację poszczególnych parametrów pominięto oznaczenia chwil (lewa strona równania odnosi się bowiem do kolejnej (n + 1) chwili). Na podstawie tych błędów dla każdego $i = \{1, \ldots, n_h\}$ oraz $j = \{1, \ldots, r\}$ dokonujemy aktualizacji przyrostów wag wejściowych w warstwie ukrytej:

$$\Delta w_{\text{in}}^{(i)}(j) := \eta \cdot \delta_h(i) \cdot \left(\frac{P(i)}{255} - 0{,}5 \right) + \alpha \cdot \Delta w_{\text{in}}^{(i)}(j) \tag{7.7}$$

oraz przyrostów wag wyjściowych w warstwie ukrytej:

$$\Delta w_{\text{out}}(i) := \eta \cdot \delta_{\text{out}} \cdot S_h(i) + \alpha \cdot \Delta w_{\text{out}}(i), \tag{7.8}$$

a następnie aktualizujemy wagi wejściowe:

$$w_{\text{in}}^{(i)}(j) := w_{\text{in}}^{(i)}(j) + \Delta w_{\text{in}}^{(i)}(j) \tag{7.9}$$

oraz wagi wyjściowe neuronów warstwy ukrytej:

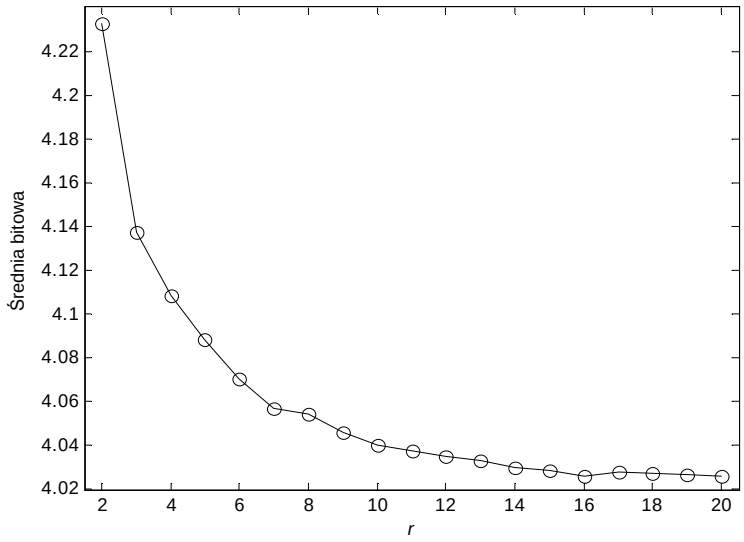
$$w_{\text{out}}(i) := w_{\text{out}}(i) + \Delta w_{\text{out}}(i). \tag{7.10}$$

Autorzy pracy [73] zasugerowali, aby przez pierwsze trzy epoki wartość szybkości uczenia η wynosiła 0,9, a parametr momentum $\alpha = 0$, natomiast w kolejnych epokach $\eta = 0{,}1$ oraz $\alpha = 0{,}9$. Po nauczeniu (wykonaniu np. 20 epok) wyjściowa wartość y_{out}, otrzymana z sieci, należąca do przedziału od 0 do 1, powinna zostać pomnożona przez współczynnik skalujący 255, aby uzyskać odpowiednią wartość przewidywaną piksela $P(0)$.

7.2. Analiza parametrów sieci AdNN

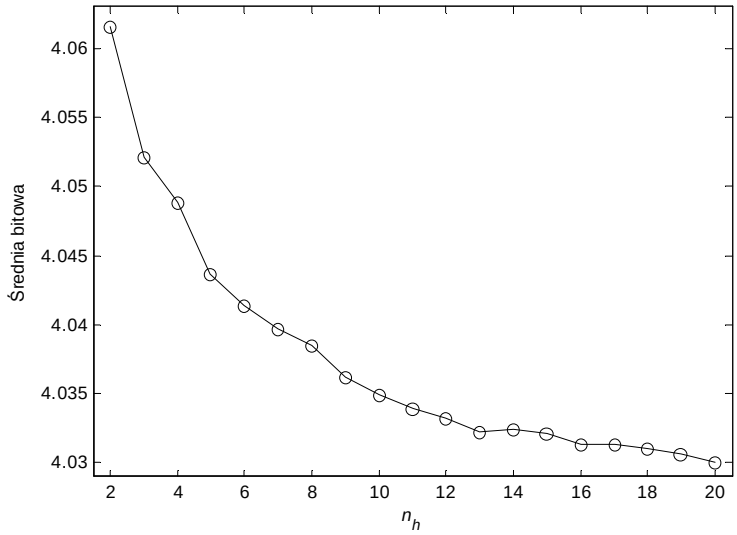
W pracy [73] autorzy zaproponowali $r = 12$, $n_h = 10$, $W = 4$ oraz $t_e = 25$ epok. Aby ustalić rozsądne wartości parametrów wykorzystywanych w sieci AdNN, przeanalizowano w tym podrozdziale wpływ każdego parametru na średnią bitową zestawu 45 obrazów testowych oraz na czas kodowania tą metodą obrazu Lennagrey.

Pierwszym testowanym parametrem był rząd *r*, czyli liczba neuronów w warstwie wejściowej. Aby uzyskać rozsądne wyniki w szerokim zakresie parametru *r*, wartość *W* zwiększono do 5, a liczbę epok zmniejszono do 10. Na rysunku 7.2 przedstawiono zależność średniej bitowej od rzędu *r*. Istotny spadek średniej uzyskujemy do wartości $r = 14$. Zależność czasu kodowania od rzędu *r* jest funkcją liniową postaci $14{,}75r + 193{,}91$ s, co już przy $r = 14$ daje ponad 400 s.

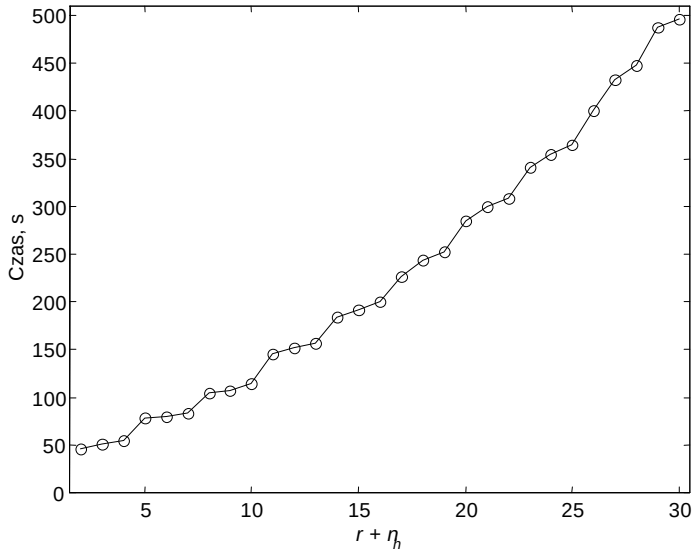


Rys. 7.2. Zależność średniej bitowej od liczby *r* neuronów w warstwie wejściowej

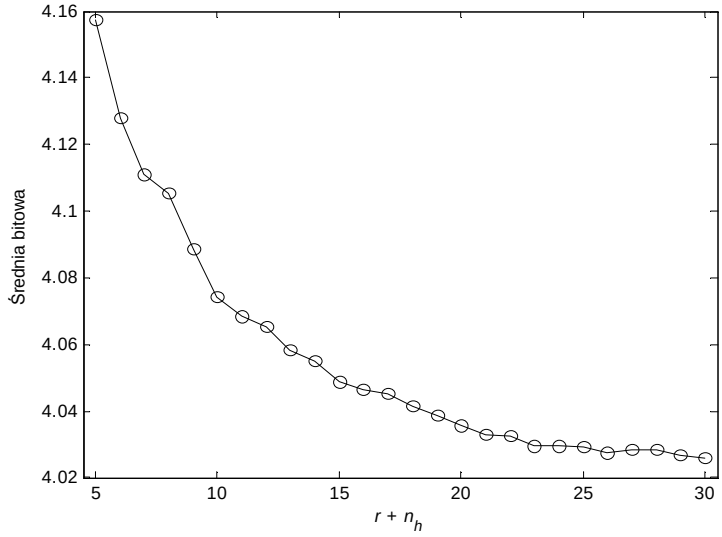
Kolejnym testowanym parametrem była liczba n_h neuronów w warstwie ukrytej. Pomiary przeprowadzono dla $r = 12$, $W = 5$ i $t_e = 10$. Także i w tym przypadku uzyskano wyrażoną w sekundach liniową zależność czasu od liczby n_h w postaci funkcji $32{,}44n_h + 45{,}97$. Widać jednak, że wzrost liczby neuronów w warstwie ukrytej ma większy wpływ na wydłużanie się czasu kodowania niż w przypadku neuronów warstwy wejściowej. Jednocześnie można zauważyć znacznie mniejszą zmienność średniej bitowej przy wzroście n_h, co pokazano na rys. 7.3. Z tego względu kolejne badanie wykonano przy łącznym wzroście liczby neuronów w obu warstwach, przy czym po dołożeniu jednego neuronu w warstwie ukrytej, następne trzy neurony były umieszczane kolejno w warstwie wejściowej. Zależność czasu nie jest już liniowa, co pokazano na rys. 7.4, ale propozycja ta pozwala bardziej elastycznie dokonać wyboru obu parametrów *r* oraz n_h. Rysunek 7.5 przedstawia zależność średniej bitowej od liczby neuronów w warstwach wejściowej i ukrytej.



Rys. 7.3. Zależność średniej bitowej od liczby n_h neuronów w warstwie ukrytej

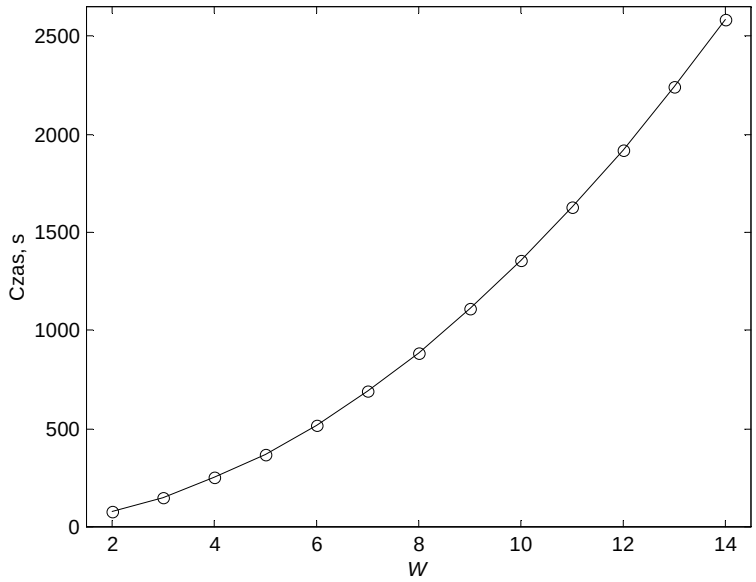


Rys. 7.4. Zależność czasu od liczby $r + n_h$ neuronów w warstwach wejściowej i ukrytej



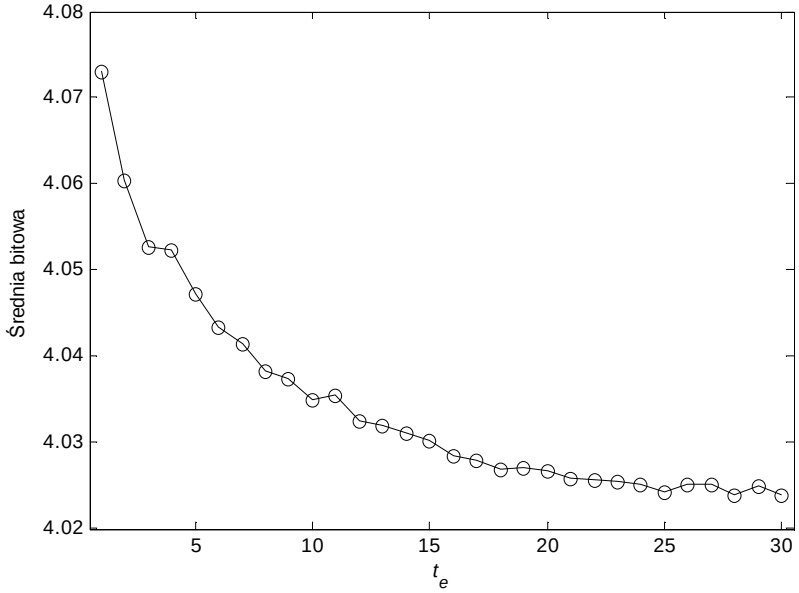
Rys. 7.5. Zależność średniej bitowej od liczby $r + n_h$ neuronów w warstwach wejściowej i ukrytej

Kolejne badanie dotyczyło wpływu wysokości W okna treningowego Q, przeprowadzono je dla $t_e = 10$ epok i parametrów $r = 12$, $n_h = 10$. Zauważalny spadek średniej bitowej otrzymano przy $W \leq 7$, dla większego okna treningowego wartość spadku była niewielka, a powyżej $W = 12$ otrzymano wzrost średniej. Kwadratową zależność wielkości obszaru Q od W potwierdza rys. 7.6 przedstawiający czas kodowania w funkcji parametru W. Przy $W = 7$ czas kodowania wyniósł 693 s.



Rys. 7.6. Zależność czasu od wysokości W okna treningowego Q

Ostatnim testowanym parametrem była liczba t_e epok treningowych przy parametrach $r = 12$, $n_h = 10$, $W = 5$. Uzyskano tu liniową funkcję czasu kodowania wynoszącą $37{,}04t_e + 2{,}34$ s. Spadek średniej bitowej można zaobserwować dla $t_e \leq 25$ (patrz rys. 7.7).



Rys. 7.7. Zależność średniej bitowej od liczby t_e epok treningowych

7.3. Zwiększanie wydajności sieci AdNN

Na podstawie analiz przedstawionych w poprzednim podrozdziale można wywnioskować, że dla parametrów proponowanych w pracy [73] (przy czasie kodowania dłuższym nawet niż w przypadku AVE-WLS) nie można uzyskać tak wysokiej efektywności kompresji, jaką otrzymano dla metod RLS_+, OLS oraz WLS. Należało zatem sprawdzić, czy kosztem zwiększenia czasu obliczeń jest możliwy dalszy wzrost efektywności. W tym celu zaproponowano następujące parametry sieci: $r = 24$, $n_h = 12$, $W = 7$, $t_e = 25$. Średnia z 45 obrazów testowych dla tej metody, określanej dalej jako $AdNN_b$, wyniosła 4,00750 bitu, co jest rezultatem zbliżonym do metody OLS, lecz czas kodowania obrazu Lennagrey, wynoszący 3000 s, pokazuje, że nie można metody AdNN zaliczyć do przodujących technik adaptacyjnych.

Po tych analizach należało podjąć próbę autorskiego udoskonalenia tej metody. Pierwszym pomysłem było wprowadzenie podziału na trzy konteksty główne, co zostało opisane w podrozdziale 3.2.1. W takim przypadku oznacza to zbudowanie trzech niezależnych sieci neuronowych, które są wykorzystywane i uczone w zależności od tego, który kontekst jest aktualnie wyznaczony. Podobną, choć bardziej rozbudowaną propozycję wykorzystującą kwantyzację wektorową do wyznaczania kontekstów, zaproponowano w pracy [94]. W takiej sytuacji wymagane jest jednak przekazywanie dekoderowi wektorów będących centroidami skojarzonymi z danym kontekstem, a tego chcemy uniknąć.

Kolejna modyfikacja polega na zmniejszeniu okna treningowego w ostatnich epokach. Możemy wyjść z założenia, że w początkowej fazie uczenia sieć dostraja się do dostatecznie dużego okna ($W = 7$). Pod koniec fazy uczenia możemy zwiększyć istotność najbliżej położonego sąsiedztwa i w epokach 23 i 24 zmniejszamy wartość W do 5, natomiast w 25 epoce do $W = 3$. Pozwala to jednocześnie zmniejszyć czas kodowania do 2781 s.

Ostatnim udoskonaleniem jest użycie dodatkowego bloku NLMS opisanego w podrozdziale 6.5, pozwalającego na dalszą redukcję błędu predykcji, co przekłada się na spadek średniej bitowej.

W tabeli 7.1 zaprezentowano porównanie opracowanej tu metody $AdNN_+$ z innymi popularnymi metodami, z którego wynika, iż udało się uzyskać niższą średnią bitową testu niż dla najlepszych znanych z literatury implementacji metod bezstratnej kompresji obrazów wykorzystujących sieci neuronowe [54, 73]. Stało się to jednak znacznym kosztem złożoności obliczeniowej, a mimo to metoda $AdNN_+$ nie dorównała efektywności innej złożonej metodzie, TMW, choć w dużym stopniu gorszy rezultat wynika ze słabej wydajności kodowania obrazu Shapes, który należy do kategorii sztucznie generowanych. Porównanie wersji podstawowej $AdNN_b$ i udoskonalonej $AdNN_+$ wraz z kolejnymi etapami modyfikacji przedstawia tab. 7.2.

Istnieje potrzeba dalszych poszukiwań zwiększenia efektywności kompresji przy użyciu sieci neuronowych, co zaproponowano w pracy [65], jednak implementacja w niej omówiona nie uwzględnia wielu podstawowych technik poprawy efektywności modelowania, dlatego wyniki zaprezentowane tam przez autorów są porównywalne jedynie z rezultatami otrzymywanymi metodą CALIC.

Tab. 7.1. Porównanie średnich bitowych metod adaptacyjnych o wysokiej złożoności obliczeniowej

Obrazy	CALIC [149]	TMW [73]	AdNN [73]	SWAP [54]	$AdNN_+$
Camera	4,190	4,098	4,13	4,39	4,120
Couple256	3,609	3,446	3,60	3,75	3,561
Noisesquare	5,443	5,542	5,18	5,16	5,201
Airplane	3,743	3,601	4,06	3,58	3,598
Baboon	5,875	5,738	5,77	5,86	5,706
$Lena_{TMW}$	4,475	4,300	4,35	4,35	4,307
Lennagrey	4,102	3,908	3,97	3,95	3,925
Peppers	4,421	4,251	4,27	4,25	4,201
Shapes	1,139	0,740	1,44	1,56	1,708
Balloon	2,825	2,649	2,80	2,49	2,647
Barb	4,413	4,084	4,16	4,12	3,868
Gold	4,394	4,266	4,33	4,30	4,238
Średnia	4,052	3,885	4,005	3,980	3,923

Tab. 7.2. Porównanie średnich bitowych $AdNN_b$ oraz kolejnych udoskonaleń metody $AdNN_+$

Obrazy	$AdNN_b$	3 konteksty	redukcja W	$AdNN_+$
Aerial	4,61911	4,59594	4,58742	4,57757
Airfield	4,87518	4,86495	4,86395	4,86138
Airplane	3,60420	3,59878	3,60078	3,59818
Baboon	5,73662	5,71526	5,71593	5,70550
Barbara	4,06787	4,06837	4,06648	4,00825
Boat	4,43542	4,43413	4,42952	4,41823
Bridge	3,38739	3,39503	3,39484	3,38797
Couple	4,10896	4,10673	4,10463	4,09985
Crowd	3,66223	3,65892	3,64532	3,63804
Elaine	4,39757	4,39524	4,39500	4,32272
Finger	5,22675	5,20843	5,20649	5,19401
Frog	4,99376	4,99387	4,99622	4,99544
Goldhill	4,48096	4,47692	4,47723	4,46246
Harbour	4,35095	4,32864	4,32735	4,31792
Lax	5,59538	5,57897	5,57943	5,57598
Lennagrey	3,92628	3,92773	3,92704	3,92457
$Lena_{TMW}$	4,30695	4,30718	4,30778	4,30703
Man	4,28841	4,28638	4,27914	4,27638
Peppers	4,22362	4,22536	4,22285	4,20117
Sailboat	4,49977	4,46864	4,46951	4,44092
Seismic	2,27558	2,27917	2,28247	2,10981
Shapes	1,81775	1,76983	1,69358	1,70793
Tank	3,79053	3,78895	3,78917	3,78973
Truck	4,04639	4,04059	4,03821	4,03798
Woman1	3,82027	3,83316	3,83398	3,83196
Woman2	3,02030	3,01658	3,01978	3,00945

Tab. 7.2. Porównanie średnich bitowych $AdNN_b$ oraz kolejnych udoskonaleń metody $AdNN_+$ (cd.)

Obrazy	$AdNN_b$	3 konteksty	redukcja W	$AdNN_+$
Balloon	2,67753	2,68646	2,68558	2,64669
Barb	3,88051	3,88455	3,88510	3,86829
Barb2	4,31221	4,29488	4,29209	4,28272
Board	3,34294	3,34387	3,34477	3,33022
Boats	3,62447	3,62497	3,62606	3,60713
Girl	3,55159	3,55128	3,54640	3,52831
Gold	4,25461	4,24895	4,24835	4,23837
Hotel	4,10533	4,10451	4,09603	4,08609
Zelda	3,52772	3,53765	3,53907	3,53046
Bridge256	5,65759	5,64415	5,63512	5,63051
Camera	4,16260	4,13542	4,12849	4,12001
Couple256	3,58792	3,58907	3,58000	3,56114
Earth	2,87056	2,86964	2,86781	2,85753
Elif	2,71182	2,71291	2,71956	2,65234
Noisesquare	5,20923	5,20200	5,20767	5,20091
Omaha	6,25705	6,23528	6,22697	6,22272
Sena	2,90048	2,90509	2,91115	2,83737
Sensin	3,17403	3,19054	3,19592	3,11641
Sinan	2,97127	2,98170	2,98119	2,90800
Średnia	4,00750	4,00237	3,99937	3,97835

8. Adaptacyjny koder arytmetyczny z podziałem kontekstowym

8.1. Długookresowa forma adaptacji rozkładu prawdopodobieństwa

Omówione we wcześniejszych rozdziałach metody modelowania dokonujące dekompozycji danych przygotowują je do etapu kodowania właściwego, zamieniającego błędy predykcji na ciąg bitów, z których powstaje plik wynikowy. Rodzaj kompresji zaproponowany w niniejszej pracy opiera się na zasadzie działania kodu arytmetycznego [34, 93, 98], Szczegółowa implementacja oparta na arytmetyce liczb całkowitych jest opisana w pracy [98]. Wersję tę rozszerzono tu o system adaptacji i przełączania między wieloma rozkładami prawdopodobieństw. Poszczególne etapy rozbudowy omówiono w podrozdziałach tego rozdziału.

Wykorzystując symetryczność rozkładu prawdopodobieństwa błędów predykcji, można kodować ich wartości bezwzględne (bit znaku koduje się osobno – patrz podrozdział 8.4). Dzięki temu mamy do czynienia z rozkładem zbliżonym do jednostronnego rozkładu Laplace'a, co przyspiesza proces adaptacji, a jest to szczególnie ważne przy kodowaniu pierwszych błędów predykcji.

Początkowy rozkład prawdopodobieństwa, a właściwie wektor liczebności $\mathbf{N}_e$ wystąpień wartości i (czyli poszczególnych wartości bezwzględnych błędów predykcji), możemy potraktować jako przybliżenie rozkładu Lapace'a, wykorzystując uproszczony wzór [85]:

$$\mathbf{N}_e(i) = \lfloor f \cdot 0{,}8^i \rfloor + 1, \tag{8.1}$$

gdzie i jest z przedziału od 0 do 255, a f jest wartością wychylenia inicjującego wektor liczebności. Dobre rezultaty uzyskuje się dla $f = 10$. Adaptacja polega na tym, iż po wczytaniu i zakodowaniu każdej kolejnej wartości błędu predykcji $|e| = |e(0)|$, należy uaktualnić wektor liczebności, zwiększając wartość pod indeksem $|e|$ o jeden: $\mathbf{N}_e(|e|) := \mathbf{N}_e(|e|) + 1$. Dodatkowo możemy wprowadzić efekt zapominania, sprzyjający zmniejszaniu tych liczebności, które wśród ostatnio kodowanych wartości nie pojawiły się lub pojawiały się rzadziej niż podczas wcześniejszego etapu kodowania [37]. W tym celu kontrolujemy łączną liczbę wartości zakodowanych do tej pory, a dokładniej wartość licznika, który jest zwiększany o 1 po każdym zakodowaniu liczby $|e|$. Jeśli licznik osiągnie z góry założoną wartość 2^s, wówczas wszystkie elementy wektora liczebności są zmniejszane o połowę z wykorzystaniem przypisania:

$$\mathbf{N}_e(i) \coloneqq \left\lfloor \frac{\mathbf{N}_e(i)}{2} \right\rfloor + 1 \text{ dla każdego } i \text{ od 0 do } e_{\max}. \tag{8.2}$$

Po przeskalowaniu ponownie jest obliczana wartość licznika jako suma wszystkich elementów $\mathbf{N}_e(i)$ wektora liczebności. Dobre rezultaty uzyskuje się dla $s \geq 10$. W proponowanym koderze użyto $s = 13$ dla kontekstów wykorzystywanych przy kodowaniu wartości błędów predykcji (poddanych wcześniej kwantyzacji – patrz podrozdział 8.3) oraz $s = 10$ dla pozostałych kontekstów (wykorzystywanych przy kodowaniu reszt kwantyzacji oraz bitu znaku, omówionych w podrozdziałach 8.3 oraz 8.4).

Stosując ten rodzaj adaptacji, w dość skuteczny sposób można zwiększyć efektywność kompresji dzięki zastosowaniu transformaty Burrowsa–Wheelera w odniesieniu do strumienia błędów predykcji [2], jednak wyjściowy strumień danych jest jednowymiarowy. Oznacza to utratę potencjalnych korzyści, jakie można uzyskać przy projektowaniu dwuwymiarowych zasad wyznacznania kontekstów. Użycie tych zasad pozwala znacznie zwiększyć efektywność kompresji dzięki wprowadzeniu adaptacyjnego kodu arytmetycznego z podziałem kontekstowym.

8.2. Podział kontekstowy w koderze arytmetycznym

Koder adaptacyjny opisany w podrozdziale 8.1 potrafi dostosować się do rozkładu w sensie długookresowym, lecz można wykorzystać jeszcze istnienie krótkookresowych zależności między kolejno kodowanymi danymi, analizując najbliższe sąsiedztwa złożone z pikseli $P(j)$ oraz dwuwymiarowego sygnału błędów predykcji $e(j)$.

Na podstawie cech sąsiednich błędów predykcji można dość dokładnie określić przybliżony typ rozkładu aktualnie kodowanej wartości $|e|$. Wychodząc z takiego założenia, możemy zaprojektować kontekstowy koder arytmetyczny, mający nie jeden, lecz t rozkładów prawdopodobieństw skojarzonych z poszczególnymi numerami konteksów od 0 do $t - 1$. Teoretycznie wraz ze wzrostem liczby kontekstów można oczekiwać zwiększenia efektywności kompresji. Przy założeniu, że początkowo nie znamy tych rozkładów, gdyż ich kształty są budowane metodą adaptacyjną (pozyskiwanie wiedzy na podstawie nadchodzących danych), pojawia się problem rozrzedzenia kontekstów, co przekłada się na zbyt wolną adaptację skojarzonych z nimi rozkładów [93]. Adaptacyjny charakter aktualizacji rozkładów prawdopodobieństwa wymaga szybkiego wyznaczenia przybliżonej docelowej postaci każdego z t rozkładów, dlatego należy ustalić pewien kompromis między liczbą kontekstów a szybkością adaptacji rozkładów. Często wykorzystuje się 8 [141], 16, a nawet 20 kontekstów [28]. Analizę wpływu liczby kontekstów na średnią bitową zaprezentowano w pracy [3].

Istnieje jeszcze druga możliwość, w której zakładamy początkową znajomość przybliżonych rozkładów prawdopodobieństwa w każdym kontekście. W takiej sytuacji można zbudować nawet koder arytmetyczny, który wyznacza indywidualny rozkład prawdopodobieństwa dla każdego kolejno kodowanego piksela. Jest to bardzo złożony problem, wymaga bowiem znalezienia jak najlepszego matematycznego opisu rozkładów pasujących do danego obrazu lub klasy obrazów. Analizę takiego podejścia na podstawie rozkładów Laplace'a, Gaussa, t-Studenta oraz uogólnionego rozkładu Gaussa przeprowadzono w pracy [151]. Wnioski te wykorzystano w publikacji [80], w której użyto uogólnionego rozkładu Gaussa.

Podobne rozwiązanie wykorzystujące rozkład t-Studenta zastosowano też w metodzie TMW [86], w której wprowadzono również zasadę mieszania rozkładów prawdopodobieństw. Natomiast dalsze badania wykazały, iż istotnym zagadnieniem jest zależność sposobu, w jaki projektuje się modele predykcyjne, od rodzaju rozkładu, którym można w przybliżony sposób opisać parametrycznie rzeczywisty rozkład błędów predykcji [115]. Dla modeli predykcyjnych wykorzystujących minimalizację błędu absolutnego (MMAE) dobrze sprawdza się mieszany rozkład Laplace'a.

W opracowanym tu autorskim rozwiązaniu zastosowano podział na 16 kontekstów ze wstępną inicjalizacją rozkładów i adaptacyjną zasadą ich uaktualniania w miarę pojawiania się kolejnych danych na podstawie każdego kodowanego piksela.

Istotnym czynnikiem, wpływającym na efektywność kompresji, jest odpowiedni dobór reguły decyzyjnej, wyznaczającej numer kontekstu. Zaprezentowana tu reguła jest rozwinięciem pomysłu z prac Denga i współautorów [28, 148]. Wartość ω_1 jest wyznaczana z wykorzystaniem błędów $e(j)$, znajdujących się w najbliższym otoczeniu kodowanego błędu $e(0)$, gdzie indeks j określa położenie błędu zgodnie z rys. 1.1:

$$\omega_1 = \max\{2|e(1)|, 2|e(2)|, \frac{9}{8}(|e(3)|+|e(4)|), |e(5)|+|e(10)|, |e(6)|+|e(7)|,$$
$$\frac{13}{8}|e(4)|, \frac{3}{2}|e(3)|, \frac{7}{8}(|e(8)|+|e(9)|), \frac{11}{8}(|e(1)|+|e(2)|)\}. \tag{8.3}$$

Nieco inne rozwiązania zastosowano w pracach [3, 78]. W proponowanym tu rozwiązaniu, podobnie jak w publikacji [78], kolejny parametr ω_2 jest wyznaczany jako średnia wagowa modułów błędów $|e(j)|$ znajdujących w najbliższym otoczeniu:

$$\omega_2 = \frac{1}{\delta}\sum_{j=1}^{m} \bar{d}_j \cdot |e(j)|, \tag{8.4}$$

gdzie δ jest wyznaczane ze wzoru (3.11), a warość $m = 28$. Na podstawie powyższych zależności możemy obliczyć parametr ω_3 jako:

$$\omega_3 = \max\{2\omega_1, 10\omega_2\}, \tag{8.5}$$

co pozwoli potem precyzyjniej określić liczbę, dzięki której, po jej nieliniowej kwantyzacji skalarnej, otrzymamy numer kontekstu. Dodatkowo skuteczność doboru właściwego kontekstu można zwiększyć, korzystając z wartości czterech pikseli najbliższego sąsiedztwa. W tym celu wyznacza się parametr ω_4:

$$\omega_4 = \max\{|P(1)-P(3)|, |P(2)-P(3)|, |P(1)-P(2)|, 1{,}1|P(2)-P(4)|,$$
$$0{,}8|P(1)-P(4)|, 0{,}9|P(3)-P(4)|\}. \tag{8.6}$$

Na podstawie parametrów ω_3 oraz ω_4 możemy obliczyć końcową wartość ω:

$$\omega = \omega_3 + 0{,}48\omega_4 . \tag{8.7}$$

Wartość ω poddajemy kwantyzacji z użyciem $t - 1$ progów $\mathbf{T}_h(i)$, aby uzyskać numer kontekstu arytmetycznego wskazującego na aktualny rozkład prawdopodobieństwa. Na przykład dla $t = 16$ progi dobrano następująco: $\mathbf{T}_h = \{3, 8, 14, 20, 27, 34, 43, 55, 66, 80, 100, 120, 150, 180, 240\}$. Oznacza to, że mamy do czynienia z modelem kodu arytmetycznego pierwszego rzędu. W modelu tym rozkłady prawdopodobieństw są dobierane warunkowo w zależności od parametru ω, choć sama wartość ω jest zależna od wielu pikseli i błędów predykcji najbliższego sąsiedztwa.

8.3. Kwantyzacja błędów predykcji

Aby zwiększyć szybkość adaptacji rozkładów skojarzonych z poszczególnymi kontekstami, często stosuje się kwantyzację wartości $|e|$ [26]. Polega to na zrzutowaniu przedziału liczb $|e|$ od 0 do 255 na mniejszy zakres liczb k, np. od 0 do 17. Idea taka jest stosowana w wielu metodach kodowania, np. w standardzie JPEG [98]. Technika kwantyzacji z bezstratnym kodowaniem błędów predykcji ma za zadanie podzielić wartość $|e|$ na dwie liczby, które są kodowane z użyciem odrębnych rozkładów (czasem przyjmuje się, iż wartość reszty powstała po kwantyzacji jest zapisywana jako ciąg bitów bez kompresji). Poniżej przedstawiono etapy kwantyzacji i kodowania.

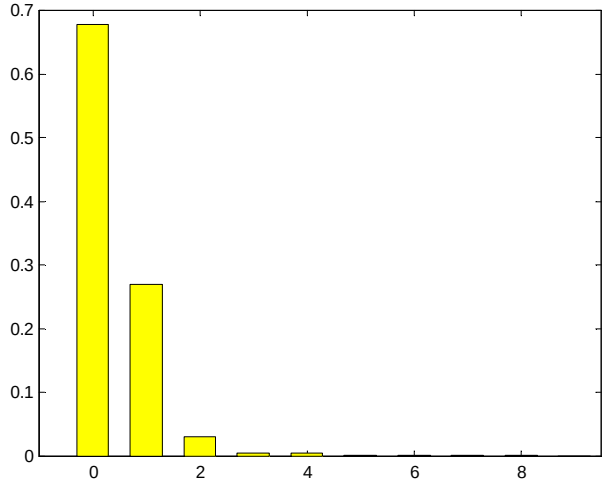
Na początku jest wyznaczana wartość k z zależności $\mathbf{T}(k) \le |e| < \mathbf{T}(k + 1)$, na podstawie danego zbioru kolejnych progów kwantyzacji $\mathbf{T} = \{0, 1, 2, 3, 4, 5, 6, 7, 8, 10, 12, 14, 16, 20, 24, 32, 64, 128\}$. Skwantyzowana wartość, oznaczona jako k, jest wysyłana do kodera arytmetycznego i kodowana po skojarzeniu jej z odpowiednim rozkładem prawdopodobieństwa, przypisanym do kontekstu wyznaczonego według zasady opisanej w podrozdziale 8.2. Wartość reszty powstała po kwantyzacji $e_q = |e| - \mathbf{T}(k)$ jest traktowana jako $\mathbf{q}(k)$-bitowa liczba, gdzie $\mathbf{q}(k)$ odczytujemy jako k-tą wartość z wektora liczb $\mathbf{q} = \{0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 2, 2, 3, 5, 6, 7\}$. Jeśli $\mathbf{q}(k) > 0$, to wartość e_q może być wysłana na wyjście w postaci $\mathbf{q}(k)$ bitów lub kodowana za pomocą jednego z siedmiu adaptacyjnych koderów arytmetycznych (o numerze $\mathbf{q}(k)$).

Dekodowanie polega na odczytaniu wartości k, będącej indeksem, do wektora wartości progowych $\mathbf{T}$ oraz do wektora liczby bitów kwantyzacji $\mathbf{q}$. Jeśli $\mathbf{q}(k) > 0$, to jest odczytywana z dekodera (o numerze $\mathbf{q}(k)$) wartość reszty e_q, w przeciwnym razie $e_q = 0$. Następnie jest wyznaczana wartość $|e|$ z zależności: $|e| = \mathbf{T}(k) + e_q$.

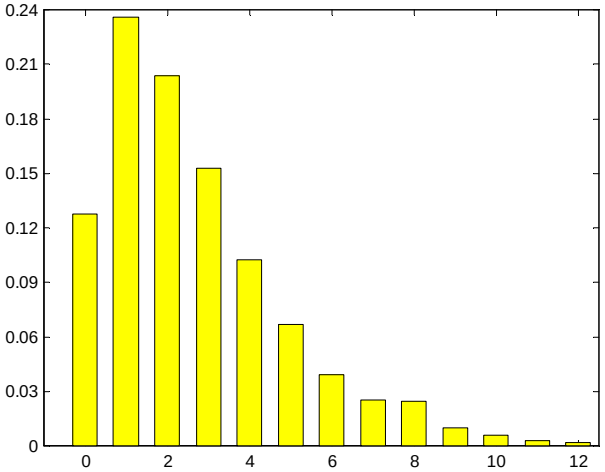
Na przykład niech $|e| = 29$. Wówczas $\mathbf{T}(14) = 24 \le 29 < \mathbf{T}(15) = 32$, czyli $k = 14$, $\mathbf{T}(14) = 24$, $e_q = 29 - 24 = 5$, $\mathbf{q}(14) = 3$. Zatem liczbę 5 kodujemy, używając trzeciego kodera arytmetycznego reszt kwantyzacji (kodującego trzybitowe liczby z przedziału od 0 do 7). W dekoderze $|e| = \mathbf{T}(14) + e_q = 24 + 5 = 29$.

Na rysunkach 8.1, 8.2 oraz 8.3 przedstawiono rozkłady prawdopodobieństw wartości $|e|$ w kontekstach, wyznaczanych według zasady z podrozdziału 8.2, o numerach, odpowiednio, 0, 4 oraz 11. Jeśli porównać dwa pierwsze rysunki (przypominające rozkład Laplace'a)

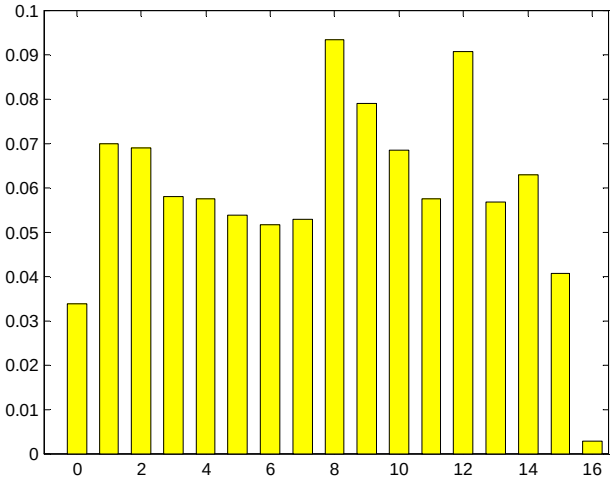
z rys. 8.3, to daje się wyraźnie na nim zauważyć wpływ zarówno kwantyzacji samej wartości $|e|$, jak i kwantyzacji zmiennej ω.



Rys. 8.1. Rozkład prawdopodobieństwa wartości k w kontekście nr 0 dla obrazu Hotel



Rys. 8.2. Rozkład prawdopodobieństwa wartości k w kontekście nr 4 dla obrazu Hotel



Rys. 8.3. Rozkład prawdopodobieństwa wartości k w kontekście nr 11 dla obrazu Hotel

8.4. Kodowanie bitu znaku

Ze względu na symetryczność rozkładu prawdopodobieństwa błędów predykcji wygodniej jest kodować ich wartości bezwzględne $|e|$, co pozwala na szybszą adaptację rozkładów w poszczególnych kontekstach kodera arytmetycznego (patrz podrozdział 8.2). Osobno kodowany jest bit znaku, przy czym informację o nim trzeba zapisywać jedynie dla niezerowych wartości e, co jest cechą charakterystyczną autorskiej metody zaprezentowanej w tym podrozdziale.

W odróżnieniu od proponowanego tu rozwiązania w większości znanych rozwiązań stosuje się rzutowanie liczb ujemnych przez wyznaczanie ich wartości bezwzględnej pomniejszonej o 1, co sprawia, że nakładane są na siebie dwa nie w pełni symetryczne rozkłady prawdopodobieństw (największą dysproporcję daje się zauważyć, gdy liczba –1 jest rzutowana na 0 z informacją o ujemnym bicie znaku). W niektórych algorytmach nie przewiduje się specjalnego sposobu kodowania bitu znaku i zapisuje się do pliku wynikowego wartość tego bitu bez zmian. Te dwa aspekty prowadzą do niepełnego wykorzystania potencjału zredukowania zakresu kodowanych symboli słownika, z którego korzysta koder arytmetyczny.

W tej pracy zaproponowano użycie autorskiego adaptacyjnego kodera arytmetycznego z podziałem na 16 kontekstów. Rozkład tego dwusymbolowego źródła jest początkowo ustawiany jako jednostajny w każdym z kontekstów (liczniki wystąpień 0 i 1 są inicjalizowane wartością 5). Numer kontekstu jest wyznaczany jako liczba składająca się z czterech bitów, dwa pierwsze są bitami znaków błędów predykcji lewego i górnego sąsiada: $\mathrm{sgn}(e(1))$ oraz $\mathrm{sgn}(e(2))$ kodowanego błędu. Kolejne dwa bity są wyznaczane jako liczba z przedziału od 0 do 3 na podstawie czteropoziomowego (dwubitowego) kwantyzatora wartości ω z progami $\{8, 20, 180\}$.

9. Metoda mieszania predykcyjnego

9.1. Wprowadzenie

W pracach [26–28] ukazano wysoką skuteczność metody mieszania predyktorów (ang. *Blending Predictors*). Mieszanie to kombinacja liniowa wartości zbioru *v* predyktorów (nazywanych dalej subpredyktorami), która staje się końcową wartością przewidywaną kodowanego piksela *P*(0). Powyższe stwierdzenie mało jednak mówi o rzeczywistym funkcjonowaniu całej metody, zatem w celu lepszego przybliżenia mieszania predykcyjnego warto je porównać do sieci neuronowej, której konstrukcję oparto na wielowarstwowej sieci perceptronowej MLP (patrz podrozdział 7.1) – porównanie to zostanie przedstawione w podrozdziale 9.3.

Mając dany zbiór prostych, stałych subpredyktorów, można zauważyć, iż wartości entropii uzyskiwane przy kodowaniu każdym z subpredyktorów z osobna nie są imponujące (patrz tab. 2.1), jednak użycie choćby siedmiu prostych subpredyktorów wspólnie dzięki metodzie mieszania pozwala na istotne zmniejszenie wartości entropii [102], i to w znaczącej większości testowanych obrazów. Dlatego właśnie ta propozycja stała się podstawą w docelowo proponowanym w niniejszej pracy autorskim systemie modelowania predykcyjnego, którego elementy składowe omówione zostaną w kolejnych podrozdziałach tego rozdziału. Jej dodatkową zaletą jest możliwość równoległego wykonywania wielu obliczeń w ramach kodowania pojedynczego piksela, co może być przydatne przy projektowaniu rozwiązań sprzętowych.

9.2. Dobór subpredyktorów

Zasada mówiąca, że spośród szerokiego zestawu stałych modeli predykcyjnych dobór *v* subpredyktorów dających indywidualnie najmniejsze wartości entropii pozwoli uzyskać najlepszy rezultat, nie zawsze jest prawdziwa w metodzie mieszania subpredyktorów. Wręcz przeciwnie, opłacalne staje się użycie jak najbardziej różnorodnych subpredyktorów, przy czym każdy z nich powinien dawać mały błąd w charakterystycznym dla niego typie najbliższego otoczenia (obszar gładki, teksturowany, krawędziowy, zaszumiony). Dlatego podążając tym tropem, badania w niniejszej pracy ukierunkowano na znalezienie różnych typów metod predykcyjnych, które indywidualnie nie muszą przodować w testach na dużych zbiorach obrazów, lecz sprawdzają się wspólnie dzięki technice mieszania, co uzasadniono w pracy [35]. W tabeli 9.1 przedstawione są propozycje zestawów subpredyktorów, metody z ich wykorzystaniem nazwano, odpowiednio, Blend-13 i Blend-17. Zestawy te zostały odpowiednio dobrane po wielu eksperymentach polegających na redukcji poszczególnych modeli predykcyjnych spośród zbioru stałych predyktorów zaprezentowanych w tab. 2.1. Usuwano pojedynczo modele, których brak powodował spadek średniej bitowej. W obu metodach wykorzy-

stano GAP_+ opisany w podrozdziale 3.1.2, ponadto w metodzie Blend-17 użyto także metody TCM, która zostanie opisana w podrozdziale 9.5.

Metoda Blend-13 jest uproszczoną wersją mieszania predyktorów zaprojektowaną do potrzeb realizacji sprzętowej, w której zrezygnowano z obliczeń zmiennopozycyjnych oraz redukcji zakresu odcieni omówionej w podrozdziale 1.5. Szczegóły jej realizacji zaprezentowano między innymi w pracach [35, 116, 123, 124, 131].

9.3. Zasada działania metody mieszania predykcyjnego

Wysoką efektywność uzyskuje się dla każdego z obrazów dzięki elastyczności, jaką cechuje się sposób adaptacji decydującej o doborze subpredyktora aktualnie dominującego (może być jeden lub kilka jednocześnie, przy czym zmieniają się one w różnych częściach obrazu). Waga przypisana danemu subpredyktorowi jest tym większa, im mniejsze błędy predykcji uzyskano w najbliższym otoczeniu wśród wcześniej zakodowanych pikseli. Całkowitą wartość błędu E_i otoczenia składającego się z m_i błędów predykcji, jakie uzyskano, posługując się i-tym subpredyktorem, wyznaczamy ze wzoru:

$$E_i = 1 + \sum_{j=1}^{m_i} \overline{d}_j \cdot \left| e_i(j) \right|, \tag{9.1}$$

gdzie $e_i(j)$ oznacza wartość błędu predykcji otrzymaną na podstawie i-tego subpredyktora w sąsiedztwie piksela $P(0)$ o numerze względnym j (patrz rys. 1.1).

Kolejnym krokiem jest wyznaczenie wartości wagi w_i i-tego subpredyktora:

$$w_i = \alpha_i \cdot \left(\frac{\delta_i}{E_i} \right)^4, \tag{9.2}$$

gdzie α_i nazywamy współczynnikiem istotności każdego subpredyktora, a δ_i jest wyznaczane ze wzoru (3.11), przy czym m_i to wielkość otoczenia i-tego subpredyktora.

Uproszczoną postać wzoru (9.1) wykorzystano w metodzie Blend-13:

$$E_i = 1 + 2(e_i^2(1) + e_i^2(2)) + \sum_{j=3}^{m_i} e_i^2(j), \tag{9.3}$$

zastępując jednocześnie wzór (9.2) poniższym:

$$w_i = \frac{\alpha_i}{E_i}. \tag{9.4}$$

Biorąc pod uwagę zbiór v subpredyktorów, możemy wyznaczyć przypisane im dodatnie współczynniki predykcji a_i, uwzględniając przy tym ich normalizację (suma wszystkich współczynników powinna wynosić 1):

$$a_i = \frac{w_i}{\sum_{j=1}^{v} w_j}. \tag{9.5}$$

Wówczas wynikowa wartość przewidywana predyktora głównego jest wyliczana jako:

$$\hat{x}_{(1)} = \sum_{i=1}^{v} a_i \cdot \hat{x}_i \ . \tag{9.6}$$

Jak widać, mieszanie to kombinacja liniowa v subpredyktorów $\hat{x}_i$, a ich współczynniki (wagi znormalizowane) a_i zależą nieliniowo od poziomu błędów predykcji z najbliższego otoczenia. Na podstawie powyższych wzorów można porównać mieszanie predykcyjne do wielowarstwowej sieci perceptronowej MLP (patrz rys. 7.1) przy następujących założeniach i ograniczeniach:

– identycznie jak w sieci neuronowej opisanej w rozdziale 7 warstwa neuronów wejściowych jest interpretowana jako sąsiedztwo m_{in} najbliższych pikseli aktualnie kodowanego $P(0)$;

– każdemu neuronowi z warstwy ukrytej odpowiada odrębny subpredyktor (a dokładniej moduł zawierający trzy techniki kaskadowo połączone: autorską metodę predykcji; opcjonalnie aktywowany blok NLMS+; opcjonalnie aktywowany blok korekcji skumulowanego błędu tego subpredyktora – przykład takiego modułu przedstawiono na rys. 6.3), który może należeć do kategorii stałych lub adaptacyjnych;

– neuron warstwy wyjściowej jest tu reprezentowany jako ostateczny moduł predykcji (również konstrukcyjnie zgodny ze schematem przedstawionym na rys. 6.3) wyznaczający wartość przewidywaną na podstawie wartości subpredyktorów i przypisanych im znormalizowanych dodatnich wag a_i (obliczanych ze wzoru (9.5)).

W porównaniu z sieciami MLP dużą zaletą rozwiązania proponowanego w tym rozdziale jest brak potrzeby użycia okna treningowego, w którym tradycyjna sieć neuronowa dokonuje iteracyjnego uczenia (adaptacyjnego doboru wag), z tego też powodu można uzyskać znacznie mniejszą złożoność w przypadku prostej wersji metody mieszania predykcyjnego. Ponadto uzyskujemy nieograniczoną wręcz elastyczność dzięki możliwości wprowadzania dowolnych zasad adaptacji każdego z subpredyktorów (a nawet używania wyłącznie stałych predyktorów, co pokazano np. w pracy [102]).

9.4. Cechy szczególne metod Blend-13 i Blend-17

Pomysł mieszania subpredyktorów przedstawiony w pracy [102] był rozwijany między innymi przez G. Denga oraz H. Ye i prezentowany np. w pracach [26, 27, 147]. Jednakże autorzy tych opracowań w dość uproszczony sposób podchodzili do sprawy doboru wielkości otoczenia branego pod uwagę przy wyznaczaniu współczynników wagowych w_i każdego z używanych subpredyktorów. W pracy [129] podjęto próbę przeanalizowania wpływu wiel-

kości otoczenia na efektywność predykcyjnego modelowania danych opartego na mieszaniu subpredyktorów. Problemem, jaki należało rozwiązać, był fakt, iż dla badanych obrazów najlepsze (wspólne dla wszystkich subpredyktorów) wielkości otoczenia mieściły się w całym zakresie analizowanego parametru m_i z przedziału od 3 do 30 (badania opisano w pracy [129]). Seemann i Tisher w pracy [102] zaproponowali wartość $m_i = 3$, Deng w swych pracach używał $m_i = 4$ [27]. Na podstawie badań (dla zbioru 45 różnych obrazów testowych) najlepsze średnie rezultaty dla Blend-13 uzyskano przy $m_i = 10$. Szary obszar zaznaczony na rys. 1.1 przedstawia wielkość tego otoczenia. W metodzie Blend-17 wartości m_i wyznaczono indywidualnie dla każdego z 17 subpredyktorów (patrz ostatnia kolumna tab. 9.1).

Tab. 9.1. Zestawy subpredyktorów stosowanych w metodzie Blend-13 oraz Blend-17

Subpredyktory Blend-13	α_i Blend-13	Subpredyktory Blend-17	α_i Blend-17	m_i Blend-17
GAP_+	1,0	Plane	1,0	20
GradWest	2,0	Plane2	1,5	32
GradNorth	2,0	GradWest	1,5	36
Plane	1,0	GradNorth	2,0	32
Plane2	1,5	$P(1)$	1,0	6
$P(1)$	1,0	$P(2)$	1,0	6
$P(2)$	1,0	$P(3)$	1,0	20
$P(3)$	1,0	$P(4)$	1,0	16
$P(4)$	1,0	$P(5)$	1,0	26
$P(5)$	1,0	$P(6)$	1,0	26
$P(10)$	1,0	$P(10)$	1,0	14
$P(18)$	1,0	$P(18)$	1,0	24
$P(28)$	1,0	$P(28)$	1,0	28
–	–	$P(13)$	1,0	28
–	–	$2P(3) - P(11)$	1,0	22
–	–	GAP_+	1,0	40
–	–	TCM	1,5	30

Zarówno indywidualizacja wielkości otoczeń m_i, jak i współczynników istotności α_i są nowym podejściem, zaproponowanym przez autora tej pracy. Dla każdego zestawu subpredyktorów osobno w iteracyjny sposób dobiera się te pary parametrów. Wartości poszczególnych α_i użytych w metodach Blend-13 i Blend-17 znajdują się w tab. 9.1.

Kolejnym problemem było ustalenie zasad użycia, mieszanej korekcji skumulowanego błędu predykcji, opisanej w podrozdziale 4.3. W pracy [103] autorzy zaproponowali korygowanie wartości predykcji zarówno każdego subpredyktora, jak i głównego modelu predykcji, wyznaczanego ze wzoru (9.6). Okazuje się, że znacznie lepsze rezultaty można uzyskać, rezygnując z aktywacji korekcji w przypadku sporej części subpredyktorów. W metodzie Blend-17 oprócz predyktora głównego jedynie subpredyktory GAP_+ oraz TCM wykorzystują aktywną korekcję skumulowanego błędu predykcji.

W tabeli 9.2 przedstawiono porównanie wartości entropii otrzymanych metodami GAP_+, SOLP [90], $GBSW_+$ [58] oraz czterema metodami wykorzystującymi mieszanie pre-

dykcyjne. Tabela 9.3 zawiera porównanie średnich bitowych dla proponowanej w tym podrozdziale sprzętowej realizacji metody Blend-13 z kilkoma programowymi, lecz szybkimi i efektywnymi metodami zaprezentowanymi w literaturze (w tym trzy metody wykorzystujące metodę mieszania predykcyjnego: CBPC [59], HBB [103] oraz P13 [28]). Wśród porównywanych znalazła się też nieco wolniejsza, lecz efektywna metoda LAT-RLMS [75], co pokazuje, jak wysoką skutecznością charakteryzuje się metoda Blend-13, dla której uzyskano w tym zestawieniu najniższą średnią bitową. Dzięki zastosowaniu wielu uproszczeń i obliczeń bez użycia liczb zmiennopozycyjnych udało się metodą Blend-13 uzyskać czas kodowania obrazu Lennagrey wynoszący zaledwie 0,48 s (z użyciem procesora Pentium4 2,8 GHz).

Tab. 9.2. Pomiar wartości entropii standardowych obrazów testowych

Obrazy	GAP_+ [132]	SOLP [90]	$GBSW_+$ [58]	Blend-7 [102]	WAVE [27]	CBP [59]	Blend-13 [131]
Balloon	2,998	2,991	3,023	2,93	2,873	2,84	2,806
Barb	5,012	5,022	4,942	4,90	4,936	4,34	4,485
Barb2	5,003	4,976	5,034	5,02	4,872	4,80	4,800
Board	3,892	3,794	3,768	3,81	3,679	3,63	3,607
Boats	4,229	4,220	4,227	4,16	4,095	4,02	3,991
Girl	4,044	4,036	3,961	3,91	3,844	3,80	3,766
Gold	4,699	4,699	4,785	4,65	4,598	4,59	4,558
Hotel	4,676	4,624	4,582	4,58	4,500	4,48	4,445
Zelda	3,936	3,938	3,930	3,90	3,817	3,81	3,756
Średnia	4,276	4,256	4,250	4,207	4,135	4,034	4,024

Tab. 9.3. Pomiar wartości średnich bitowych dla standardowych obrazów testowych

Obrazy	JPEG-LS [107]	HBB [103]	CALIC [149]	CBPC [59]	Lee [66]	P13 [28]	LAT-RLMS [75]	Blend-13 [131]
Balloon	2,889	2,80	2,78	2,78	2,79	2,74	2,75	2,746
Barb	4,690	4,28	4,31	4,14	4,20	4,29	4,15	4,172
Barb2	4,684	4,48	4,46	4,47	4,47	4,47	4,45	4,441
Board	3,674	3,54	3,51	3,49	3,50	3,48	3,48	3,465
Boats	3,930	3,80	3,78	3,78	3,76	3,75	3,74	3,716
Girl	3,922	3,74	3,72	3,70	3,70	3,67	3,68	3,640
Gold	4,475	4,37	4,35	4,38	4,35	4,33	4,34	4,323
Hotel	4,378	4,27	4,18	4,23	4,24	4,19	4,21	4,189
Zelda	3,884	3,72	3,69	3,72	3,68	3,68	3,61	3,673
Średnia	4,058	3,889	3,864	3,854	3,854	3,844	3,823	3,818

9.5. Metoda dopasowania teksturowanego

W tym podrozdziale zostanie przedstawiony adaptacyjny algorytm kontekstowego dopasowania teksturowanego TCM (ang. *Texture Context Matching*). Metoda ta sprawdza się głównie w kodowaniu obrazów o charakterystycznych obszarach teksturowanych. Chodzi o fragmenty obrazu, na których pojawiają się pewne powtarzalne wzorce (np. obrus w kształ-

cie kraty oraz prążkowana chusta i spodnie widoczne na obrazie Barb – patrz rys. 9.1). Jest to kolejna z metod, którą dołączono do zestawu subpredyktorów, projektując metodę Blend-17.

Na potrzeby wykazania przydatności metody dopasowania teksturowanego zostanie w tym podrozdziale wprowadzona metoda Blend-16, którą otrzymujemy z metody Blend-17, rezygnując z subpredyktora TCM (patrz tab. 9.4 i 9.6).

Rys. 9.1. Testowy obraz Barb

Metoda dopasowania teksturowanego polega na porównywaniu wzorca złożonego z m pikseli $P(i)$ najbliższego sąsiedztwa aktualnie kodowanego piksela $P(0)$ (z zachowaniem zasady przyczynowości niezbędnej do właściwej pracy dekodera) z innymi wzorcami $\mathbf{Y}_{\text{off}}$ składającymi się z pikseli od $P_{\text{off}}(1)$ do $P_{\text{off}}(m)$ skojarzonymi z wcześniej zakodowanymi pikselami, należącymi do pewnego otoczenia Q określanego jako okno treningowe. Obszar Q składa się z pikseli leżących w W wierszach powyżej $P(0)$, ale oddalonych nie więcej niż o W pikseli w lewą lub prawą stronę. Uzupełnieniem obszaru Q jest W pikseli leżących w aktualnie kodowanym wierszu po lewej stronie $P(0)$. Obszar Q składa się zatem z $2W(W + 1)$ pikseli. Przykład dla $m = 5$ oraz $W = 3$ przedstawia rys. 6.2. Miarą podobieństwa wzorców może być odległość typu Manhattan:

$$\Delta = \frac{1}{m}\sum_{i=1}^{m}\left|P(i) - P_{\text{off}}(i)\right|. \tag{9.7}$$

Im wartość Δ jest mniejsza, tym większa szansa, że aktualnie kodowany piksel $P(0)$ jest podobny do piksela $P_{\text{off}}(0)$ skojarzonego z otoczeniem $\mathbf{Y}_{\text{off}}$. Możemy zatem jako wartość przewidywaną użyć piksela $P_{\text{off}}(0)$. Wcześniej podobną ideę TCM zaprezentowano w pracy [55] jako jedną z metod przełączalnych w zależności od wariancji otoczenia. W tamtym rozwiązaniu używano wyłącznie odległości euklidesowej i stałych parametrów $m = 4$ przy obszarze Q składającym się z 36 pikseli.

W omawianej tu autorskiej metodzie TCM istotny jest dobór parametrów W oraz m. Wzrost W wiąże się ze zwiększeniem liczby pikseli $P_{\text{off}}(0)$. Jednak czasami wraz ze wzrostem odległości piksela $P_{\text{off}}(0)$ od $P(0)$, mimo znajdowanych mniejszych poziomów Δ, wartość przewidywana może być mniej korzystna, niż gdyby użyć mniejszego obszaru poszukiwań Q (wynika to z zachowania właściwości podobieństwa lokalnego). Z tego względu dobór parametru W ma duże znaczenie dla końcowego rezultatu kodowania poszczególnych obrazów.

Proponowane tu autorskie rozwiązanie uwzględnia zatem dwa typy algorytmów TCM, dając w rezultacie kombinację liniową dwóch predyktorów. Wielkość wzorca ustalono eksperymentalnie na $m = 22$.

W pierwszym przypadku wyszukujemy globalną wartość minimum Δ dla całego obszaru Q, przy czym Δ wyznaczamy, modyfikując wzór (9.7) do postaci:

$$\Delta = \sum_{i=1}^{m} \bar{d}_i \cdot \left| P(i) - P_{\text{off}}(i) \right|, \tag{9.8}$$

gdzie $\bar{d}_i$ jest współczynnikiem wagowym obliczanym ze wzoru (3.9). Dla tak wyznaczonego wzorca $\mathbf{Y}_{\text{global}}$ otrzymujemy wartość predykcji P_{global}.

Drugi sposób obliczania metodą TCM wartości predykcji P_{mix} polega na iteracyjnym wyznaczaniu kolejnych wartości predykcji P_W dla każdej wysokości W od 1 do $W_{\max}$ – wcześniej ustalonej maksymalnej wartości W. Jest to zatem wyznaczanie minimalnej wartości Δ_W i skojarzonej z nią wartości P_W dla coraz większego okna treningowego Q. Wówczas przewidywana wartość P_{mix} jest wyznaczana ze wzoru:

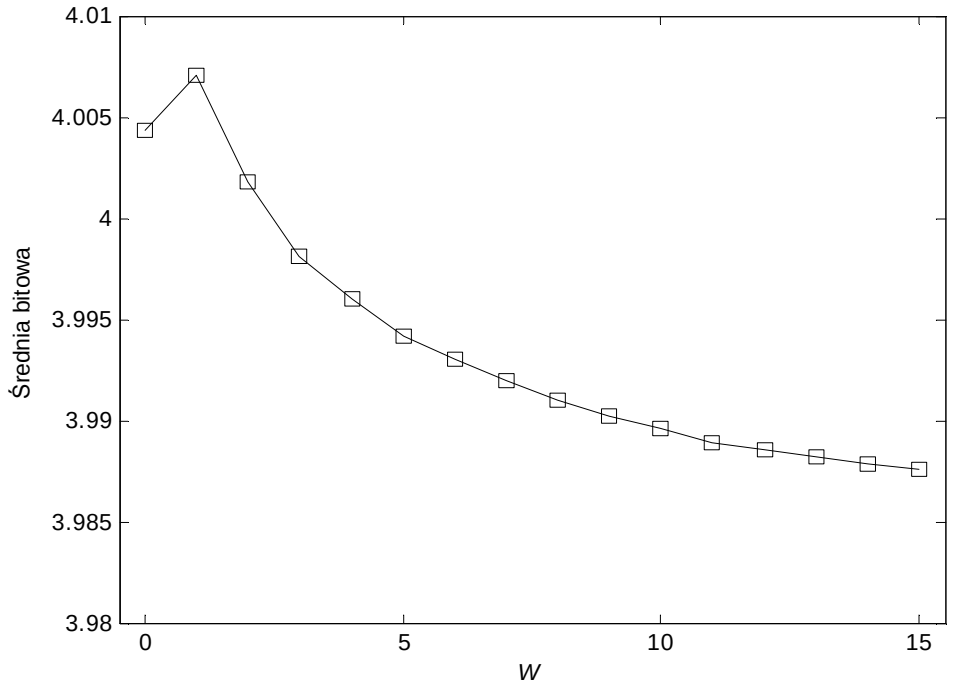
$$P_{\text{mix}} = \frac{\sum_{W=1}^{W_{\max}} \frac{1}{1+\Delta_W^2} \cdot P_W}{\sum_{W=1}^{W_{\max}} \frac{1}{1+\Delta_W^2}}. \tag{9.9}$$

Ostateczną wartość predykcji P_{TCM} obliczamy jako:

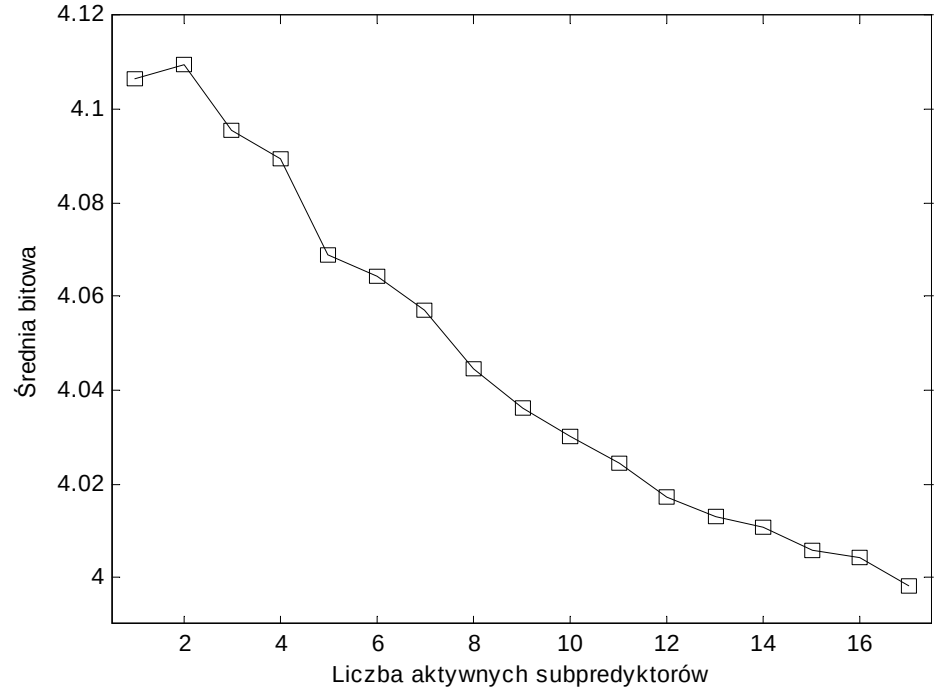
$$P_{TCM} = 0{,}2P_{\text{global}} + 0{,}8P_{\text{mix}}. \tag{9.10}$$

Metodę TCM można z powodzeniem wykorzystać jako jeden z subpredyktorów przy rozbudowie algorytmu mieszania predykcyjnego. Jeśli do 16 podstawowych subpredyktorów stałych w metodzie mieszania dołączymy P_{TCM} jako 17. subpredyktor, to uzyskamy spadek średniej bitowej przeciętnie o 0,01 bitu na piksel przy $W = 5$ oraz 0,017 bitu na piksel przy $W = 15$. Wykres średniej bitowej dla metody Blend-17, w zależności od wysokości W okna danych treningowych, przedstawia rys. 9.2. Jedynie dla $W = 1$, czyli gdy obszar Q składa się zaledwie z czterech pikseli, metoda daje gorszy rezultat od Blend-16. Dla $W > 1$ otrzymujemy zmniejszenie średniej. Z analizy poszczególnych pomiarów wynika, że jedynie w 14 (dla $W = 3$) oraz w 8 (dla $W = 15$) na 45 przypadków uzyskano nieznaczne pogorszenie w stosunku do metody Blend-16, w pozostałych przypadkach mamy wzrost efektywności.

Na rysunku 9.3 przedstawiono wykres średniej bitowej dla metody mieszania Blend-17 w zależności od liczby aktywnych subpredyktorów (przy $W = 3$ w metodzie TCM). Wykres ten pokazuje, jak istotne są kolejno aktywowane subpredyktory i w jaki sposób, projektując nową metodę kompresji, można regulować ich liczbę, gdy jest rozważana zależność efektywności i czasu kodowania.

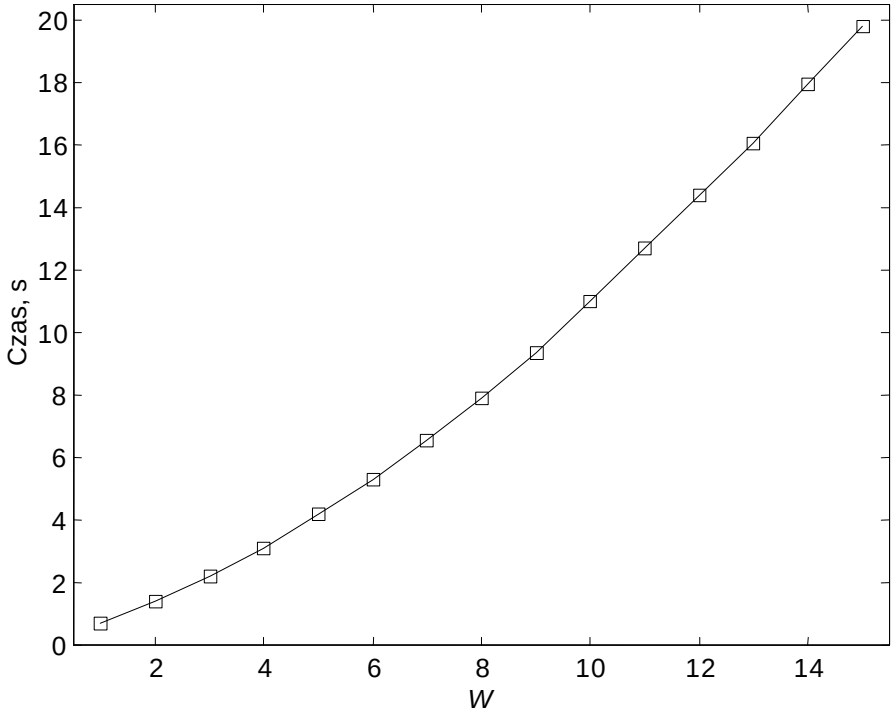


Rys. 9.2. Zależność średniej bitowej od wysokości *W* okna danych treningowych w metodzie Blend-17



Rys. 9.3. Średnia bitowa w zależności od liczby *v* aktywnych subpredyktorów

W metodzie dopasowania teksturowanego złożoność wyznaczania każdej wartości przewidywanej P_{TCM} wynosi $O(m \cdot W^2)$. Na rysunku 9.4 przedstawiono czas kodowania metodą TCM w funkcji *W* oznaczającej wysokość okna danych treningowych *Q*. Dzięki odpowiedniej implementacji (dla małych wartości $W \leq 15$) udało się uzyskać prawie liniową zależność czasu kodowania od parametru *W*.



Rys. 9.4. Zależność czasu kodowania metodą TCM od wysokości boku *W* okna danych treningowych w metodzie Blend-17

Na podstawie pomiarów z użyciem 45 obrazów testowych efektywność metod Blend-16 i Blend-17 możemy określić jako zbliżoną do M-LMS (patrz tab. 9.4). Czas kodowania obrazu Lennagrey metodą Blend-16 wynosi 4,48 s, a metodą Blend-17 (przy $W = 3$) 6,75 s. Jest to zatem czas ponaddwukrotnie dłuższy niż metodą M-LMS, ale nadal ponadtrzykrotnie krótszy niż metodą GLICBAWLS.

Zastosowanie wysokiej wartości $W = 15$ w metodzie TCM, będącej 17. subpredyktorem w Blend-17, ma sens jedynie przy projektowaniu kodera o bardzo dużej efektywności, bowiem czas kodowania metodą Blend-17 dla takiego okna treningowego wynosi 24,25 s.

9.6. Metoda Blend-19

Guang Deng i współautorzy w swoich kolejnych pracach przedstawiali różne możliwości zwiększenia efektywności mieszania predykcyjnego. W publikacji [28] zaprezentowano metodę P13 opartą na 13 stałych subpredyktorach. W pracy [26] zamiast zwiększania liczby subpredyktorów w metodzie APC-A postanowiono użyć tylko sześciu najistotniejszych stałych predyktorów oraz jednej metody adaptacyjnej TDLMS. Pozwoliło to uzyskać spadek średniej bitowej z 3,844 w P13 do 3,777 bitu na piksel (średnia uzyskana dla dziewięciu obrazów testowych – patrz tab. 9.4). Idąc tą drogą, przy projektowaniu kolejnego rozszerzenia metody mieszania predykcyjnego można wykorzystać opisane wcześniej metody o średniej złożoności czasowej.

Tab. 9.4. Pomiar średnich bitowych standardowych obrazów testowych

Obrazy	Blend-16	Blend-17 ($W = 3$)	Blend-19 ($W = 3$)	P13 [28]	APC-A [26]	M-LMS	GLICBAWLS [85]
Balloon	2,670	2,671	2,630	2,74	2,73	2,672	2,640
Barb	4,086	4,049	3,934	4,29	4,04	4,037	3,916
Barb2	4,358	4,353	4,238	4,47	4,40	4,305	4,318
Board	3,375	3,364	3,310	3,48	3,43	3,357	3,392
Boats	3,638	3,633	3,568	3,75	3,70	3,620	3,628
Girl	3,572	3,568	3,500	3,67	3,61	3,547	3,565
Gold	4,261	4,258	4,222	4,33	4,30	4,258	4,276
Hotel	4,090	4,078	4,038	4,19	4,15	4,103	4,177
Zelda	3,603	3,606	3,538	3,68	3,63	3,559	3,537
Średnia	3,739	3,731	3,664	3,844	3,777	3,717	3,717

W projektowanej tu metodzie Blend-19 bazowano na Blend-17. W miejsce GAP_+ została wprowadzona metoda wykorzystująca statyczne predyktory liniowe z podziałem na cztery konteksty, którą opisano w podrozdziale 4.4. Do tych 17 subpredyktorów dołączono jeszcze dwa adaptacyjne (metody $ALCM_+$ oraz $CoBALP_+$ opisane, odpowiednio, w podrozdziałach 5.3 i 5.4) – szczegóły dotyczące wag α_i i wielkości otoczeń m_i umieszczono w tab. 9.5. W metodzie Blend-19, oprócz predyktora głównego, jedynie subpredyktory statyczny oraz adaptacyjne $ALCM_+$ i $CoBALP_+$ wykorzystują aktywną korekcję skumulowanego błędu predykcji.

Przy parametrze $W = 3$ w subpredyktorze TCM czas kodowania obrazu Lennagrey metodą Blend-19 zwiększył się w porównaniu z Blend-17 z 6,75 do 10,46 s, co oznacza dwukrotnie krótszy czas w porównaniu z kodowaniem metodą GLICBAWLS. Średnia bitowa z 45 pomiarów zmniejszyła się z 3,99835 do 3,93654 (patrz tab. 9.6). W teście podstawowym dla dziewięciu obrazów testowych metoda Blend-19 uzyskała znacznie lepszy rezultat w porównaniu z metodą APC-A, a także z GLICBAWLS (patrz tab. 9.4).

9.7. Metoda Blend-20

Aby uzyskać dalszy wzrost efektywności w stosunku do Blend-19, należy nie tylko zwiększyć liczbę subpredyktorów, lecz także wykorzystać te o znacznie wyższej złożoności obliczeniowej (patrz tab. 9.5). To z kolei sprawia, iż ich indywidualne wysokie efektywności silnie wpływają na ostateczny rezultat średniej bitowej. Z tego powodu należy zrezygnować z metody statycznej z przełączaniem kontekstowym ze względu na zbyt duży rozmiar wymaganego nagłówka, pozbywamy się w ten sposób czterech zestawów współczynników predykcji. W Blend-20 [130] metoda statyczna zostaje zastąpiona opisaną w podrozdziale 6.1 techniką RLS_+ z rzędem predykcji $r = 40$ (a $r = 72$ dla obrazów o rozdzielczości większej od 512×512). Dodatkowo wprowadzono subpredyktor OLS (patrz podrozdział 6.2) z rzędami predykcji: $r = 46$ ($W = 12$) przy rozdzielczości 256×256, $r = 38$ ($W = 14$) przy rozdzielczości 512×512 oraz $r = 44$ ($W = 12$) przy rozdzielczości 720×576. Jak widać, są to znacznie wyższe rzędy w porównaniu z pierwotnie dobraną wartością $r = 14$ dla indywidualnie aktyw-

nej metody OLS. Pełny zestaw subpredyktorów z przypisanymi im parametrami α_i oraz m_i umieszczono w prawej części tab. 9.5.

Udoskonalenia te pozwalają uzyskać duży wzrost efektywności kosztem znacznego wydłużenia czasu kodowania.

Tab. 9.5. Zestawy subpredyktorów dla metod Blend-19 oraz Blend-20

Subpredyktory Blend-19	α_i Blend-19	m_i Blend-19	Subpredyktory Blend-20	α_i Blend-20	m_i Blend-20
Plane	1,0	20	Plane	1,0	20
Plane2	1,5	32	Plane2	1,5	32
GradWest	1,5	36	GradWest	1,5	36
GradNorth	2,0	32	GradNorth	2,0	32
$P(1)$	1,0	6	$P(1)$	1,0	6
$P(2)$	1,0	6	$P(2)$	1,0	6
$P(3)$	1,0	20	$P(3)$	1,0	20
$P(4)$	1,0	16	$P(4)$	1,0	16
$P(5)$	1,0	26	$P(5)$	1,0	26
$P(6)$	1,0	26	$P(6)$	1,0	26
$P(10)$	1,0	14	$P(10)$	1,0	14
$P(18)$	1,0	24	$P(18)$	1,0	24
$P(28)$	1,0	28	$P(28)$	1,0	28
$P(13)$	1,0	28	$P(13)$	1,0	28
$2P(3) - P(11)$	1,0	22	$2P(3) - P(11)$	1,0	22
Stat-4 kontekst	3,5	40	RLS_+	2,0	32
TCM	2,5	30	TCM	1,5	30
$ALCM_+$	2,0	32	$ALCM_+$	1,5	32
$CoBALP_+$	4,0	36	$CoBALP_+$	2,0	36
–	–	–	OLS	4,5	40

Na wyjściu metod Plane, Plane2, GradWest, RLS_+, TCM, $ALCM_+$, $CoBALP_+$ i OLS dołączona zostaje metoda NLMS+ z predykcją rzędu $r_{NLMS} = 72$, $r_{NLMS+} = 30$ (patrz podrozdział 6.5). Ponadto w metodzie Blend-20 oprócz predyktora głównego jedynie subpredyktory RLS_+, $ALCM_+$, $CoBALP_+$ i OLS mają aktywną korekcję skumulowanego błędu predykcji.

Pierwowzorem kolejnego udoskonalenia był pomysł opisany w pracach [66, 101]. W Blend-20 zostaje wprowadzona druga metoda mieszania predykcyjnego, która wykorzystuje funkcję wykładniczą do pomiaru całkowitego błędu otoczenia $E_i^{(2)}$ skojarzonego z i-tym subpredyktorem:

$$E_i^{(2)} = \exp\left(\frac{\chi}{\delta}\sum_{j=1}^{m=72}\overline{d}_j \cdot \left|e_i(j)\right|\right), \tag{9.11}$$

gdzie χ przy $m = 72$ wyznaczamy ze wzoru:

$$\chi = \frac{24}{\sqrt[3]{\overline{\sigma}^2}}, \tag{9.12}$$

a wartość $\overline{\sigma}^2$ jest średnią arytmetyczną wszystkich wariancji wagowych $\tilde{\sigma}^2$ wyznaczanych zgodnie ze wzorem (3.12). Wagi subpredyktorów, wyznaczane ze wzoru:

$$w_i = \frac{\alpha_i}{E_i^{(2)}}. \tag{9.13}$$

podstawia się do wzoru (9.5) celem otrzymania znormalizowanych współczynników a_i drugiej metody mieszania. Wagi są wyznaczane podobnie do uproszczonej postaci odwrotności funkcji sigmoidalnej, którą wykorzystuje się w sieciach neuronowych.

Dzięki nowej metodzie mieszania uzyskujemy drugą wartość przewidywaną, $\hat{x}_{(2)}$ (pierwsza, $\hat{x}_{(1)}$, jest wyznaczana zgodnie z opisem zawartym w podrozdziale 9.3). Pozwala to na zbudowanie kolejnej warstwy mieszania predykcyjnego z dwoma subpredyktorami: $\hat{x}_{(1)}$ i $\hat{x}_{(2)}$. Mieszanie predykcyjne w tej warstwie odbywa się zgodnie ze schematem podobnym do opisanego w podrozdziale 9.3, zatem E_1 i E_2 są wyznaczane ze wzoru (9.1), przy czym $m_1 = m_2 =$ 18, a wartość predyktora głównego $\hat{x}$ przy parametrach $\alpha_1 = 1{,}8$, $\alpha_2 = 1$ jest obliczana ze wzoru:

$$\hat{x} = \frac{\frac{\alpha_1}{E_1^3} \cdot \hat{x}_{(1)} + \frac{\alpha_2}{E_2^3} \cdot \hat{x}_{(2)}}{\frac{\alpha_1}{E_1^3} + \frac{\alpha_2}{E_2^3}}. \tag{9.14}$$

Biorąc pod uwagę podobieństwa konstrukcyjne metody mieszanej do wielowarstwowej sieci perceptronowej MLP (patrz podrozdział 9.3), można przez analogię porównać wprowadzenie tej nowej warstwy do użycia drugiej warstwy ukrytej w sieciach neuronowych.

Przy parametrze $W = 3$ w subpredyktorze TCM czas kodowania metodą Blend-20 zwiększył się w porównaniu z Blend-19 z 10,46 do 107,3 s, co oznacza ponad 11-krotnie krótszy czas w porównaniu z kodowaniem metodą MRP 0.5 [78] (1229,05 s dla obrazu Lennagrey).

W przypadku metody TMW$^{\text{LEGO}}$ w literaturze podaje się jedynie orientacyjnie, że czas kodowania włącznie z analizą obrazu jest liczony w godzinach. Sam czas kodowania (dla ustalonych wcześniej parametrów, bez żmudnego etapu ich dostrajania) jest około 325 razy dłuższy niż w przypadku JPEG-LS [26], a także 2,33 razy dłuższy niż dla metody zaprezentowanej w artykule [150], która jest bardzo zbliżona złożonością do Blend-20.

Zaprezentowana w podrozdziale 6.7 analiza metody WLS, będącej uproszczeniem metody multi-WLS [148], pozwala ocenić, iż czas kodowania metodą Blend-20 jest znacznie krótszy niż w przypadku multi-WLS.

Średnia bitowa z 45 pomiarów zmniejszyła się z 3,93654 dla metody Blend-19 do 3,88627 bitu na piksel dla metody Blend-20 (patrz tab. 9.6). Przy tak atrakcyjnym czasie kodowania dla podstawowego zestawu dziewięciu obrazów testowych metodą Blend-20 uzyskano znacznie lepszy rezultat średniej bitowej niż innymi znanymi z literatury metodami (patrz tab. 9.9).

Tab. 9.6. Porównanie średnich bitowych dla metod mieszanych

Obrazy	Blend-13	Blend-16	Blend-17 ($W = 3$)	Blend-17 ($W = 15$)	Blend-19 ($W = 3$)	Blend-20 ($W = 3$)
Aerial	4,56574	4,52557	4,50998	4,49664	4,46741	4,43325
Airfield	5,43027	4,85392	4,85199	4,84449	4,82117	4,80478
Airplane	3,66932	3,60704	3,61573	3,61103	3,58409	3,56769
Baboon	5,77811	5,72903	5,72328	5,71805	5,69750	5,64709
Barbara	4,39990	4,29164	4,25175	4,19413	4,08284	3,87992
Boat	4,55189	4,50221	4,49393	4,48547	4,40979	4,36032
Bridge	5,36295	3,33620	3,33434	3,33270	3,33131	3,32143
Couple	4,50260	4,06311	4,05561	4,04956	4,05420	4,02384
Crowd	3,64229	3,56265	3,56775	3,56574	3,53807	3,51441
Elaine	4,72020	4,61842	4,61091	4,59482	4,35364	4,25054
Finger	5,33377	5,32555	5,34278	5,33051	5,16614	5,13428
Frog	5,97184	5,00173	5,00383	4,99693	4,98932	4,97578
Goldhill	4,55489	4,50058	4,49646	4,48856	4,45730	4,41833
Harbour	4,34235	4,25655	4,22919	4,21490	4,21286	4,17579
Lax	5,57565	5,52003	5,51237	5,50509	5,50697	5,49239
Lennagrey	4,00647	3,93311	3,92733	3,92218	3,91882	3,87912
Lena_{TMW}	4,38941	4,31926	4,31361	4,30693	4,30657	4,26780
Man	4,26123	4,20187	4,19380	4,19062	4,20958	4,18358
Peppers	4,31247	4,24582	4,24210	4,23724	4,21375	4,15127
Sailboat	4,59416	4,52079	4,52768	4,51816	4,42773	4,39043
Seismic	2,71399	2,66070	2,65935	2,65852	2,13420	2,03130
Shapes	1,33946	1,04999	1,00784	0,98316	1,01139	0,99232
Tank	4,67417	3,76619	3,77036	3,76408	3,76488	3,75564
Truck	4,42160	4,01561	4,01900	4,01694	4,00839	4,00023
Woman1	4,44671	3,80158	3,78343	3,78005	3,79049	3,74586
Woman2	3,09476	3,01289	3,01845	3,01463	3,00203	2,96936
Balloon	2,74579	2,67039	2,67111	2,66727	2,63030	2,56612
Barb	4,17235	4,08640	4,04878	3,97805	3,93449	3,76829
Barb2	4,44108	4,35767	4,35337	4,31097	4,23805	4,17540
Board	3,46477	3,37518	3,36360	3,35382	3,31010	3,27187
Boats	3,71599	3,63842	3,63348	3,62476	3,56777	3,52037
Girl	3,64001	3,57173	3,56826	3,54470	3,50047	3,44930
Gold	4,32262	4,26123	4,25792	4,25149	4,22195	4,18491
Hotel	4,18866	4,09017	4,07786	4,06486	4,03755	4,00658
Zelda	3,67347	3,60387	3,60603	3,60188	3,53812	3,49799
Bridge256	5,58856	5,54564	5,54587	5,54260	5,55017	5,53207
Camera	4,09923	4,02026	4,00090	3,99002	4,00830	3,97505
Couple256	3,51659	3,40364	3,39555	3,39352	3,41801	3,39333
Earth	3,10558	2,82579	2,82376	2,82198	2,83054	2,80734
Elif	2,87910	2,82335	2,82423	2,81932	2,67824	2,56599
Noisesquare	5,34915	5,30368	5,30310	5,28937	5,26442	5,23250
Omaha	6,12172	6,02260	6,00809	6,00520	6,00261	5,99388
Sena	3,08542	3,02290	3,01770	3,01147	2,87247	2,75143
Sensin	3,35162	3,29991	3,30490	3,30115	3,15092	3,00415
Sinan	3,12166	3,05426	3,05827	3,05586	2,92953	2,81892
Średnia	4,20532	4,00442	3,99835	3,98777	3,93654	3,88627

9.8. Metody Blend-24 oraz Blend-25

9.8.1. Wprowadzenie

Metody Blend-24 i Blend-25 zostaną opisane wspólnie ze względu na niewielką różnicę konstrukcyjną. Dotychczas najwydajniejszą z opublikowanych metod była multi-WLS [148] wykorzystująca technikę mieszania wielu subpredyktorów typu WLS (patrz podrozdział 6.3). Rozszerzenia AVE-WLS1 i AVE-WLS2, zaproponowane w podrozdziale 6.4, pozwoliły uzyskać wzrost wydajności względem podstawowej wersji WLS. Dlatego metody te dołączono do Blend-20, co spowodowało znaczny wzrost czasu kodowania w przypadku nowej propozycji określanej dalej jako Blend-24. Ponadto wprowadzono kilka innych zmian względem Blend-20. Szczegóły dotyczące wag α_i i wielkości otoczeń m_i umieszczono w tab. 9.7.

Tab. 9.7. Zestawy subpredyktorów dla metod Blend-24 oraz Blend-25

Subpredyktory Blend-24	α_i Blend-24	m_i Blend-24	NLMS+	Subpredyktory Blend-25	α_i Blend-25	m_i Blend-25	NLMS+
Plane	0,5	20	nie	Plane	0,5	20	nie
Plane2	1,5	32	tak	Plane2	1,5	32	tak
GradWest	1,5	36	tak	GradWest	1,5	36	tak
GradNorth	2,0	32	nie	GradNorth	2,0	32	nie
$P(1)$	1,0	6	nie	$P(1)$	1,0	6	nie
$P(2)$	1,0	6	nie	$P(2)$	1,0	6	nie
$P(3)$	1,0	20	nie	$P(3)$	1,0	20	nie
$P(4)$	1,0	16	nie	$P(4)$	1,0	16	nie
$P(5)$	1,0	26	nie	$P(5)$	1,0	26	nie
$P(6)$	1,0	26	nie	$P(6)$	1,0	26	nie
$P(10)$	1,0	14	nie	$P(10)$	1,0	14	nie
$P(18)$	1,0	24	nie	$P(18)$	1,0	24	nie
$P(28)$	1,0	28	nie	$P(28)$	1,0	28	nie
$P(13)$	1,0	28	nie	$P(13)$	1,0	28	nie
$2P(3) - P(11)$	1,0	22	nie	$2P(3) - P(11)$	1,0	22	nie
RLS_+	2,0	32	tak	RLS_+	2,0	32	tak
TCM_+	2,5	30	nie	TCM_+	2,5	30	nie
$ALCM_+$	1,5	32	tak	$ALCM_+$	1,5	32	tak
$CoBALP_+$	2,0	36	tak	$CoBALP_+$	2,0	36	tak
OLS	3,0	40	tak	OLS	3,0	40	tak
AVE-WLS1	2,0	40	nie	AVE-WLS1	2,0	40	nie
AVE-WLS1	2,0	36	tak	AVE-WLS1	2,0	36	tak
AVE-WLS2	2,0	36	nie	AVE-WLS2	2,0	36	nie
AVE-WLS2	2,0	36	tak	AVE-WLS2	2,0	36	tak
–	–	–	–	$AdNN_+$	2,5	40	tak

Jak można zauważyć, Blend-25 jest rozszerzoną o subpredyktor $AdNN_+$ (patrz rozdział 7) wersją Blend-24 i jest to między nimi jedyna różnica w algorytmie działania.

Wśród metod mieszanych opisywanych w tej pracy w Blend-24 pierwszy raz zastosowano pomysł dwukrotnego wykorzystania subpredyktora, przy czym w pierwszym przypadku nie jest czynna korekcja skumulowanego błędu predykcji, a w drugim jest ona aktywna.

Dotyczy to subpredyktorów AVE-WLS1 i AVE-WLS2. Podwójne ich użycie oznacza wzrost łącznej liczby subpredyktorów do 24. Z pozostałych 20 subpredyktorów (znanych z metody Blend-20) jedynie $P(28)$, RLS_+, $ALCM_+$, $CoBALP_+$ i OLS, a także predyktor główny $\hat{x}$ wykorzystują aktywną korekcję skumulowanego błędu predykcji. Kolumna NLMS+ w tab. 9.7 informuje, na wyjściu którego z subpredyktorów jest dołączona metoda NLMS+ z predykcją rzędu $r_{NLMS} = 96$, $r_{NLMS+} = 30$ (patrz podrozdział 6.5).

W podrozdziałach od 9.8.2 do 9.8.5 opisano kolejne elementy zwiększające efektywność metod Blend-24 i Blend-25 względem Blend-20.

9.8.2. Udoskonalenie metody TCM

Jeśli przyjąć, że technika poszukiwania wzorców w sąsiedztwie kodowanego piksela $P(0)$ wygląda podobnie do poszukiwania (na podstawie wcześniejszych klatek) wektorów ruchu w sekwencji wideo, to można wykorzystać ideę poszukiwania wektorów z dokładnością do ułamkowej części piksela. Przykładem takiego pomysłu jest metoda Quarter Pixel, która dzięki wzrostowi rozdzielczości (przez interpolację) poprzednich klatek pozwala na lepsze dopasowanie najbliższego sąsiedztwa do danego wzorca. Na przykład, jeśli weźmiemy pod uwagę wzorzec $\mathbf{Y}_{off}$ i wzorzec $\mathbf{Y}_{off+}$ o tym samym kształcie, lecz położony o jeden piksel wyżej, to wektor utworzony jako średnia arytmetyczna obu tych wzorców będzie wektorem z obrazu o podwojonej rozdzielczości w pionie (interpolacja na bazie średniej arytmetycznej). Wartość przewidywana związana z tym wzorcem także będzie średnią arytmetyczną skojarzonych z nimi pikseli $P_{off}(0)$ oraz $P_{off+}(0)$. Pomysł ten został wykorzystany w metodzie Blend-24 przez trzykrotne rozszerzenie zbioru wzorców (dodanie interpolacji w pionie i pod kątem 45 stopni). Ponadto w subpredyktorze TCM znacznie zwiększono wielkość okna treningowego z $W = 3$ do $W = 15$, tworząc tym samym nową technikę TCM_+, którą wykorzystują metody Blend-24 i Blend-25.

9.8.3. Trzecia metoda mieszania

Kolejne udoskonalenie Blend-24 to trzecia metoda mieszania wzorowana na pracy [59]. Charakteryzuje się ona większą złożonością obliczeniową w porównaniu z dwiema poprzednimi, w których błędy otoczenia i-tego subpredyktora są wyznaczane, odpowiednio, za pomocą wzorów (9.1) i (9.11). Jednak dzięki jednoczesnemu zastosowaniu metod AVE--WLS istnieje możliwość wykonywania wielu wspólnych obliczeń, co pozwala zredukować wzrost całkowitego czasu kodowania w Blend-24.

Znane z Blend-20 dwie pierwsze metody mieszania do wyznaczania wag wykorzystują wartości m_i (w przypadku drugiej metody $m_i = 72$ w Blend-20 oraz $m_i = 96$ w Blend-24) kolejnych błędów predykcji z najbliższego sąsiedztwa według numeracji zgodnej z rys. 1.1. Przy czym w pierwszej metodzie każdy spubpredyktor ma własną wielkość otoczenia (patrz tab. 9.7), a w trzeciej metodzie wykorzystuje się 35 błędów $e^{(i)}_{off(j)}$ ($j = \{1, 2, \ldots, 35\}$) każdego i-tego subpredyktora znajdujących się w pewnym otoczeniu Q aktualnie kodowanego piksela.

Jest to ten sam obszar, który jest wykorzystywany w metodzie AVE-WLS (patrz rys. 6.1). Z obszaru Q wybierane są te błędy $e^{(i)}_{\mathrm{off}(j)}$, których najbliższe otoczenia (wzorce $\mathbf{Y}_{\mathrm{off}(j)}$ – patrz definicja wzorca w podrozdziale 6.3) są jak najbliższe otoczenia aktualnie kodowanego piksela $P(0)$. Miara Δ podobieństwa tych otoczeń została zdefiniowana wzorem (9.8), przy czym $m = r_{\max}$, gdzie $r_{\max}$ jest maksymalnym rzędem zdefiniowanym w metodzie AVE-WLS. Z obszaru Q dla każdego subpredyktora jest wyznaczanych 35 błędów $e^{(i)}_{\mathrm{off}(j)}$ o najmniejszych wartościach $\Delta^{(i)}_j$ i na ich podstawie jest obliczany błąd $E^{(3)}_i$ otoczenia i-tego subpredyktora w trzeciej technice mieszania:

$$E^{(3)}_i = 1 + \sum_{j=1}^{35} \left| e^{(i)}_{\mathrm{off}(j)} \right| . \tag{9.15}$$

Następnie wyznaczamy wartość w_i wagi subpredyktora:

$$w_i = \alpha_i \cdot \left(\frac{\delta_i}{E^{(3)}_i} \right)^4 , \tag{9.16}$$

po czym wagi są podstawiane do wzoru (9.5) w celu wyznaczenia znormalizowanych współczynników mieszania, z pomocą których uzyskujemy trzecią wartość przewidywaną, $\hat{x}_{(3)}$. Druga warstwa mieszania predykcyjnego (wykorzystująca zasady pierwszej metody mieszania), z trzema subpredyktorami: $\hat{x}_{(1)}$, $\hat{x}_{(2)}$, $\hat{x}_{(3)}$ i odpowiadającymi im błędami ich najbliższych otoczeń E_1, E_2, E_3, wyznaczanymi ze wzoru (9.1) ($m_1 = m_2 = m_3 = 18$), pozwala zdefiniować wartość predyktora głównego jako:

$$\hat{x} = \frac{\dfrac{41}{E_1^3} \cdot \hat{x}_{(1)} + \dfrac{23}{E_2^3} \cdot \hat{x}_{(2)} + \dfrac{32}{E_3^3} \cdot \hat{x}_{(3)}}{\dfrac{41}{E_1^3} + \dfrac{23}{E_2^3} + \dfrac{32}{E_3^3}} . \tag{9.17}$$

9.8.4. Dobór fazy obrotu

Następnym udoskonaleniem względem poprzednich metod jest wstępne przeszukiwanie w celu znalezienia najlepszej spośród ośmiu faz obrotu. Wykorzystuje się tu możliwość bezstratnego obrotu obrazu przed rozpoczęciem kodowania o wielokrotność 90°, co daje cztery różne ustawienia obrazu (fazy obrotu). Jeśli do tego dołączymy możliwość odbicia lustrzanego, to liczba faz obrotu zwiększa się dwukrotnie. Po znalezieniu najlepszej fazy obrotu jej numer jest zapisywany w nagłówku na trzech bitach. Aby wyznaczyć numer fazy, ze względu na długi czas modelowania, należało znaleźć uproszczony sposób wstępnej selekcji fazy obrotu. Na tym etapie metodę mieszanego wyznaczania predyktora głównego zastą-

piono kombinacją liniową prostych metod predykcyjnych CoBALP$_+$ (patrz podrozdział 5.4) oraz GAP$_+$ (patrz podrozdział 3.1.2):

$$\hat{x} = 0{,}7\hat{x}_{CoBALP_+} + 0{,}3\hat{x}_{GAP_+} . \tag{9.18}$$

W przypadku metody CoBALP$_+$ aktywna jest też jej korekcja metodą NLMS. Łączny czas poszukiwania fazy obrotu wynosi dla obrazu Lennagrey 32,8 s.

9.8.5. Dobór parametrów kodera arytmetycznego

Analizując rozkłady prawdopodobieństwa wartości *k* (skwantyzowanych błędów predykcji) w każdym z 16 kontekstów (patrz rys. od 8.1 do 8.3), można zauważyć, że ze względu na cechy charakterystyczne skojarzonych z nimi rozkładów, warto dobrać indywidualne pary wektorów **T** i **q** (patrz podrozdział 8.3) zawierające progi kwantyzacji błędów predykcji i wartości pomocnicze. Dla kontekstów o numerach od 5 do 15 ustalono, że para wektorów może być wspólna, natomiast każdy z kontekstów o numerach od 0 do 4 ma swoją charakterystyczną parę **T** i **q**.

W metodzie Blend-24 dominującą część czasu poświęconego na kompresję obrazu zabiera etap modelowania danych, związany z wyznaczaniem wartości przewidywanych. Znając ten fakt i wykorzystując gotową już mapę błędów predykcji, po etapie dekompozycji danych można dokonać wielokrotnego kodowania (maksymalnie 140 pomiarów) tych danych przy niewielkim wzroście całkowitego czasu kompresji. Pozwala to na dobór indywidualnych dla danego obrazu parametrów kodera arytmetycznego opisanego w rozdziale 8.

Na początku jest wyznaczana maksymalna wartość bezwzględna błędu predykcji, co może umożliwić ograniczenie zakresu kodowanych wartości; rozpatrywane są następujące górne granice: 255, 191, 159, 127, 95, 63, 51, 37, 29, 20. Każdej z tych granic przyporządkowano w każdym kontekście od jednej do trzech par wektorów **T** i **q**. W przypadku ograniczenia zakresu kodowanych liczb |*e*| można zmniejszyć też liczbę symboli kodowych *k* powstałych po kwantyzacji błędu |*e*|, co sprzyja wzrostowi efektywności kodowania arytmetycznego. Na przykład (dla kontekstów o numerach od 5 do 15) przy maksymalnym błędzie wynoszącym 51 zakres wartości *k* można zmniejszyć z 18 do 10. Szczegółowo zawartość wszystkich par wektorów **T** i **q** (wspólnych dla kontekstów o numerach od 5 do 15) umieszczono w tab. 9.8. Kodowanie jest wykonywane dla kolejnych par wektorów, które spełniają warunek, że maksymalny błąd bezwzględny jest mniejszy (lub równy) od wartości maksymalnej ustalonej dla danej pary (patrz druga kolumna tab. 9.8). Następnie wybiera się parę wektorów **T** i **q**, dla których uzyskano najkrótszą średnią bitową.

Drugim krokiem jest sprawdzenie, z jaką częstotliwością dokonywać okresowego zapominania wektorów liczebności wartości skwantyzowanych błędów predykcji (patrz podrozdział 8.1). Sprawdzamy parametr *s* ze zbioru {12, 13, 14} i zapamiętujemy w nagłówku dwubitowy indeks wartości *s*, dla której uzyskujemy najniższą średnią bitową.

Tab. 9.8. Zestaw par wektorów używanych do kwantyzacji błędów predykcji

Numer zestawu	Wartość maksymalna błędu $\|e\|$	Liczba stanów wartości k	Para wektorów
1	255	18	**T** = {0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 2, 2, 3, 5, 6, 7} **q** = {0, 0, 0, 0, 0, 0, 0, 0, 8, 10, 12, 14, 16, 20, 24, 32, 64, 128}
2	191	18	**T** = {0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 2, 2, 3, 5, 6, 6} **q** = {0, 0, 0, 0, 0, 0, 0, 0, 8, 10, 12, 14, 16, 20, 24, 32, 64, 128}
3	191	17	**T** = {0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 2, 2, 3, 5, 7} **q** = {0, 0, 0, 0, 0, 0, 0, 0, 8, 10, 12, 14, 16, 20, 24, 32, 64}
4	159	18	**T** = {0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 2, 2, 3, 5, 5, 6} **q** = {0, 0, 0, 0, 0, 0, 0, 0, 8, 10, 12, 14, 16, 20, 24, 32, 64, 96}
5	159	17	**T** = {0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 2, 2, 3, 6, 6} **q** = {0, 0, 0, 0, 0, 0, 0, 0, 8, 10, 12, 14, 16, 20, 24, 32, 96}
6	159	15	**T** = {0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 3, 3, 7} **q** = {0, 0, 0, 0, 0, 0, 0, 0, 8, 10, 12, 14, 16, 24, 32}
7	127	17	**T** = {0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 2, 2, 3, 5, 6} **q** = {0, 0, 0, 0, 0, 0, 0, 0, 8, 10, 12, 14, 16, 20, 24, 32, 64}
8	127	18	**T** = {0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 2, 2, 3, 5, 6} **q** = {0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 10, 12, 14, 16, 20, 24, 32, 64}
9	95	15	**T** = {0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 3, 3, 6} **q** = {0, 0, 0, 0, 0, 0, 0, 0, 8, 10, 12, 14, 16, 24, 32}
10	95	13	**T** = {0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 2, 4, 6} **q** = {0, 0, 0, 0, 0, 0, 0, 0, 8, 10, 12, 16, 32}
11	63	11	**T** = {0, 0, 0, 0, 0, 0, 1, 2, 2, 4, 5} **q** = {0, 0, 0, 0, 0, 0, 6, 8, 12, 16, 32}
12	63	14	**T** = {0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 2, 3, 3, 5} **q** = {0, 0, 0, 0, 0, 0, 0, 0, 8, 10, 12, 16, 24, 32}
13	51	10	**T** = {0, 0, 0, 0, 0, 0, 1, 2, 3, 5} **q** = {0, 0, 0, 0, 0, 0, 6, 8, 12, 20}
14	37	10	**T** = {0, 0, 0, 0, 0, 0, 2, 2, 3, 4} **q** = {0, 0, 0, 0, 0, 0, 6, 10, 14, 22}
15	29	8	**T** = {0, 0, 0, 0, 0, 0, 3, 4} **q** = {0, 0, 0, 0, 0, 0, 6, 14}
16	20	6	**T** = {0, 0, 0, 1, 3, 3} **q** = {0, 0, 0, 3, 5, 13}

Kolejny zestaw pomiarów polega na iteracyjnym doborze (zgrubnym i szczegółowym) trzech współczynników wagowych: h_1, h_2, h_3. Zmodyfikowane zostały też inne współczynniki występujące we wzorach od (8.3) do (8.7) omówionych w podrozdziale 8.2. W przypadku Blend-24 rozszerzony algorytm wyznaczania wartości ω jest następujący. Najpierw jest obliczana wartość ω_1:

$$\omega_1 = \max\{2{,}3|e(1)|,\ 2|e(2)|,\ 1{,}6|e(4)|,\ 0{,}95(|e(3)|+|e(4)|),\ 1{,}25(|e(5)|+|e(10)|),$$
$$1{,}3|e(3)|,\ 1{,}375(|e(1)|+|e(2)|),\ 0{,}4(|e(6)|+|e(7)|),\ 0{,}4(|e(8)|+|e(9)|)\}. \qquad (9.19)$$

Parametr ω_2 jest wyznaczany ze wzoru (8.4), następnie ω_3 jako:

$$\omega_3 = \max\{2{,}1h_3\omega_1, 10{,}2\omega_2\}, \tag{9.20}$$

a ω_4 jako:

$$\omega_4 = \max\{|P(1)-P(3)|, |P(2)-P(4)|, 1{,}1|P(1)-P(2)|, 0{,}7|P(2)-P(3)|, \\ 0{,}9|P(1)-P(4)|, 0{,}9|P(3)-P(4)|\}. \tag{9.21}$$

Na podstawie parametrów ω_3 oraz ω_4 możemy wyznaczyć końcową wartość ω:

$$\omega = h_1 \cdot \omega_3 + \frac{h_2}{3d+1} \cdot \omega_4, \tag{9.22}$$

gdzie parametr d jest dopuszczalną wartością błędu w trybie prawie bezstratnym (patrz podrozdział 10.2). W trybie bezstratnym $d = 0$.

Ponadto dokonuje się pomiarów, zmniejszając liczbę t kontekstów określających odrębne rozkłady prawdopodobieństw (patrz podrozdział 8.2). Wartość t zmniejszana jest, począwszy od 16, tak długo, aż uzyskamy najkrótszy plik wynikowy, a kolejna próba zmniejszania t spowoduje wzrost średniej bitowej.

Ostatni pomiar polega na sprawdzeniu wpływu podwojenia liczby kontekstów skojarzonych z arytmetycznym koderem bitu znaku (patrz podrozdział 8.4). Wprowadzając dodatkowy, piąty bit do liczby tworzącej numer kontekstu, uzyskujemy 32 konteksty bitu znaku. Bit ten uzyskuje wartość 1, gdy $|e(0)| > 2$, w przeciwnym razie ma wartość 0.

Wszystkie te kroki można uznać za autorskie rozwiązania zwiększające efektywność zaprojektowanego kodera, który został opisany w rozdziale 8. W porównaniu z Blend-20 w metodzie Blend-24, w wyniku powyższych poszukiwań parametrów kodera arytmetycznego, łączna wielkość informacji nagłówkowych przekazywanych do dekodera zwiększa się zaledwie o 25 bitów, a średni czas tych poszukiwań dla obrazu Lennagrey wynosi 43,4 s.

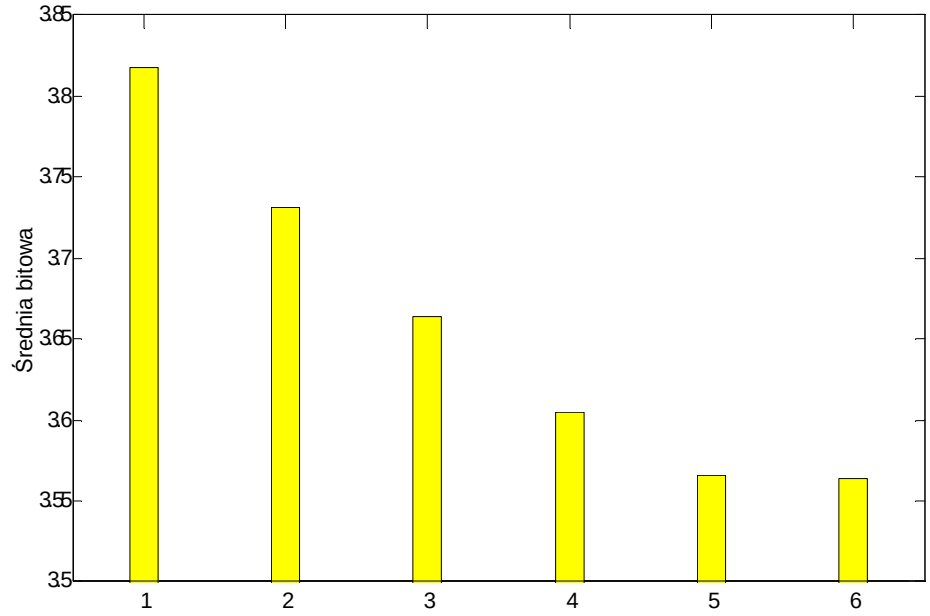
9.9. Analiza efektywności proponowanych metod

W metodzie Blend-24 łączny czas kodowania wzrósł w porównaniu z Blend-20 ze 107,3 do 1121,6 s, co stanowi zaledwie 91,3% czasu kodowania metodą MRP 0.5. Średnia bitowa z 45 pomiarów zmniejszyła się z 3,88627 w metodzie Blend-20 do 3,84412 w metodzie Blend-24.

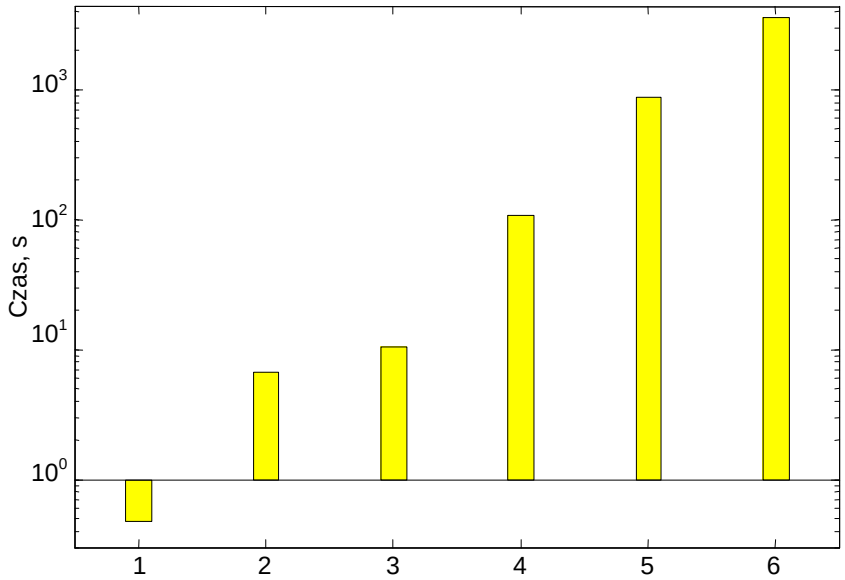
Metoda Blend-25 jest wersją Blend-24 wzbogaconą o dodatkowy subpredyktor AdNN+, oparty na sieciach neuronowych (patrz rozdział 7). Użycie tak złożonego obliczeniowo subpredyktora wpłynęło na poprawę średniej bitowej w marginalnym stopniu, bo zaledwie średnio o 0,002 bitu na piksel, natomiast czas wzrósł znacząco z 1121,6 do 3635 s.

Na rysunku 9.5 przedstawiono porównanie średnich bitowych dla dziewięciu standardowych obrazów testowych uzyskanych metodami Blend-13 (1), Blend-17 (2), Blend-19 (3), Blend-20 (4), Blend-24 (5) i Blend-25 (6), gdzie liczba w nawiasie określa numer słupka

reprezentującego daną metodę. Czasy kodowania tymi metodami obrazu Lennagrey przedstawia rys. 9.6, przy czym należy zwrócić uwagę na logarytmiczną skalę reprezentacji wyników.



Rys. 9.5. Średnie bitowe dla dziewięciu standardowych obrazów testowych, uzyskane metodami Blend-13 (1), Blend-17 (2), Blend-19 (3), Blend-20 (4), Blend-24 (5) i Blend-25 (6)



Rys. 9.6. Czas kodowania obrazu Lennagrey metodami Blend-13 (1), Blend-17 (2), Blend-19 (3), Blend-20 (4), Blend-24 (5) i Blend-25 (6)

Analizując wartości średnich bitowych dla zestawu dziewięciu standardowych obrazów testowych, można stwierdzić, że proponowana metoda Blend-24 daje o 12,18% niższą średnią w porównaniu z JPEG-LS. W tabeli 9.9 zaprezentowano dla tego zestawu wyniki kilku najlepszych metod znanych z literatury, a w tab. 9.10 znajduje się zestawienie zbiorcze wszystkich dostępnych rezultatów, jakie udało się zebrać.

Tab. 9.9. Pomiar wartości średnich bitowych dla standardowych obrazów testowych

Obrazy	BMF [78]	TMW^{LEGO} [87]	MRP 0.5 [78]	Multi-WLS [148]	Blend-20	Blend-24	Blend-25
Balloon	2,649	2,60	2,579	2,60	2,566	2,539	2,546
Barb	3,959	3,84	3,815	3,75	3,768	3,689	3,686
Barb2	4,276	4,24	4,216	4,18	4,175	4,118	4,119
Board	3,331	3,27	3,268	3,27	3,272	3,237	3,233
Boats	3,593	3,53	3,536	3,53	3,520	3,480	3,482
Girl	3,517	3,47	3,465	3,45	3,449	3,408	3,408
Gold	4,238	4,22	4,207	4,20	4,185	4,158	4,156
Hotel	4,066	4,01	4,026	4,01	4,007	3,966	3,961
Zelda	3,549	3,50	3,495	3,51	3,498	3,479	3,477
Średnia	3,686	3,631	3,623	3,611	3,605	3,564	3,562

Tab. 9.10. Pomiar wartości średnich bitowych dla standardowych obrazów testowych – zestawienie zbiorcze

Metoda	Balloon	Barb	Barb2	Board	Boats	Girl	Gold	Hotel	Zelda	Średnia
PNG-crush [105]	3,253	5,214	5,147	3,982	4,287	4,314	4,677	4,809	4,128	4,423
Sunset [82]	2,89	4,68	4,77	3,73	4,01	3,91	4,56	4,46	3,81	4,091
JPEG-LS [107]	2,889	4,690	4,684	3,674	3,930	3,922	4,475	4,378	3,884	4,058
UCM [82]	2,81	4,44	4,57	3,57	3,85	3,81	4,45	4,28	3,80	3,889
HBB [103]	2,80	4,28	4,48	3,54	3,80	3,74	4,37	4,27	3,72	3,889
$CoBALP_{max}$ [107]	2,853	4,176	4,440	3,492	3,780	3,696	4,382	4,219	3,749	3,865
CALIC [149]	2,78	4,31	4,46	3,51	3,78	3,72	4,35	4,18	3,69	3,864
WinRK 3.0b	2,766	4,337	4,493	3,425	3,763	3,751	4,331	4,145	3,747	3,862
CBPC [59]	2,78	4,14	4,47	3,49	3,78	3,70	4,38	4,23	3,72	3,854
Lee [66]	2,79	4,20	4,47	3,50	3,76	3,70	4,35	4,24	3,68	3,854
P13 [28]	2,74	4,29	4,47	3,48	3,75	3,67	4,33	4,19	3,68	3,844
LAT-RLMS [75]	2,75	4,15	4,45	3,48	3,74	3,68	4,34	4,21	3,61	3,823
ALPC [88]	2,74	4,00	4,41	3,48	3,77	3,68	4,39	4,33	3,60	3,822
Blend-13	2,746	4,172	4,441	3,465	3,716	3,640	4,323	4,189	3,673	3,818
AdNN [75]	2,79	4,03	4,45	3,46	3,72	3,63	4,34	4,19	3,61	3,802
$ALCM_{+}$	2,735	4,175	4,372	3,418	3,684	3,640	4,296	4,199	3,599	3,791
APC-A [26]	2,73	4,04	4,40	3,43	3,70	3,61	4,30	4,15	3,63	3,777
Blend-17	2,671	4,049	4,353	3,364	3,633	3,568	4,258	4,078	3,606	3,731
OLS [148]	2,690	3,939	4,310	3,388	3,638	3,576	4,273	4,162	3,549	3,725
$CoBALP_{+}$	2,677	4,043	4,316	3,367	3,627	3,555	4,266	4,101	3,564	3,724
GLICBAWLS [85]	2,640	3,916	4,318	3,392	3,628	3,565	4,276	4,177	3,537	3,717
M-LMS	2,672	4,037	4,305	3,357	3,620	3,547	4,258	4,103	3,559	3,717
BMF [78]	2,649	3,959	4,276	3,331	3,593	3,517	4,238	4,066	3,549	3,686
$AdNN_{+}$	2,647	3,868	4,283	3,330	3,607	3,528	4,238	4,086	3,530	3,680
Blend-19	2,630	3,934	4,238	3,310	3,568	3,500	4,222	4,038	3,538	3,664
TMW^{LEGO} [87]	2,60	3,84	4,24	3,27	3,53	3,47	4,22	4,01	3,50	3,631
MRP 0.5 [78]	2,579	3,815	4,216	3,268	3,536	3,465	4,207	4,026	3,495	3,623
Multi-WLS [148]	2,60	3,75	4,18	3,27	3,53	3,45	4,20	4,01	3,51	3,611
Blend-20	2,566	3,768	4,175	3,272	3,520	3,449	4,185	4,007	3,498	3,605
AVE-WLS1-NLMS+	2,570	3,744	4,169	3,270	3,524	3,441	4,196	4,019	3,504	3,604
Blend-24	2,539	3,689	4,118	3,237	3,480	3,408	4,158	3,966	3,479	3,564
Blend-25	2,539	3,686	4,117	3,234	3,478	3,408	4,156	3,961	3,477	3,562

Uzyskanie bardziej obiektywnych wyników było możliwe jedynie dla kilku dostępnych programów i polegało na pomiarze średnich bitowych dla znacznie szerszego zbioru 45 obrazów testowych (patrz tab. 9.11). Wśród nich znalazło się kilka obrazów o mniejszej niż 200 liczbie kolorów, co pokazuje, że jedynie WinRK 3.0b oraz metody Blend-24 i Blend-25 potrafiły wykryć ten fakt i wykorzystać go do efektywniejszego kodowania (np. obrazy Frog – 102 kolory, Tank – 138 kolorów). Ponadto, aby sprawdzić możliwości testowanych metod nie tylko do kodowania obrazów naturalnych, wśród testowanych znalazły się dwa obrazy z kategorii sztucznie generowanych: Noisesquare (o dużym poziomie zaszumienia) oraz Shapes (o płynnych przejściach barw z wyróżnionymi elementami krawędziowymi). Dla obu tych obrazów najniższą średnią uzyskano dla WinRK 3.0b, programu służącego do kompresji danych ogólnego przeznaczenia (nie tylko do kodowania obrazów). Jeśli zrezygnować z tych czterech wymienionych wcześniej obrazów, okaże się, że WinRK 3.0b wyraźnie ustępuje metodzie MRP. Jeśli nie uwzględnimy metod opracowanych w tej pracy, to można zauważyć, iż MRP okazał się efektywniejszy w przypadku aż 33 na 45 testowanych obrazów.

Z porównania metody Blend-24 i najlepszych jej konkurentów wynika, że dla MRP uzyskano niższą średnią bitową jedynie w trzech, a dla WinRK 3.0b w sześciu przypadkach. Średnia bitowa także zdecydowanie wskazuje na dominację metod Blend-24 i Blend-25.

Porównanie wybranych metod dla drugiego zestawu testowego zawiera tab. 9.12.

Tab. 9.11. Porównanie Blend-24 i Blend-25 z innymi znanymi metodami

Obrazy	PNG-Crush	$CoBALP_{max}$	GLICBAWLS	BMF	MRP	WinRK 3.0b	Blend-24	Blend-25
Aerial	5,41376	4,70953	4,79489	4,53699	4,46072	4,68707	4,37657	4,37640
Airfield	5,82443	5,48029	5,44501	5,40369	5,32312	4,90930	4,75550	4,75294
Airplane	4,24405	3,76077	3,66830	3,60193	3,59097	3,66364	3,55478	3,54803
Baboon	6,23380	5,91769	5,66595	5,71399	5,66339	5,84299	5,61974	5,61786
Barbara	5,43671	4,37051	4,07889	4,17261	4,00476	4,58087	3,80016	3,79766
Boat	5,09241	4,59860	4,46191	4,45020	4,38779	4,62534	4,32162	4,31946
Bridge	4,25705	5,43445	5,38998	5,31580	5,27969	3,40729	3,29893	3,29919
Couple	4,88821	4,58615	4,47464	4,45386	4,42477	4,02838	3,99253	3,99300
Crowd	4,51071	3,77643	3,75247	3,57593	3,54932	3,53369	3,46811	3,46652
Elaine	5,18347	4,76852	4,58109	4,58960	4,30060	4,79608	4,20606	4,20381
Finger	5,81857	5,40906	5,23102	5,22998	5,17783	5,39871	5,12675	5,12457
Frog	3,85339	5,97952	5,93512	5,93652	5,90344	2,62317	4,96292	4,96014
Goldhill	4,89267	4,61362	4,50192	4,48181	4,45114	4,59174	4,39223	4,39017
Harbour	4,85431	4,42517	4,42792	4,26660	4,17395	4,26013	4,12811	4,12774
Lax	5,98752	5,67267	5,59326	5,54700	5,50629	5,56250	5,45569	5,45418
Lennagrey	4,60120	4,09262	3,90125	3,92871	3,88947	4,09488	3,85028	3,84875
$Lena_{TMW}$	4,91208	4,46762	4,29486	4,31702	4,28012	4,47571	4,24190	4,23900
Man	4,92224	4,38654	4,30075	4,22266	4,21222	4,29462	4,13459	4,13482
Peppers	4,91071	4,37213	4,24612	4,24146	4,19843	4,44754	4,11500	4,11178
Sailboat	5,15198	4,66296	4,53671	4,48401	4,41739	4,61682	4,36459	4,36149
Seismic	3,13849	2,70041	2,25027	2,40662	2,04944	2,71143	1,98443	1,98477
Shapes	1,17764	1,24393	2,29117	1,17554	0,70081	0,52853	0,82306	0,82428
Tank	4,88257	4,76971	4,67889	4,65259	4,63702	3,81491	3,74431	3,74335
Truck	4,75665	4,49835	4,45926	4,39551	4,39966	3,88278	3,98652	3,98493

Tab. 9.11. Porównanie Blend-24 i Blend-25 z innymi znanymi metodami (cd.)

Obrazy	PNG-Crush	$CoBALP_{max}$	GLICBAWLS	BMF	MRP	WinRK 3.0b	Blend-24	Blend-25
Woman1	4,98132	4,52441	4,36124	4,38708	4,32474	3,96823	3,71192	3,71136
Woman2	3,68335	3,23935	2,93158	3,01282	2,92432	3,16534	2,93536	2,93496
Balloon	3,25293	2,85316	2,63958	2,64931	2,57897	2,76626	2,53949	2,53911
Barb	5,21418	4,17622	3,91647	3,95903	3,81485	4,33727	3,68851	3,68613
Barb2	5,14697	4,44049	4,31794	4,27600	4,21566	4,49265	4,11782	4,11747
Board	3,98175	3,49171	3,39169	3,33125	3,26792	3,42456	3,23693	3,23377
Boats	4,28738	3,77998	3,62791	3,59259	3,53576	3,76296	3,48044	3,47850
Girl	4,31431	3,69587	3,56493	3,51744	3,46474	3,75102	3,40810	3,40750
Gold	4,67681	4,38171	4,27629	4,23796	4,20702	4,33088	4,15763	4,15575
Hotel	4,80910	4,21929	4,17735	4,06582	4,02623	4,14456	3,96601	3,96146
Zelda	4,12377	3,74917	3,53706	3,54877	3,49483	3,74705	3,47925	3,47705
Bridge256	6,01770	5,73694	5,62585	5,56689	5,54858	5,67273	5,48317	5,48215
Camera	4,76941	4,20117	4,20825	4,06006	3,94946	4,05005	3,91841	3,91840
Couple256	4,15649	3,64282	3,54285	3,44775	3,38965	3,53333	3,34174	3,34383
Earth	3,37463	3,16211	3,26257	3,08496	3,03723	2,53333	2,77161	2,77153
Elif	3,60498	3,05811	2,73901	2,79443	2,62134	2,94653	2,52121	2,52126
Noisesquare	5,76636	5,59338	5,41467	5,29785	5,26819	5,09900	5,21141	5,20515
Omaha	6,07300	6,13977	6,30408	6,07324	5,54504	4,31409	5,85039	5,84940
Sena	3,83704	3,23523	2,91931	2,98975	2,80164	3,19373	2,71310	2,71330
Sensin	4,25110	3,51807	3,22705	3,28662	3,09192	3,56567	2,96457	2,96547
Sinan	3,96106	3,26416	2,96191	3,04053	2,88147	3,27771	2,78413	2,78093
Średnia	4,64952	4,28445	4,17576	4,11824	4,02160	3,98789	3,84412	3,84265

Tab. 9.12. Pomiar wartości średnich bitowych dla standardowych obrazów testowych – zestaw drugi

Obrazy	Blend-24	MRP [76]	xMRP [43]	BMF [76]	TMW [84]	SWAP [54]	RALP [53]	GLICBA-WLS [83]
Camera	3,91888	3,949	3,971	4,060	4,098	4,39	4,24	4,208
Couple256	3,34244	3,388	3,389	3,448	3,446	3,75	3,63	3,543
Noisesquare	5,21117	5,270	5,301	5,298	5,542	5,16	5,37	5,415
Airplane	3,55497	3,591	5,590	3,602	3,601	3,58	3,71	3,668
Baboon	5,62015	6,663	5,662	5,714	5,738	5,86	5,81	5,666
LenaTMW	4,24449	4,280	4,273	4,317	4,300	4,35	4,35	4,295
Lennagrey	3,85348	3,889	3,885	3,929	3,908	3,95	3,95	3,901
Peppers	4,11530	4,199	4,208	4,241	4,251	4,25	4,27	4,246
Shapes	0,82380	0,685	0,769	0,730	0,740	1,56	1,52	2,291
Balloon	2,54753	2,579	2,613	2,649	2,649	2,49	2,55	2,640
Barb	3,68868	3,815	3,817	3,959	4,084	4,12	4,12	3,916
Barb2	4,12063	4,216	4,226	4,276	4,378	4,55	4,51	4,318
Gold	4,15769	4,207	4,216	4,238	4,266	4,30	4,32	4,276
Średnia	3,78456	3,826	3,840	3,882	3,923	4,024	4,027	4,030

10. Zastosowania kodowania mieszanego

10.1. Wprowadzenie

Rozdział ten jest poświęcony przedstawieniu kilku sposobów wykorzystania omówionych w tej pracy metod mieszania predykcyjnego. Szczególną uwagę poświęcono najbardziej praktycznej z metod[1], a mianowicie Blend-13, której prosty algorytm może być w łatwy sposób zaimplementowany sprzętowo, a realizacja programowa charakteryzuje się krótkim czasem kodowania (patrz podrozdział 9.3). Rozszerzeniem metod bezstratnych jest technika prawie bezstratnej kompresji obrazów, omówiona w podrozdziale 10.2. Oprócz sugerowanych głównych zastosowań do kodowania obrazów naturalnych metody mieszania predykcyjnego możemy wykorzystać do kompresji obrazów astronomicznych, a także zdjęć satelitarnych, co pokazano w podrozdziale 10.3. Metodę mieszania predykcyjnego można wykorzystać także do bezstratnego i prawie bezstratnego kodowania sekwencji wideo (patrz podrozdział 10.4). W kolejnych punktach przedstawiono wyniki badań na temat możliwości zastosowania transformacji do wzrostu efektywności kompresji obrazów kolorowych, a także wykazano dla tej kategorii obrazów wysoką wydajność metody mieszania predykcyjnego.

10.2. Kodowanie w trybie prawie bezstratnym

Uzyskanie wysokiego stopnia kompresji jest możliwe dzięki kodowaniu nieodwracalnemu. Istnieje wiele metod kompresji stratnej wykorzystującej między innymi transformatę DCT (np. JPEG) czy transformatę falkową (np. JPEG2000). Metody te umożliwiają regulację stopnia kompresji, jednak ich podstawowe implementacje nie pozwalają decydować o maksymalnym możliwym błędzie. Możliwość taką daje tryb prawie bezstratny (ang. *near-lossless*), dzięki któremu możemy zdefiniować maksymalną dopuszczalną wartość bezwzględną d różnicy wartości pikseli między obrazem oryginalnym i zdekodowanym. Pomysł ten wykorzystuje się często do zapisu obrazów w sposób wizualnie bezstratny. W tym przypadku akceptowalny poziom błędów jest określany przez grupę ekspertów. Najczęściej tryb ten jest oparty na kodowaniu predykcyjnym, w którym zestaw błędów predykcji e jest poddawany kwantyzacji w sposób następujący [145]:

$$\hat{e} = \begin{cases} \left\lfloor \dfrac{e+d}{2d+1} \right\rfloor, & \text{gdy } e \geq 0, \\ \left\lfloor \dfrac{e-d}{2d+1} \right\rfloor, & \text{gdy } e < 0. \end{cases} \tag{10.1}$$

[1] Większość badań wykonano w latach 2007–2010 w ramach grantu własnego „Algorytmy bezstratnej i prawie bezstratnej kompresji danych multimedialnych dedykowane architekturze Networks on Chip". Kierownikiem projektu był dr inż. Piotr Dziurzański.

Wartość pierwotną błędu predykcji możemy odtworzyć z dokładnością ± *d*, uzyskując:

$$\hat{e} = \hat{e} \cdot (2d + 1). \tag{10.2}$$

Przy $d = 0$ otrzymujemy tryb bezstratny.

W pracy [41] zaprezentowano analizę kilku podejść do kodowania prawie bezstratnego. Jedną z propozycji jest kwantyzacja wartości pikseli (zamiast wartości błędów predykcji):

$$\overline{x} = \left\lfloor \frac{x + d}{2d + 1} \right\rfloor. \tag{10.3}$$

W takiej sytuacji wartość pierwotną piksela możemy odtworzyć z dokładnością ± *d*, uzyskując:

$$\hat{x} = \overline{x} \cdot (2d + 1). \tag{10.4}$$

Metoda kodowania prawie bezstratnego może być wykorzystywana w tych przypadkach kompresji obrazów medycznych, w których eksperci powinni ustalić tolerancję błędu *d*, niemającą wpływu na ocenę diagnostyczną danego typu obrazów medycznych (czasem określa się taki rodzaj kompresji mianem *visually lossless* [7]), a także wszędzie tam, gdzie jest wymagany zapis sekwencji wideo o bardzo dużej liczbie klatek, a dopuszcza się niewielką tolerancję błędu. Kodowanie prawie bezstratne można połączyć np. z metodą bezstratnego kodowania, w której tryb bezstratny będzie aktywowany wyłącznie na fragmencie obrazu najbardziej istotnym dla użytkownika – obszarze zainteresowania ROI (ang. *Region of Interest*) [106].

W pracy [144] zaproponowano system służący do sprzętowej kompresji obrazów medycznych – endoskopowych zdjęć kolorowych o trzech składowych *RGB* o ośmiobitowej głębi i rozdzielczości maksymalnej 640 × 480 pikseli (przy kodowaniu dwóch klatek na sekundę). Taka realizacja sprzętowa jest jednym z przykładów, gdzie prostota algorytmu i tryb prawie bezstratny stają się koniecznością. Wynika to bowiem z potrzeby miniaturyzacji kapsuły – bezprzewodowego urządzenia zawierającego kamerę z systemem oświetlenia, układ kodowania oraz nadajnik i baterię zasilającą. Rozmiar kapsuły, którą połyka pacjent, jest mniejszy od 9 × 20 mm. Głównym problemem jest ilość energii potrzebna do zasilania układu, w tym do oświetlenia i pracy nadajnika. Oszczędność energii uzyskano przez znaczne zmniejszenie ilości transmitowanych danych, co było możliwe dzięki zastosowaniu prawie bezstratnej kompresji obrazu.

W systemie tym zastosowano kompresję prawie bezstratną o maksymalnym błędzie $d = 2$, co zostało zaproponowane jako kompromis między jakością obrazu a uzyskanym stopniem kompresji. Projektanci postanowili dokonać kwantyzacji bezpośrednio składowych obrazu (poddanych wcześniej odwracalnej filtracji) zamiast kwantyzacji błędu predykcji. Zaimplementowano także możliwość użycia bezstratnej kompresji w obszarze zainteresowania (ROI).

Istnieje wiele opracowań dotyczących prawie bezstratnej kompresji obrazów. Do najistotniejszych możemy zaliczyć [5, 11, 13, 20, 41, 46, 61, 84, 85, 139, 142, 144, 145], z części tych prac zaczerpnięto do celów porównawczych wyniki pomiarów średnich bitowych dla obrazów testowych.

Opisane w rozdziale 9 autorskie metody mieszania predykcyjnego są także przystosowane do kodowania w trybie prawie bezstratnym. W tabeli 10.1 porównano wyniki LOCO-I oraz rozwinięcia metody TMW do trybu prawie bezstratnego [84] z szybką, a zarazem efektywną metodą Blend-13 dla dopuszczalnych wartości błędów $d = 2$ oraz $d = 10$. Najszerszy zestaw porównawczy udało się zgromadzić dla wartości $d = 1$ (patrz tab. 10.2). Wynika z niego, że spośród dotychczas opublikowanych metod dwie najefektywniejsze to GLICBAWLS oraz BMF, przy czym dla większych d cecha ta jest także zachowana. Dlatego właśnie wyniki otrzymane tymi metodami wzięto pod uwagę przy porównaniu z autorskimi propozycjami Blend-13 (krótki czas kodowania), Blend-20 (długi czas kodowania), Blend-24 i Blend-25 (bardzo długi czas kodowania). Porównania dla wartości $d = \{1, 2, 3, 5, 7, 10\}$ zawierają tabele od 10.3 do 10.8, w których przedstawiono wyniki średnich bitowych dla szerokiego zestawu 45 obrazów testowych. W przypadku $d = 1$ metoda Blend-24 daje średnio o 17,74% krótsze pliki w porównaniu z JPEG-LS.

Tab. 10.1. Porównanie średnich bitowych przy dopuszczalnej wartości błędu $d = \{2, 10\}$

	$d = 2$			$d = 10$		
Image	LOCO-I v.0.90N	TMW (tryb 3)	Blend-13	LOCO-I v.0.90N	TMW (tryb 4)	Blend-13
Balloon.y	1,242	0,90	0,905	0,49	0,32	0,198
Boats.y	1,902	1,65	1,646	0,78	0,66	0,546
Gold.y	2,333	2,19	2,166	0,99	0,81	0,710
Airplane	1,84	1,64	1,660	0,72	0,57	0,517
Baboon	3,72	3,49	3,481	1,91	1,68	1,656
Lennagrey	2,09	1,83	1,855	0,93	0,55	0,516
Peppers	2,29	2,09	2,096	0,93	0,64	0,558
Shapes	0,79	0,75	0,709	0,47	0,58	0,320
Bridge256	3,49	3,38	3,325	1,73	1,63	1,570
Camera256	2,28	2,08	2,024	0,96	0,90	0,795
Couple256	1,82	1,60	1,606	0,86	0,70	0,559
Średnia	2,163	1,964	1,952	0,979	0,822	0,722

Tab. 10.2. Porównanie średnich bitowych przy dopuszczalnej wartości błędu $d = 1$

Obrazy	JPEG-LS [136]	Sunset [145]	BAROLTO [145]	ASBOSC [46]	PNLIC [61]	GLICBA WLS [85]	BMF	Blend-24
Balloon	1,465	1,45	1,48	1,457	1,554	1,321	1,323	1,254
Barb1	3,149	3,10	2,88	2,868	3,093	2,428	2,475	2,223
Barb2	3,174	3,17	3,06	3,012	3,258	2,813	2,783	2,628
Board	2,203	2,22	2,09	2,033	2,234	1,938	1,884	1,802
Boats	2,478	2,48	2,39	2,306	2,554	2,168	2,139	2,037
Girl	2,446	2,38	2,28	2,258	2,500	2,107	2,070	1,973
Gold	2,996	3,06	2,92	2,904	3,080	2,754	2,740	2,662
Hotel	2,873	2,94	2,75	2,719	3,043	2,660	2,559	2,459
Zelda	2,375	2,26	2,25	2,221	2,264	2,048	2,070	2,005
Średnia	2,573	2,56	2,46	2,420	2,260	2,249	2,227	2,116

Tab. 10.3. Porównanie metod $CoBALP_{max}$, GLICBAWLS i BMF z autorskimi Blend-13, Blend--20, Blend-24 i Blend-25 przy $d = 1$

Obrazy	$CoBALP_{max}$	Blend-13	GLICBAWLS	BMF	Blend-20	Blend-24	Blend-25
Aerial	3,25272	3,08078	3,27167	3,05237	2,96064	2,90316	2,90415
Airfield	3,94489	3,86434	3,88931	3,89539	3,78739	3,73405	3,73103
Airplane	2,44287	2,23898	2,21619	2,14832	2,12761	2,10864	2,10396
Baboon	4,34439	4,20175	4,09189	4,13892	4,07256	4,04440	4,04320
Barbara	2,89307	2,87958	2,56845	2,66504	2,42119	2,34564	2,34079
Boat	3,08087	3,00255	2,92020	2,90979	2,83141	2,78872	2,78724
Bridge	3,93756	3,80943	3,82776	3,79028	3,74301	3,71515	3,71500
Couple	3,06934	2,95778	2,92926	2,91357	2,86226	2,82931	2,82850
Crowd	2,53033	2,30199	2,35587	2,21533	2,17553	2,13854	2,13622
Elaine	3,24829	3,17030	3,02686	3,04358	2,72555	2,68284	2,68070
Finger	3,81198	3,76931	3,66495	3,66187	3,57261	3,56253	3,56199
Frog	4,49728	4,39138	4,35995	4,38501	4,33386	4,31746	4,31523
Goldhill	3,11340	3,01194	2,95847	2,94128	2,88422	2,85332	2,85169
Harbour	2,99500	2,88678	2,93097	2,76880	2,69130	2,64048	2,63902
Lax	4,10571	3,99614	4,01636	3,96960	3,91319	3,87529	3,87336
Lennagrey	2,63748	2,49144	2,38406	2,42346	2,37470	2,34582	2,34219
$Lena_{TMW}$	2,97852	2,84619	2,74747	2,78369	2,73288	2,70295	2,70189
Man	2,93018	2,75593	2,78036	2,70825	2,67795	2,62708	2,62707
Peppers	2,87369	2,76506	2,70206	2,69861	2,61937	2,57999	2,57751
Sailboat	3,16635	3,05508	3,00381	2,96765	2,87117	2,84571	2,84298
Seismic	1,73523	1,52189	1,31192	1,35498	1,12302	1,08665	1,08475
Shapes	0,99277	0,90550	1,28415	0,72180	0,59164	0,49335	0,49802
Tank	3,22137	3,11932	3,11823	3,09460	3,07286	3,05710	3,05560
Truck	3,01349	2,88473	2,91672	2,85522	2,82837	2,81234	2,81190
Woman1	3,04620	2,91141	2,82031	2,88013	2,79374	2,75984	2,75966
Woman2	2,02914	1,74276	1,57288	1,62659	1,59953	1,56235	1,55969
Balloon	1,79938	1,41912	1,32054	1,32299	1,27900	1,25420	1,25489
Barb	2,71472	2,67514	2,42849	2,47500	2,29389	2,22256	2,21935
Barb2	2,97951	2,94089	2,81312	2,78333	2,68533	2,63074	2,63169
Board	2,12772	2,00028	1,93796	1,88426	1,83491	1,80462	1,80309
Boats	2,41950	2,25058	2,16842	2,13943	2,07529	2,03714	2,03639
Girl	2,29701	2,17772	2,10735	2,07029	2,01264	1,97278	1,97085
Gold	2,93856	2,81812	2,75415	2,73981	2,68842	2,66209	2,65915
Hotel	2,74848	2,67980	2,65961	2,55864	2,50169	2,45905	2,45299
Zelda	2,27375	2,17970	2,04846	2,06968	2,02544	2,00571	2,00198
Bridge256	4,16907	4,02943	4,05750	4,02197	3,97212	3,92700	3,92874
Camera	2,85474	2,63928	2,71375	2,58789	2,51895	2,45837	2,45512
Couple256	2,34253	2,16171	2,13306	2,07959	2,04301	1,99210	1,99652
Earth	2,30554	2,23251	2,28149	2,26221	2,19373	2,15373	2,15424
Elif	1,95605	1,61021	1,50781	1,53418	1,41861	1,37479	1,37469
Noisesquare	3,92920	3,77599	3,83325	3,71924	3,65662	3,63022	3,62212
Omaha	4,66345	4,57608	4,72314	4,53760	4,46324	4,33235	4,32965
Sena	2,05103	1,77155	1,63892	1,69238	1,56096	1,52116	1,51682
Sensin	2,21143	1,98984	1,89478	1,93994	1,79160	1,75319	1,74734
Sinan	2,03845	1,80795	1,69666	1,74756	1,62741	1,58717	1,58026
Średnia	2,90472	2,76218	2,71975	2,68400	2,60068	2,55982	2,55798

Tab. 10.4. Porównanie metod $CoBALP_{max}$, GLICBAWLS i BMF z autorskimi Blend-13, Blend--20, Blend-24 i Blend-25 przy $d = 2$

Obrazy	$CoBALP_{max}$	Blend-13	GLICBAWLS	BMF	Blend-20	Blend-24	Blend-25
Aerial	2,68747	2,46407	2,61246	2,42297	2,34090	2,29051	2,28910
Airfield	3,24490	3,15264	3,17288	3,19421	3,07632	3,02260	3,02026
Airplane	2,01883	1,66010	1,62277	1,55554	1,54062	1,51914	1,51324
Baboon	3,64609	3,48126	3,37500	3,42261	3,35847	3,33164	3,32974
Barbara	2,37302	2,24135	1,94241	2,03723	1,82471	1,74978	1,74865
Boat	2,47070	2,33046	2,25656	2,24915	2,17512	2,13592	2,13505
Bridge	3,27719	3,11489	3,12537	3,09973	3,04910	3,02484	3,02709
Couple	2,47971	2,29458	2,26031	2,25391	2,19938	2,16780	2,16758
Crowd	2,11667	1,77355	1,79941	1,69531	1,65798	1,62399	1,62135
Elaine	2,60983	2,47703	2,33401	2,36011	2,06915	2,02652	2,02386
Finger	3,10583	3,06179	2,96213	2,96045	2,87874	2,86637	2,86118
Frog	3,79221	3,66194	3,63074	3,66699	3,60502	3,58989	3,58905
Goldhill	2,50488	2,33467	2,28564	2,26770	2,21748	2,18396	2,18317
Harbour	2,35712	2,27852	2,30341	2,16260	2,08923	2,05384	2,05344
Lax	3,41632	3,27551	3,29449	3,24951	3,19175	3,15323	3,15178
Lennagrey	2,16663	1,85465	1,74966	1,78430	1,73843	1,71033	1,70692
$Lena_{TMW}$	2,36911	2,17650	2,07657	2,11621	2,06619	2,03779	2,03792
Man	2,39590	2,12388	2,13562	2,07336	2,04807	2,00364	1,99966
Peppers	2,27277	2,09628	2,03613	2,03760	1,96399	1,92180	1,91980
Sailboat	2,54425	2,38825	2,33279	2,30273	2,21363	2,18574	2,18207
Seismic	1,61310	1,11417	0,96552	0,97668	0,76809	0,72933	0,72955
Shapes	0,91925	0,70946	0,93677	0,57349	0,46334	0,39044	0,39090
Tank	2,60468	2,43078	2,42523	2,40430	2,38475	2,36772	2,36749
Truck	2,44308	2,22527	2,24478	2,18909	2,17051	2,15271	2,15215
Woman1	2,43677	2,24482	2,15262	2,21643	2,12906	2,09467	2,09462
Woman2	1,67407	1,22188	1,05798	1,09888	1,08282	1,04689	1,04696
Balloon	1,54796	0,90531	0,80309	0,79059	0,75906	0,73639	0,73638
Barb	2,23254	2,06399	1,82841	1,87400	1,70572	1,63732	1,63430
Barb2	2,44311	2,30051	2,17375	2,14136	2,05650	2,00170	1,99966
Board	1,74205	1,41703	1,36360	1,29190	1,25937	1,22522	1,22236
Boats	1,95243	1,64638	1,56757	1,52832	1,47482	1,44011	1,43832
Girl	1,86136	1,60000	1,53646	1,49591	1,45054	1,41026	1,41126
Gold	2,34329	2,16550	2,09904	2,09028	2,04539	2,01984	2,01755
Hotel	2,24921	2,04147	2,02440	1,92855	1,88108	1,84388	1,83942
Zelda	1,87988	1,57682	1,45866	1,48380	1,44117	1,42221	1,41943
Bridge256	3,48364	3,32474	3,34448	3,32471	3,27019	3,22949	3,22731
Camera	2,41809	2,02367	2,08325	1,96826	1,90450	1,84480	1,84946
Couple256	1,87976	1,60574	1,57275	1,57129	1,50999	1,45894	1,45821
Earth	1,94336	1,83894	1,85266	1,87402	1,79811	1,76332	1,76106
Elif	1,68481	1,13298	1,02588	1,03564	0,95992	0,91943	0,92162
Noisesquare	3,22144	3,05687	3,10510	3,00049	2,94208	2,91222	2,90332
Omaha	3,95984	3,87306	3,99438	3,83643	3,76164	3,65523	3,65199
Sena	1,66199	1,28874	1,15405	1,18896	1,09862	1,05759	1,05867
Sensin	1,86255	1,49013	1,40515	1,43213	1,32591	1,28857	1,29120
Sinan	1,75110	1,34189	1,22839	1,25781	1,17184	1,13560	1,13274
Średnia	2,39242	2,15293	2,10472	2,07746	2,00265	1,96407	1,96260

Tab. 10.5. Porównanie metod $CoBALP_{max}$, GLICBAWLS i BMF z autorskimi Blend-13, Blend--20, Blend-24 i Blend-25 przy $d = 3$

Obrazy	$CoBALP_{max}$	Blend-13	GLICBAWLS	BMF	Blend-20	Blend-24	Blend-25
Aerial	2,39658	2,08287	2,20148	2,04004	1,96955	1,91899	1,91418
Airfield	2,84579	2,69722	2,71127	2,73669	2,62010	2,56756	2,56348
Airplane	1,72733	1,31060	1,27423	1,21265	1,19826	1,17769	1,17237
Baboon	3,22183	3,02242	2,91721	2,96265	2,90751	2,87764	2,87595
Barbara	2,08551	1,85480	1,56186	1,65234	1,46784	1,38861	1,38586
Boat	2,16940	1,92197	1,85703	1,84558	1,78070	1,74252	1,74027
Bridge	2,86539	2,67762	2,68021	2,65735	2,61374	2,58962	2,58987
Couple	2,18930	1,89582	1,85721	1,84851	1,79514	1,76167	1,75891
Crowd	1,84158	1,46627	1,47778	1,39209	1,36192	1,32819	1,32263
Elaine	2,21408	2,04239	1,90579	1,93347	1,66950	1,62673	1,62432
Finger	2,73953	2,61245	2,52371	2,52063	2,44152	2,43338	2,42983
Frog	3,31650	3,18753	3,15405	3,19507	3,12971	3,11528	3,11379
Goldhill	2,16223	1,92111	1,87396	1,86145	1,81038	1,77743	1,77720
Harbour	2,04550	1,92452	1,93625	1,82153	1,74788	1,70625	1,70538
Lax	2,98352	2,81415	2,83014	2,78809	2,73394	2,69257	2,69196
Lennagrey	1,93915	1,46825	1,35739	1,38904	1,34874	1,31409	1,31242
$Lena_{TMW}$	2,11417	1,76588	1,66742	1,70654	1,65874	1,63102	1,62671
Man	2,11761	1,74843	1,74240	1,69556	1,67520	1,62455	1,62563
Peppers	2,03348	1,68889	1,62894	1,62476	1,56023	1,52007	1,51351
Sailboat	2,22885	1,97567	1,92029	1,89038	1,81023	1,78164	1,77757
Seismic	1,46069	0,87879	0,75909	0,75708	0,57758	0,53165	0,53273
Shapes	0,88892	0,60048	0,75235	0,48840	0,39367	0,33553	0,33763
Tank	2,23596	2,00565	1,99442	1,97656	1,96045	1,94447	1,94366
Truck	2,12500	1,82035	1,83185	1,78271	1,76510	1,75490	1,75196
Woman1	2,17435	1,84056	1,74756	1,81409	1,72667	1,69082	1,69089
Woman2	1,48911	0,92566	0,78452	0,81787	0,80145	0,77025	0,76683
Balloon	1,32027	0,63021	0,53958	0,50833	0,48433	0,47112	0,46906
Barb	1,96038	1,69473	1,46696	1,50718	1,34908	1,28068	1,27747
Barb2	2,12272	1,90297	1,76725	1,73465	1,65646	1,59901	1,59874
Board	1,41651	1,04995	0,99904	0,91235	0,88771	0,85618	0,85305
Boats	1,62490	1,28160	1,20793	1,16235	1,11747	1,08653	1,08286
Girl	1,66096	1,26049	1,19942	1,14969	1,11802	1,07823	1,07550
Gold	2,02706	1,76878	1,70378	1,69591	1,65739	1,63130	1,62757
Hotel	1,96447	1,65551	1,63551	1,53503	1,49484	1,45903	1,45301
Zelda	1,76061	1,21585	1,09512	1,12361	1,07892	1,05371	1,05355
Bridge256	3,05310	2,88083	2,88757	2,87598	2,82346	2,78325	2,78674
Camera	2,00891	1,64407	1,67651	1,57373	1,51649	1,46042	1,46100
Couple256	1,69568	1,30562	1,25281	1,26025	1,20007	1,15216	1,15234
Earth	1,72498	1,58844	1,58801	1,62305	1,55247	1,51488	1,51212
Elif	1,43884	0,86581	0,76318	0,76074	0,71437	0,67476	0,67192
Noisesquare	2,81323	2,59465	2,63281	2,53662	2,47731	2,44279	2,43474
Omaha	3,51831	3,42111	3,51868	3,38721	3,31197	3,21376	3,21445
Sena	1,40613	0,99258	0,87134	0,89453	0,82793	0,79610	0,78281
Sensin	1,69885	1,20445	1,10718	1,13965	1,04890	1,01622	1,00908
Sinan	1,59460	1,06329	0,95081	0,98193	0,90245	0,87619	0,86996
Średnia	2,09826	1,78158	1,72915	1,70609	1,63879	1,60110	1,59848

Tab. 10.6. Porównanie metod $CoBALP_{max}$, GLICBAWLS i BMF z autorskimi Blend-13, Blend--20, Blend-24 i Blend-25 przy $d = 5$

Obrazy	$CoBALP_{max}$	Blend-13	GLICBAWLS	BMF	Blend-20	Blend-24	Blend-25
Aerial	1,95938	1,62332	1,69556	1,57898	1,51808	1,46994	1,46704
Airfield	2,34836	2,11120	2,12195	2,15356	2,04198	1,99202	1,98810
Airplane	1,32632	0,91208	0,88922	0,82568	0,81674	0,80159	0,79819
Baboon	2,70761	2,43461	2,32886	2,37732	2,32648	2,29661	2,29448
Barbara	1,72949	1,38784	1,11368	1,19629	1,03902	0,96143	0,96164
Boat	1,86911	1,41295	1,36908	1,34973	1,29399	1,25702	1,25396
Bridge	2,38031	2,12640	2,11542	2,10364	2,06416	2,04288	2,04140
Couple	1,87616	1,40162	1,34851	1,34668	1,29159	1,25385	1,25590
Crowd	1,56509	1,09514	1,10712	1,04858	1,02335	0,99387	0,99267
Elaine	1,89114	1,50069	1,37796	1,41479	1,18237	1,13412	1,13047
Finger	2,15958	2,05915	1,98761	1,98145	1,91697	1,90637	1,90548
Frog	2,76059	2,56370	2,52847	2,57800	2,51046	2,49295	2,49185
Goldhill	1,81046	1,41720	1,37402	1,35962	1,31786	1,28708	1,28582
Harbour	1,66470	1,49130	1,49875	1,41443	1,34572	1,31768	1,31799
Lax	2,49307	2,22512	2,23193	2,18909	2,14293	2,10590	2,10423
Lennagrey	1,64804	0,99264	0,88632	0,90918	0,87481	0,84196	0,84079
$Lena_{TMW}$	1,82385	1,24711	1,14636	1,18201	1,13627	1,10472	1,10017
Man	1,79562	1,29333	1,26880	1,23730	1,22603	1,18136	1,17839
Peppers	1,75317	1,16701	1,10928	1,10425	1,04799	1,00452	1,00015
Sailboat	1,85153	1,45260	1,41391	1,38464	1,32521	1,30078	1,29306
Seismic	1,09888	0,60155	0,55103	0,51721	0,38039	0,33921	0,33756
Shapes	0,79092	0,47927	0,54865	0,39197	0,31303	0,27110	0,26902
Tank	1,93823	1,47783	1,46332	1,44788	1,43514	1,41815	1,41625
Truck	1,77319	1,32570	1,32547	1,28516	1,27557	1,26062	1,25434
Woman1	1,84030	1,32990	1,22983	1,29260	1,21147	1,17233	1,16938
Woman2	1,25931	0,60078	0,49200	0,51489	0,49291	0,47204	0,47377
Balloon	1,01636	0,38722	0,32917	0,28642	0,27667	0,26413	0,26355
Barb	1,60617	1,25439	1,04151	1,07840	0,93503	0,88339	0,87947
Barb2	1,68216	1,41768	1,28339	1,25833	1,19291	1,14004	1,13907
Board	1,00095	0,65449	0,60141	0,52461	0,50215	0,47941	0,47992
Boats	1,24169	0,90668	0,83997	0,79769	0,76561	0,74077	0,73837
Girl	1,46098	0,86840	0,80806	0,77114	0,74431	0,70825	0,70556
Gold	1,70519	1,28258	1,22637	1,22106	1,18720	1,16028	1,15929
Hotel	1,50372	1,15379	1,13731	1,03549	1,00385	0,96888	0,96057
Zelda	1,57467	0,74792	0,64861	0,65833	0,62046	0,59785	0,59498
Bridge256	2,52722	2,30721	2,29871	2,29883	2,25325	2,22137	2,22113
Camera	1,53369	1,22578	1,21960	1,13867	1,09868	1,06285	1,05563
Couple256	1,33911	0,95071	0,90271	0,93213	0,86607	0,81058	0,81415
Earth	1,46167	1,26324	1,23645	1,28564	1,22264	1,18959	1,18887
Elif	1,10974	0,59695	0,52576	0,50977	0,47737	0,44826	0,44551
Noisesquare	2,15112	1,99857	2,00415	1,92725	1,86960	1,81816	1,81067
Omaha	2,94275	2,82930	2,89172	2,80371	2,73798	2,65097	2,65523
Sena	1,10815	0,67389	0,57336	0,59326	0,53972	0,50954	0,51132
Sensin	1,50562	0,86346	0,78809	0,80469	0,73616	0,70438	0,70107
Sinan	1,38879	0,73715	0,65491	0,66602	0,61826	0,58305	0,57973
Średnia	1,73276	1,32999	1,27854	1,26170	1,20441	1,16937	1,16725

Tab. 10.7. Porównanie metod $CoBALP_{max}$, GLICBAWLS i BMF z autorskimi Blend-13, Blend--20, Blend-24 i Blend-25 przy $d = 7$

Obrazy	$CoBALP_{max}$	Blend-13	GLICBAWLS	BMF	Blend-20	Blend-24	Blend-25
Aerial	1,69406	1,34378	1,38651	1,30005	1,24701	1,19857	1,19822
Airfield	2,10809	1,73568	1,74722	1,77539	1,66750	1,62237	1,61749
Airplane	1,10681	0,70199	0,68320	0,62207	0,62059	0,60567	0,60318
Baboon	2,41144	2,05276	1,94449	1,99121	1,94852	1,91574	1,91246
Barbara	1,50079	1,11769	0,86539	0,94543	0,80317	0,73251	0,73228
Boat	1,64172	1,09155	1,05899	1,03088	0,98792	0,94739	0,94475
Bridge	2,10419	1,77657	1,75452	1,75720	1,71778	1,69841	1,69661
Couple	1,64441	1,09517	1,03613	1,04028	0,98845	0,95483	0,95458
Crowd	1,37271	0,88088	0,89066	0,84717	0,82682	0,80118	0,79464
Elaine	1,73572	1,15545	1,04700	1,07642	0,87299	0,82996	0,82549
Finger	1,90448	1,72345	1,66464	1,65710	1,59724	1,59040	1,58601
Frog	2,34726	2,15205	2,11536	2,16333	2,10228	2,08337	2,08221
Goldhill	1,59821	1,09987	1,06567	1,05457	1,01284	0,99041	0,98853
Harbour	1,43347	1,23481	1,22293	1,16077	1,09979	1,07164	1,07370
Lax	2,14987	1,83273	1,83401	1,78809	1,74761	1,70911	1,70639
Lennagrey	1,44946	0,72745	0,63641	0,65698	0,61356	0,59338	0,59065
$Lena_{TMW}$	1,56583	0,92058	0,82938	0,85315	0,80888	0,78247	0,77774
Man	1,57498	1,01827	0,99435	0,96936	0,96274	0,91929	0,91994
Peppers	1,47678	0,84110	0,79181	0,78406	0,73956	0,70290	0,69648
Sailboat	1,55667	1,12948	1,09985	1,06665	1,02975	1,01095	1,00544
Seismic	0,85342	0,45325	0,44812	0,38196	0,29208	0,25114	0,24976
Shapes	0,69907	0,40182	0,43817	0,32532	0,26525	0,23905	0,23973
Tank	1,67712	1,13572	1,12146	1,10266	1,09469	1,07067	1,07219
Truck	1,53503	1,02062	1,02420	0,98633	0,97658	0,95888	0,95720
Woman1	1,59637	1,00660	0,90976	0,96399	0,89136	0,84372	0,84394
Woman2	1,10239	0,42093	0,34738	0,35510	0,33297	0,32111	0,31936
Balloon	0,86094	0,28104	0,24772	0,20123	0,19631	0,18906	0,18597
Barb	1,39823	1,01435	0,81325	0,85579	0,71930	0,66451	0,66309
Barb2	1,45658	1,15373	1,01618	1,00872	0,95046	0,90285	0,90039
Board	0,75496	0,47347	0,42282	0,35648	0,34790	0,32584	0,32406
Boats	1,05625	0,72094	0,65401	0,62415	0,59920	0,57434	0,57452
Girl	1,31887	0,65071	0,60417	0,57253	0,55339	0,51674	0,51420
Gold	1,46676	0,98891	0,93881	0,93194	0,90077	0,87960	0,87693
Hotel	1,21163	0,87263	0,85035	0,76898	0,74577	0,71666	0,71036
Zelda	1,31520	0,50214	0,44606	0,43279	0,39947	0,38341	0,38139
Bridge256	2,21460	1,94209	1,92358	1,93652	1,88792	1,86122	1,85991
Camera	1,29272	0,99916	0,97388	0,93066	0,89977	0,85721	0,85143
Couple256	1,16406	0,74380	0,70618	0,73926	0,67517	0,62927	0,62596
Earth	1,28918	1,05688	1,02527	1,08203	1,01686	0,98830	0,98613
Elif	0,94275	0,46732	0,40723	0,39160	0,36919	0,34024	0,32681
Noisesquare	1,99268	1,62746	1,65576	1,58447	1,57310	1,55028	1,54990
Omaha	2,56909	2,44740	2,47668	2,41650	2,36052	2,28847	2,28889
Sena	0,92163	0,52423	0,42529	0,43750	0,40785	0,38686	0,38457
Sensin	1,35364	0,68402	0,62073	0,63086	0,57208	0,53439	0,53381
Sinan	1,25452	0,56384	0,50940	0,50098	0,46561	0,43552	0,43376
Średnia	1,50388	1,06188	1,01500	1,00130	0,95308	0,92155	0,91913

Tab. 10.8. Porównanie metod $CoBALP_{max}$, GLICBAWLS i BMF z autorskimi Blend-13, Blend--20, Blend-24 i Blend-25 przy $d = 10$

Obrazy	$CoBALP_{max}$	Blend-13	GLICBAWLS	BMF	Blend-20	Blend-24	Blend-25
Aerial	1,43384	1,07604	1,08914	1,03320	0,99440	0,94705	0,94392
Airfield	1,79233	1,34031	1,35251	1,36890	1,27148	1,22378	1,22020
Airplane	0,85489	0,51738	0,50461	0,46228	0,46453	0,44697	0,45005
Baboon	2,07510	1,65618	1,55502	1,59485	1,56229	1,52746	1,52626
Barbara	1,30429	0,87835	0,65686	0,73022	0,59999	0,53618	0,53564
Boat	1,29700	0,79183	0,76807	0,74255	0,70350	0,66588	0,66440
Bridge	1,79770	1,42540	1,39725	1,40222	1,37327	1,34849	1,34801
Couple	1,32712	0,82089	0,76208	0,76501	0,71973	0,68277	0,68402
Crowd	1,19003	0,67750	0,69339	0,65820	0,63808	0,61653	0,61378
Elaine	1,44934	0,79472	0,71301	0,72559	0,57636	0,53813	0,53540
Finger	1,73969	1,40419	1,35709	1,34583	1,29255	1,28836	1,28486
Frog	2,02353	1,72316	1,68912	1,72058	1,67302	1,65551	1,65267
Goldhill	1,29840	0,79817	0,78021	0,76123	0,74117	0,70612	0,70102
Harbour	1,18320	0,97704	0,96057	0,91272	0,86224	0,83170	0,82758
Lax	1,76978	1,41145	1,41617	1,37683	1,33539	1,29673	1,29393
Lennagrey	1,22937	0,51554	0,44370	0,45349	0,43024	0,40895	0,40724
$Lena_{TMW}$	1,29291	0,63151	0,56000	0,57971	0,53993	0,51974	0,51671
Man	1,35474	0,76656	0,75064	0,72742	0,72263	0,68892	0,68544
Peppers	1,20081	0,55812	0,52127	0,51208	0,47710	0,45057	0,44413
Sailboat	1,22437	0,81431	0,79883	0,76892	0,75041	0,72567	0,72224
Seismic	0,56909	0,33029	0,34790	0,26733	0,20844	0,18412	0,18101
Shapes	0,60916	0,32121	0,34787	0,26331	0,21834	0,19893	0,19849
Tank	1,30896	0,78332	0,78256	0,75415	0,74511	0,72694	0,72464
Truck	1,24103	0,73389	0,74231	0,70349	0,69584	0,67255	0,67089
Woman1	1,32413	0,72330	0,63962	0,68958	0,61685	0,58200	0,58282
Woman2	0,92328	0,28107	0,24472	0,23450	0,21185	0,20296	0,20342
Balloon	0,72828	0,19809	0,18152	0,14483	0,14068	0,13833	0,13811
Barb	1,22811	0,80282	0,62274	0,66358	0,53954	0,49711	0,49404
Barb2	1,28416	0,91313	0,78609	0,79113	0,74535	0,69773	0,69816
Board	0,60222	0,34380	0,29603	0,25856	0,24232	0,22621	0,22237
Boats	0,88002	0,54583	0,49192	0,47523	0,45158	0,42929	0,42650
Girl	1,04713	0,46463	0,44676	0,40910	0,39175	0,36694	0,36623
Gold	1,18717	0,70960	0,67888	0,66435	0,64376	0,61441	0,61455
Hotel	0,98825	0,65730	0,62238	0,57106	0,55245	0,52144	0,52000
Zelda	1,06842	0,33651	0,30230	0,28310	0,25737	0,25070	0,24875
Bridge256	1,89294	1,57043	1,54492	1,55615	1,51836	1,49068	1,49269
Camera	1,06763	0,79543	0,76892	0,75342	0,71588	0,66606	0,66835
Couple256	0,96716	0,55919	0,53406	0,55518	0,49425	0,46465	0,46439
Earth	1,11572	0,85580	0,81689	0,86670	0,82283	0,78636	0,78445
Elif	0,78101	0,34303	0,31226	0,29297	0,27530	0,24933	0,24765
Noisesquare	1,65332	1,32123	1,32996	1,29932	1,30162	1,27722	1,27585
Omaha	2,18835	2,04185	2,03528	2,02051	1,95625	1,91216	1,90703
Sena	0,77124	0,38531	0,31116	0,32080	0,30209	0,28503	0,27483
Sensin	1,25806	0,51871	0,48462	0,47363	0,42960	0,40152	0,40257
Sinan	1,13074	0,41934	0,39014	0,36865	0,34268	0,32156	0,32243
Średnia	1,25898	0,81186	0,77403	0,76272	0,72330	0,69488	0,69306

W przypadku kodowania ze stratą jakości często stosowany wskaźnik jakości PSNR jest uznawany za nie w pełni pozwalający oceniać jakość obrazu z punktu widzenia ludzkiej percepcji wzrokowej. Opracowany w pracy [134] wskaźnik jakości MSSIM (ang. *Mean Structural SIMilarity Index*) wykorzystuje do porównywania obrazów (źródłowego i zdekodowanego) cechy zarówno strukturalne, jak i te związane z luminancją i kontrastem. Wskaźnik ten jest wartością z przedziału od −1 do 1, przy czym wyższa wartość oznacza większe podobieństwo obrazów. W omówionych tu badaniach wykorzystano oba wskaźniki jakości.

Na rysunku 10.1 przedstawiono porównanie wartości PSNR, a na rys. 10.2 porównanie wartości MSSIM otrzymanych metodami Blend-13 oraz JPEG2000 na przykładzie pomiarów kodowania obrazu Lennagrey. Dla większości obrazów testowych wartość PSNR oraz MSSIM uzyskana metodą Blend-13 przy dopuszczalnym błędzie $d \leq 3$ jest wyższa niż z użyciem JPEG2000 dla plików o takiej samej średniej bitowej. Jednocześnie przy tych samych średnich bitowych maksymalny błąd d jest w przypadku JPEG2000 znacznie wyższy, co pokazuje tab. 10.9. Biorąc pod uwagę średnią bitową, lepszą wartość PSNR oraz MSSIM uzyskuje się metodą Blend-13 przy średnich bitowych wyższych od 1,5 bitu na piksel.

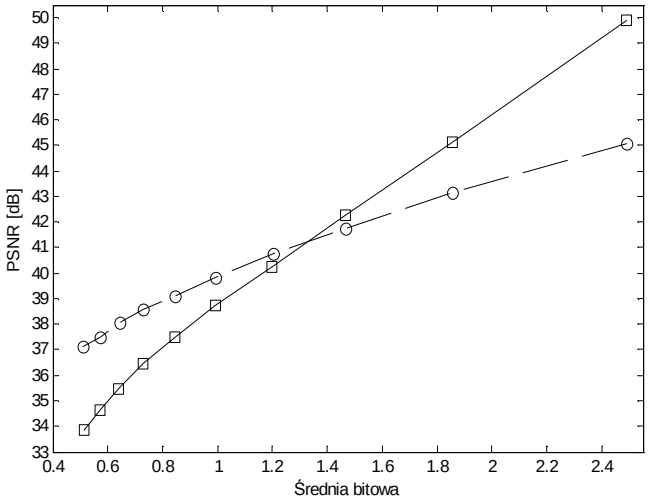
Tab. 10.9. Porównanie Blend-13 w trybie prawie bezstratnym z JPEG2000 na przykładzie obrazu Lennagrey

Blend-13				JPEG2000			
d	średnia	PSNR	MSSIM	średnia	PSNR	MSSIM	*d*
1	2,49144	49,89429	0,99473	2,49145	45,09458	0,98501	7
2	1,85465	45,15079	0,98462	1,85480	43,15705	0,97573	8
3	1,46825	42,26534	0,97124	1,46783	41,74485	0,96705	11
4	1,19771	40,26459	0,95682	1,20132	40,74678	0,95868	12
5	0,99264	38,75684	0,94257	0,99277	39,81996	0,95150	17
6	0,84358	37,51494	0,92906	0,84363	39,10160	0,94507	17
7	0,72745	36,45099	0,91631	0,72781	38,60727	0,93805	24
8	0,64068	35,47136	0,90388	0,64258	38,06484	0,93290	27
9	0,57260	34,61310	0,89212	0,57275	37,49548	0,92723	27
10	0,51554	33,86520	0,88133	0,51157	37,12479	0,92410	29

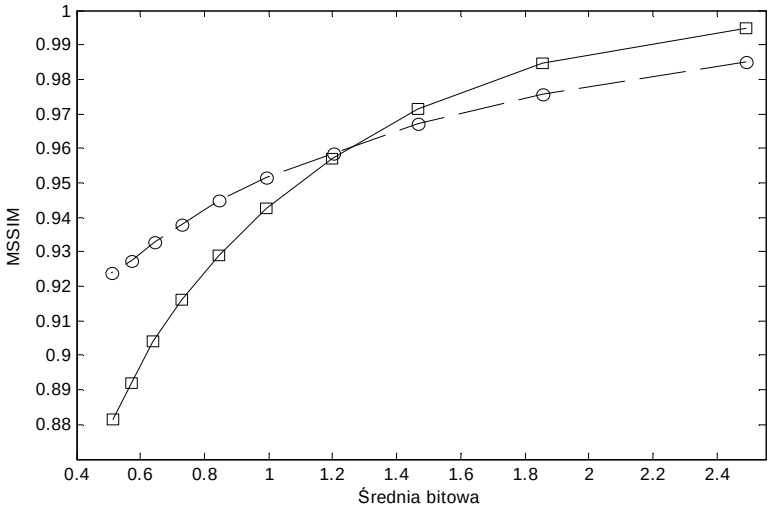
Podobną analizę dla obrazu Lennagrey zaprezentowano z użyciem metody Blend-24 (patrz tab. 10.10). Na rysunku 10.3 przedstawiono porównanie wartości PSNR, a na rys. 10.4 porównanie wartości MSSIM uzyskanych metodami Blend-24 oraz JPEG2000 na przykładzie obrazu Lennagrey. W większości przypadków obrazów testowych wartość PSNR oraz MSSIM uzyskana metodą Blend-24 przy maksymalnym błędzie $d \leq 4$ jest wyższa niż z użyciem JPEG2000 dla plików o takiej samej średniej bitowej (dla niektórych obrazów także przy $d = 5$ wskaźniki obu metod są porównywalne). Biorąc pod uwagę średnią bitową, lepszą wartość PSNR oraz MSSIM uzyskuje się metodą Blend-24 przy średnich bitowych wyższych od 1 bitu na piksel.

Tab. 10.10. Porównanie Blend-24 w trybie prawie bezstratnym z JPEG2000 na przykładzie obrazu Lennagrey

Blend-24				JPEG2000			
d	średnia	PSNR	MSSIM	średnia	PSNR	MSSIM	*d*
1	2,34583	49,89332	0,99474	2,34370	44,69073	0,98403	7
2	1,71033	45,18175	0,98479	1,71344	42,63859	0,97311	9
3	1,31409	42,36445	0,97208	1,31378	41,20315	0,96309	12
4	1,03839	40,44408	0,95887	1,03824	40,04546	0,95334	13
5	0,84198	38,99028	0,94601	0,84195	39,08650	0,94522	17
6	0,69720	37,83855	0,93385	0,69861	38,44492	0,93605	25
7	0,59338	36,83331	0,92197	0,59512	37,68537	0,92961	27
8	0,51578	35,93403	0,91071	0,51575	37,12720	0,92413	29
9	0,45319	35,10413	0,89917	0,45392	36,59587	0,91839	43
10	0,40897	34,38855	0,88937	0,40897	36,04312	0,91300	43

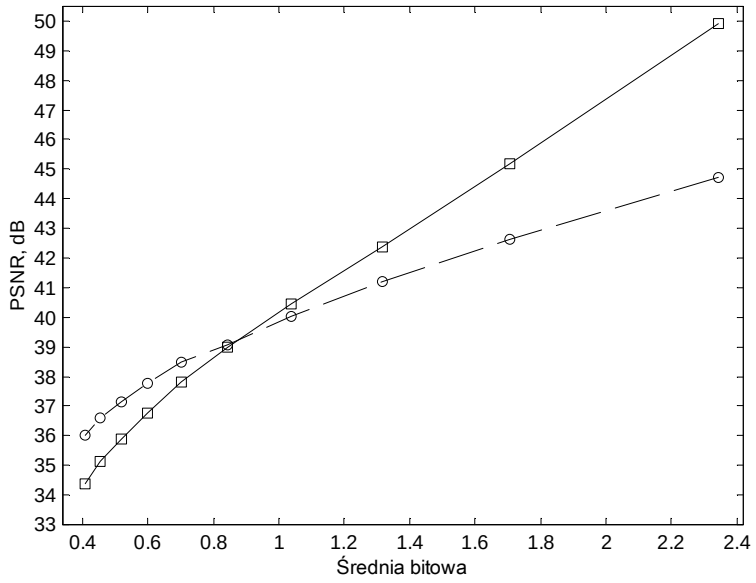


Rys. 10.1. Porównanie wartości PSNR obrazu Lennagrey w zależności od średniej bitowej dla metody Blend-13 w trybie prawie bezstratnym dla $d \leq 10$ (linia ciągła) oraz JPEG2000 (linia przerywana)

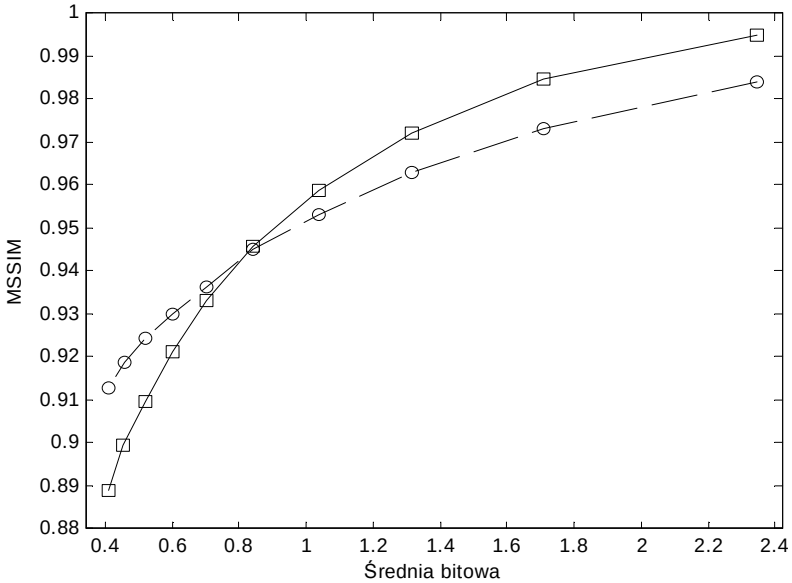


Rys. 10.2. Porównanie wartości MSSIM obrazu Lennagrey w zależności od średniej bitowej dla metody Blend-13 w trybie prawie bezstratnym dla $d \leq 10$ (linia ciągła) oraz JPEG2000 (linia przerywana)

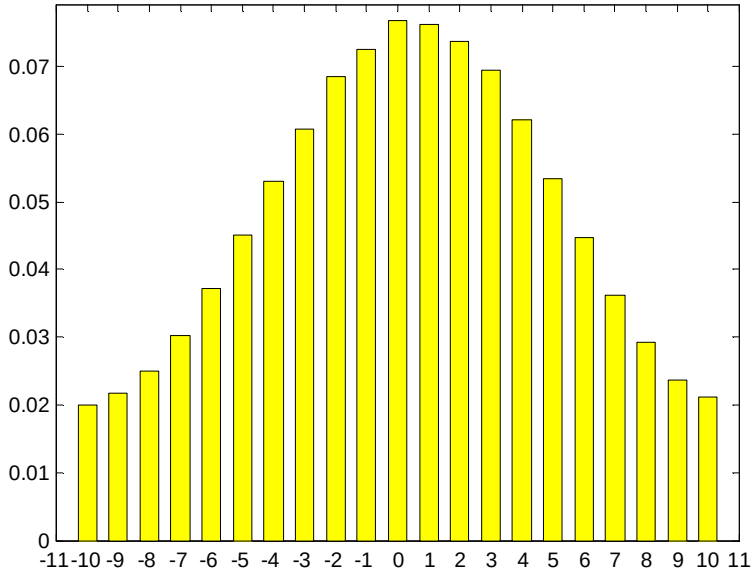
Na rysunku 10.5 przedstawiono histogram różnic między obrazem źródłowym Lennagrey oraz obrazem po kodowaniu metodą Blend-24 (średnia bitowa 0,408) i zdekodowaniu z dopuszczalnym błędem d = 10. Porównując to z przedstawionym na rys. 10.6 histogramem różnic między obrazem źródłowym Lennagrey oraz obrazem powstałym po kodowaniu (średnia bitowa 0,408) i zdekodowaniu metodą JPEG2000, można zauważyć, iż rozkład błędów w metodzie Blend-24 jest bardziej spłaszczony i znacznie odbiega od rozkładu Laplace'a, charakterystycznego dla JPEG2000. Uzasadnia to w pewnym stopniu przewagę JPEG2000 nad metodami wykorzystującymi tryb prawie bezstratny przy użyciu większych dopuszczalnych błędów d.



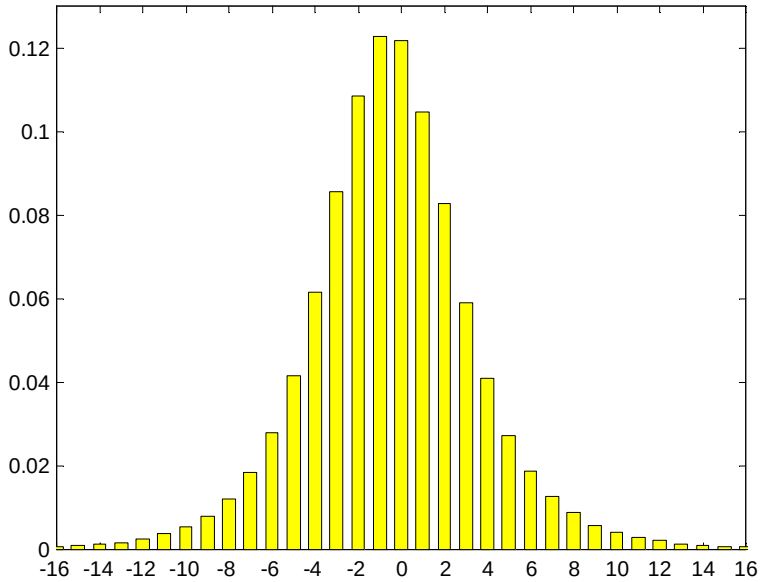
Rys. 10.3. Porównanie wartości PSNR obrazu Lennagrey w zależności od średniej bitowej dla metody Blend-24 w trybie prawie bezstratnym dla $d \leq 10$ (linia ciągła) oraz JPEG2000 (linia przerywana)



Rys. 10.4. Porównanie wartości MSSIM obrazu Lennagrey w zależności od średniej bitowej dla metody Blend-24 w trybie prawie bezstratnym dla $d \leq 10$ (linia ciągła) oraz JPEG2000 (linia przerywana)



Rys. 10.5. Histogram różnic między obrazem źródłowym Lennagrey oraz obrazem po kodowaniu metodą Blend-24 i zdekodowaniu z dopuszczalnym błędem d = 10 (przy średniej bitowej 0,408)



Rys. 10.6. Histogram różnic między obrazem źródłowym Lennagrey oraz obrazem po kodowaniu i zdekodowaniu metodą JPEG2000 (przy średniej bitowej 0,408)

10.3. Bezstratna kompresja obrazów astronomicznych i satelitarnych

Zdjęcia Ziemi, Marsa i innych obiektów kosmicznych pozyskiwane z satelitów i pojazdów kosmicznych powinny być przesyłane i przechowywane w postaci bezstratnej. Szczególnego znaczenia nabiera tu potrzeba uzyskania wysokiej efektywności kompresji przy zachowaniu odpowiednio niskiego poziomu złożoności algorytmu i jego sprzętowej imple-

mentacji. Należy bowiem zwrócić uwagę na wysoki koszt samego przesyłania wynikający z ograniczeń transmisyjnych, zwłaszcza w sytuacji, gdy mamy do czynienia z transmisją na duże odległości, np. z wypraw na Marsa.

Mając na względzie rozwój technologii prowadzących do miniaturyzacji i zmniejszania wymagań energetycznych, do przeprowadzania procesu kodowania w pracy [120] zaproponowano algorytm o nieco wyższej złożoności implementacyjnej od rozwiązań proponowanych w literaturze [21, 68], lecz nadający się do realizacji sprzętowej. Propozycja pozwala na bezstratne kodowanie około trzech klatek na sekundę (o rozdzielczości 512 × 512 pikseli) w przypadku kompresji obrazów statycznych (metoda Blend-13) oraz w rozbudowanej wersji do 30 klatek na sekundę dla sekwencji wideo (metoda Blend-V opisana w podrozdziale 10.4). Sekwencje wówczas są dzielone na niezależne grupy po 10 klatek, kodowanych w trybie potokowym uwzględniającym zależności czasowe między sąsiednimi klatkami.

Testy systemu przeprowadzono dla obrazów w ośmiobitowej głębi odcieni szarości, wykorzystując referencyjny zbiór 14 obrazów testowych CCSDS (ang. *The Consultative Committee for Space Data Systems*). Elastyczność proponowanej metody pozwala kodować obrazy nawet o 16-bitowej głębi, co jest możliwe dzięki rozszerzeniu opisanego w podrozdziale 8.3 kwantyzatora błędów predykcji o zaledwie kilka nowych symboli, a także przez dodanie kilku wektorów przechowujących rozkłady reszt kwantyzacji. Wyniki testów metody Blend-13 zaprezentowano w tab. 10.11 wraz z porównaniem z innymi rozwiązaniami o niskiej złożoności implementacyjnej, które zaczerpnięto z pracy [69]. Znalazły się wśród nich metody falkowe – CCSDS [69], JPEG2000 [72], SPIHT [95], ICER [57], jak i predykcyjne – PRDC [68], JPEG-LS [136], FPGA (w pełni sprzętowa realizacja zaprojektowana dla rekonfigurowalnych układów cyfrowych) [21].

Tab. 10.11. Porównanie metod bezstratnej kompresji zastosowanych do obrazów testowych CCSDS

Obrazy	CCSDS	PRDC	JPEG-LS	JPEG 2000	SPIHT	ICER	FPGA	Blend-13
coastal_b1	3,36	3,56	3,09	3,13	3,09	3,07	3,00	2,934
coastal_b2	3,22	3,32	2,90	2,97	2,94	2,92	2,84	2,781
coastal_b3	3,48	3,68	3,22	3,23	3,21	3,20	3,14	3,052
coastal_b4	2,81	2,91	2,41	2,53	2,57	2,55	2,37	2,372
coastal_b5	3,16	3,30	2,81	2,94	2,91	2,89	2,79	2,709
coastal_b6h	3,02	2,75	2,50	2,60	2,71	2,54	2,52	2,432
coastal_b6l	2,35	2,03	1,76	1,96	2,02	1,87	1,84	1,842
coastal_b7	3,45	3,66	3,17	3,22	3,17	3,15	3,10	3,009
coastal_b8	3,66	3,93	3,42	3,40	3,35	3,31	3,28	3,241
europa3	6,61	7,48	6,64	6,52	6,46	6,30	6,42	6,100
marstest	4,78	5,39	4,69	4,74	4,64	4,63	4,60	4,345
lunar	4,58	5,23	4,35	4,49	4,43	4,40	4,20	4,114
spot-la_b3	4,80	5,20	4,53	4,69	4,70	4,56	4,43	4,365
spot-la_panchr	4,27	4,87	4,00	4,13	4,11	4,03	3,90	3,829
Średnia	3,82	4,09	3,54	3,61	3,59	3,53	3,46	3,366

W pracy [64] zaprezentowano metodę stratnej kompresji obrazów astrofizycznych, która wprowadza jedynie szum na poziomie zbliżonym do szumu zawartego w pozyskanym

obrazie źródłowym. Metoda takiego kodowania, nazwana kompresją wirtualnie bezstratną (ang. *virtually lossless compression*), polega na użyciu kwantyzacji, której poziom szumu jest nie większy, niż wynika to z zastosowanych urządzeń pozyskiwania obrazów. Algorytm decyduje o maksymalnym błędzie *d* indywidualnie dla każdego obrazu.

Kolejne badania przeprowadzono dla trybu prawie bezstratnego, przedstawiając w tab. 10.12 średnie bitowe dla obrazów testowych z maksymalną wartością błędu $d = \{1, 2, 3, 4, 5, 6, 7\}$. Wnioski dotyczące trybu prawie bezstratnego metody Blend-13 są podobne do tych przedstawionych w podrozdziale 10.2.

Tab. 10.12. Średnia bitowa dla metody Blend-13 zastosowanej w prawie bezstratnej kompresji obrazów testowych CCSDS

Obrazy	Wymiary	$d = 1$	$d = 2$	$d = 3$	$d = 4$	$d = 5$	$d = 6$	$d = 7$
coastal_b1	1024×1024	1,54649	1,00679	0,73927	0,57920	0,47143	0,39340	0,33086
coastal_b2	1024×1024	1,44960	0,97971	0,73545	0,57889	0,47777	0,39892	0,34018
coastal_b3	1024×1024	1,65843	1,13236	0,85610	0,68240	0,56003	0,47115	0,40992
coastal_b4	1024×1024	1,16778	0,79846	0,58793	0,46904	0,39859	0,33440	0,28964
coastal_b5	1024×1024	1,44091	0,96228	0,74179	0,59852	0,50086	0,42289	0,36571
coastal_b6h	512×512	1,20087	0,76711	0,54322	0,41087	0,32664	0,26411	0,21122
coastal_b6l	512×512	0,81638	0,45780	0,28993	0,21752	0,16360	0,13373	0,09814
coastal_b7	1024×1024	1,63261	1,09017	0,82391	0,66092	0,54639	0,46259	0,39371
coastal_b8	2048×2048	1,77795	1,18395	0,80468	0,57762	0,44084	0,34949	0,28795
europa3	557×600	4,51808	3,79911	3,33670	3,00004	2,73961	2,53042	2,35396
marstest	512×512	2,89312	2,29355	1,93206	1,68176	1,49508	1,35336	1,23317
lunar	512×512	2,73184	2,15055	1,79757	1,55219	1,36909	1,23189	1,11657
spot-la_b3	500×500	2,86430	2,22738	1,84197	1,57548	1,37938	1,21347	1,07920
spot-la_panchr	1000×1000	2,39958	1,82608	1,49580	1,26921	1,09727	0,95881	0,84352
Średnia	–	2,00700	1,47681	1,18046	0,98955	0,85475	0,75133	0,66812

10.4. Bezstratna kompresja sekwencji wideo

Do istotnych zastosowań kompresji bezstratniej należy zaliczyć kodowanie sekwencji wideo. W tej kategorii stawia się duży nacisk na implementację z zastosowaniem trybu rzeczywistego. A to oznacza, że przy tak dużej liczbie kodowanych pikseli niezbędne są wysoce efektywne implementacje sprzętowe.

W artykule [7] zaprezentowano analizę bezstratnego kodowania sekwencji wideo, w której porównano metody LOPT-3D, GLICBAWLS-3D, JPEG-LS oraz JPEG2000. Na podstawie analizy złożoności implementacyjnej pod kątem systemów czasu rzeczywistego w pracy [7] zaproponowano kodowanie sekwencji wideo jako zbioru niezależnych klatek, wykorzystując zredukowaną wersję standardu JPEG2000. Umożliwia ona kompresję również w trybie *visually lossless* (z niewielką stratnością niezauważalną dla człowieka). Propozycje takie jak GLICBAWLS-3D, LOPT-3D oraz jej rozwinięcie do metody LPOSTC [10] należą do rozwiązań o wysokiej efektywności, lecz mają zbyt duże wymagania obliczeniowe, aby mogły zostać wdrożone jako implementacje czasu rzeczywistego.

Istnieje stosunkowo niewiele propozycji sprzętowej realizacji systemu bezstratnej kompresji sekwencji wideo. Jednym z rozwiązań znanych z literatury jest system zaprezentowany w pracy [33]. Jednak wykorzystane w nim mechanizmy są nastawione głównie na możliwość równoległego przetwarzania niezależnych bloków obrazu o wielkości 64×64 piksele, z zastosowaniem do tego celu prostego kodera różnicowego i uproszczonej wersji kodu Huffmana, co nie pozwala osiągnąć wysokiej efektywności kompresji. Na przykład dla owej propozycji średnia bitowa dla obrazu Lennagrey o rozmiarach 512×512 pikseli w ośmiobitowej głębi odcieni szarości wynosi 4,609 bitu na piksel. Z porównania tego rezultatu z wynoszącą 4,0006 bitu na piksel średnią dla metody Blend-13 (wykorzystującej algorytm zaprojektowany na potrzeby realizacji sprzętowej, patrz podrozdział 9.3) wynika, że drugą metodą otrzymujemy wartość o 13% miejszą niż pierwszą.

Podstawową zasadą bezstratnego kodowania sekwencji wideo jest wykorzystanie zarówno zależności przestrzennej (tryb *intraframe* – w ramach aktualnie kodowanego obrazu), jak i zależności czasowej (tryb *interframe*) uwzględniającej sąsiednie klatki sekwencji. Najczęściej jest stosowane kodowanie tzw. grupy obrazów GoP (ang. *Group of Picture*), polegające na tym, że pierwsza klatka jest kodowana w trybie *intraframe*, a pozostałe $N_{GoP} - 1$ klatek w trybie *interframe*. Ułatwia to szybki dostęp do dowolnego fragmentu sekwencji wideo z dokładnością do N_{GoP} klatek, natomiast w sytuacji kodowania całej sekwencji w trybie *interframe* poprawne zdekodowanie ostatniej klatki możliwe jest wyłącznie po zdekodowaniu wszystkich wcześniejszych klatek.

Jak podają autorzy pracy [10], istnieje tylko kilka artykułów dotyczących bezstratnej kompresji sekwencji wideo, dlatego trudno jest nawet o rzetelne wyniki porównawcze. Z tego powodu skuteczność proponowanej tu metody Blend-13 oceniono wcześniej głównie w trybie *intraframe*, a poniżej zostanie pokazany jej dalszy wzrost efektywności dzięki wprowadzeniu trybu *interframe*.

Wśród metod stratnej kompresji sekwencji wideo najpopularniejszą techniką wyznaczania zależności między sąsiednimi klatkami jest estymacja ruchu wykorzystująca podział obrazu na kwadratowe bloki (najczęściej 8×8 lub 16×16 pikseli), dla których wyznacza się indywidualne wektory ruchu, a następnie do każdego kodowanego bloku dołącza się informacje o jego przesunięciu względem poprzedniej klatki F_{i-1} (tzw. referencyjnej, jednej lub wielu). Taką technikę w odniesieniu do kodowania bezstratnego zastosowano w pracy [79], jednak metodę zawartą w owej pracy należy uznać za cechującą się wysoką złożonością obliczeniową. W pracy [91] zastosowano technikę kompensacji ruchu także z wykorzystaniem jednej klatki referencyjnej F_{i-1}, lecz oparto się na całkowitoliczbowej transformacji falkowej IWT (ang. *Integer Wavelet Transform*) bez potrzeby dołączania wektorów ruchu do strumienia danych. Dodatkowo uwzględniono detekcję opłacalności między kodowaniem przestrzennym a czasowym. Natomiast w pracy [17] opisano metodę przewidywania wartości aktualnego piksela na podstawie piksela z poprzedniej klatki, o współrzędnych, które obliczono z wykorzystaniem wektora ruchu wyznaczonego na podstawie dwóch poprzednich klatek. Przyjęto zasadę, iż krótkoterminowy ruch między klatką F_{i-2} a klatką F_{i-1} będzie kontynuowany w ten sam sposób między klatką F_{i-1} a bieżącą F_i. W tej metodzie wektor ruchu jest wyznaczany dla każdego piksela indywidualnie. W najbardziej złożonej obliczeniowo metodzie zaprezentowanej w artykule [68] do wyznaczenia zależności czasowej

wykorzystano aż pięć klatek referencyjnych. Algorytmy estymacji ruchu są powszechnie stosowane głównie w metodach kodowania sekwencji wideo w trybie stratnym, szerzej można przeczytać o tym, zapoznając się np. z grupą standardów MPEG [15].

Każda z tych metod wymaga dodatkowych nakładów obliczeniowych związanych z wyznaczaniem wektorów ruchu. Dodatkowo należy uwzględniać dość złożony mechanizm detekcji zmiany sceny, który powinien aktywować przejście w tryb *intraframe* na czas kodowania każdej pierwszej klatki z nowej sceny [146]. W pracy [6] zrezygnowano z kompensacji ruchu, uzyskując dobre rezultaty przy zmianach scen dzięki dość złożonej obliczeniowo technice Octopus.

Propozycja wykorzystania metody Blend-13 także została pozbawiona analizy wektora ruchu, a kodowanie odbywa się z wykorzystaniem 13 predyktorów działających w trybie *intraframe* oraz jednego odwołującego się do poprzedniej klatki (ta wersja będzie określana dalej jako Blend-V). Pozbycie się kompensacji ruchu jest dużym uproszczeniem, które wynikało ze zdefiniowanych wcześniej zadań określonych w projekcie badawczym nad zaprojektowaniem sprzętowej realizacji metody Blend-V. Jako predyktor trybu *interframe* wykorzystano wartość piksela $P_{i-1}(0)$ z poprzedniej klatki (F_{i-1}), o współrzędnych aktualnie kodowanego piksela $x_n = P_i(0)$. Technika mieszania predykcyjnego automatycznie redukuje negatywny wpływ predyktora trybu *interframe* w sytuacji zmiany sceny. Należy podkreślić, że użycie prostego predyktora, który nie uwzględnia wektora ruchu, daje zaskakująco dobre rezultaty, a wynika to z faktu, iż w wielu sytuacjach duże fragmenty obrazu charakteryzują się statycznością (najczęściej tło sceny). Na podstawie analizy zestawu sekwencji wideo dobrano kompromisową wartość istotności predyktora międzyklatkowego $\alpha_{14} = 6$ (pozostałe wartości α_j zamieszczono w tab. 9.1).

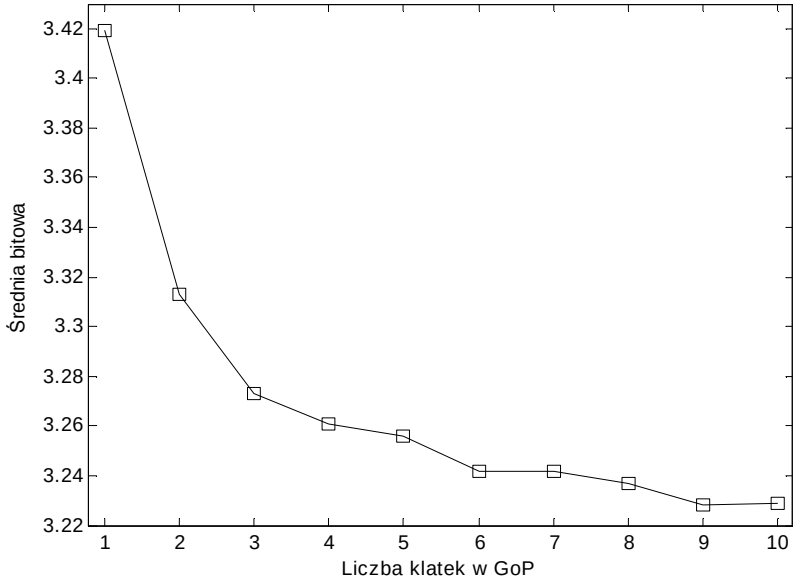
Wpływ liczby N_{GoP} klatek tworzących GoP na średnią bitową sekwencji dla składowej luminancji przedstawiono w tab. 10.13 (przykład dla sekwencji Carphone przedstawia rys. 10.7). Dla $N_{GoP} = 1$ otrzymujemy tryb *intraframe* dla całej kodowanej sekwencji, a $N_{\max}$ oznacza, że w całej sekwencji jedynie pierwsza klatka była kodowana w trybie *intrafame*.

Tab. 10.13. Średnia bitowa w zależności od liczby N_{GoP} klatek w grupie obrazów kodowanych metodą Blend-V

Sekwencja	N_{GoP} = 1	N_{GoP} = 2	N_{GoP} = 3	N_{GoP} = 4	N_{GoP} = 5	N_{GoP} = 6	N_{GoP} = 7	N_{GoP} = 8	N_{GoP} = 9	N_{GoP} = 10	N_{GoP} = 20	$N_{\max}$
Carphone	3,419	3,313	3,273	3,261	3,256	3,242	3,242	3,237	3,228	3,229	3,212	3,207
Foreman	3,761	3,660	3,632	3,613	3,605	3,602	3,594	3,589	3,592	3,590	3,578	3,569
Tennis	4,952	4,561	4,421	4,353	4,317	4,283	4,268	4,268	4,237	4,239	4,194	4,154

Metodę Blend-V wybrano jako przykład kodowania sekwencji wideo ze względu na cechy sprzyjające jej implementacji sprzętowej. Projektując układ sprzętowy, należy tak określić kompromis między efektywnością kompresji a wydajnością, aby system mógł bez przeszkód realizować kodowanie w czasie rzeczywistym. Ze względu na zależności międzyklatkowe występujące podczas dekodowania zdecydowano się na wprowadzenie mechanizmu potokowego kodowania/dekodowania poszczególnych klatek w grupie obrazów o wielkości $N_{GoP} = 10$ [123]. Większe wartości N_{GoP} nie dają dużego wzrostu efektywności, natomiast

wydłużenie potoku wiązałoby się z dalszym wzrostem zasobów sprzętowych i utrudnionym dostępem do dowolnego fragmentu zakodowanej sekwencji.



Rys. 10.7. Wartość średniej bitowej sekwencji Carphone w zależności od liczby N_{GoP} klatek w grupie obrazów

W tabelach 10.14 oraz 10.15 przedstawiono przykłady porównania efektywności kompresji projektowanego systemu, określanego tu mianem Blend-V, z wynikami koderów JPEG-LS oraz CPC [146]. Wyniki dotyczą 30 klatek sygnału luminancji sekwencji Salesman i Tennis. Badanie przeprowadzono zarówno dla trybu bezstratnego ($d = 0$), jak i prawie bezstratnego z maksymalnym błędem $d = \{1, 2, 3, 4\}$. Pomiary metody Blend-V dokonano dla trybu *intraframe* ($N_{GoP} = 1$), a także dla trybów *interframe* z parametrem $N_{GoP} = 10$ oraz $N_{\max} = 30$.

Tab. 10.14. Średnia bitowa dla sekwencji Salesman (30 klatek o rozdzielczości 352 × 288) w zależności od wartości maksymalnego błędu *d* przy kodowaniu prawie bezstratnym

d	JPEG-LS	CPC [146]	Blend-V *intraframe*	Blend-V $N_{GoP} = 10$	Blend-V $N_{GoP} = 30$
0	4,377	3,760	4,10174	3,68598	3,65624
1	2,872	2,321	2,62296	2,22523	2,19701
2	2,243	1,765	2,01064	1,64664	1,61997
3	1,868	1,453	1,64413	1,30918	1,28594
4	1,617	1,252	1,39269	1,07070	1,04752

Tab. 10.15. Średnia bitowa dla sekwencji Tennis (30 klatek o rozdzielczości 352 × 240) w zależności od wartości maksymalnego błędu *d* przy kodowaniu prawie bezstratnym

d	JPEG-LS	CPC [146]	Blend-V *intraframe*	Blend-V $N_{GoP} = 10$	Blend-V $N_{GoP} = 30$
0	5,623	4,729	5,38558	4,47792	4,37834
1	4,064	3,215	3,83947	2,96826	2,87302
2	3,363	2,566	3,14764	2,32954	2,24263
3	2,892	2,212	2,70944	1,94422	1,86435
4	2,572	1,970	2,39452	1,68312	1,60922

10.5. Kodowanie obrazów kolorowych

10.5.1. Korygowanie błędów predykcji

Przygotowując system bezstratnego kodowania obrazów kolorowych, należy wybrać odpowiedni sposób usuwania korelacji między składowymi *RGB* (ang. *spectral correlation*). W literaturze pojawiają się dwa typy dekorelacji, które zostały w tym podrozdziale przebadane.

Pierwszy typ dekorelacji został zaprezentowany w pracach [6] oraz [60]. Autorzy zaproponowali go ze względu na niewielką złożoność obliczeniową, gdyż wprowadza się do zwykłego predykcyjnego algorytmu kodującego tylko dwie dodatkowe operacje odejmowania błędów predykcji sąsiedniej składowej. Błąd predykcji e_R składowej R (w pracy [60] składowej G) jest kodowany bez uwzględnienia zależności spektralnej, natomiast błąd predykcji e_G składowej G, powstały po kodowaniu dowolną metodą predykcyjną, należy pomniejszyć o wartość błędu predykcji składowej R, uzyskując skorygowany błąd predykcji:

$$\tilde{e}_G = e_G - e_R . \tag{10.5}$$

Podobnie korygujemy błąd predykcji składowej B:

$$\tilde{e}_B = e_B - e_G . \tag{10.6}$$

W pracach [6] oraz [60] oba skorygowane błędy predykcji wyznacza się na podstawie tych samych zależności między błędami predykcji.

Choć operacje mają niską złożoność obliczeniową, co pozwala na szybką realizację w trybie przetwarzania sekwencyjnego, to w przypadku modułowego projektowania kodera wymagana byłaby dodatkowa synchronizacja w środkowej części układu, tak aby trzy kodery predykcyjne równolegle wykonujące obliczenia mogły, po wyznaczeniu błędów predykcji każdej składowej koloru, wymienić się informacjami o ich wartościach w celu wprowadzenia korekcji spektralnej. Podczas dalszego etapu kodowanie arytmetyczne trzech błędów predykcji ($e_R, \tilde{e}_G, \tilde{e}_B$) może ponownie przebiegać w sposób równoległy.

10.5.2. Dekorelacja transformacyjna

Drugą możliwością dekorelacji spektralnej jest zastosowanie bezstratnej (odwracalnej) transformacji kolorów z sygnału *RGB* na zestaw trzech innych składowych. W pracy [56] przedstawiono porównanie czterech podstawowych transformacji wraz z ich analizą przydatności do bezstratnej falkowej kompresji obrazów.

W niniejszej pracy wszystkie cztery transformacje zostały przebadane i porównane z rezultatami opisanej w podrozdziale 10.5.1 dekorelacji spektralnej opartej na różnicach błędów predykcji poszczególnych składowych.

Najsłabsze rezultaty uzyskano, stosując wymagającą obliczeń zmiennopozycyjnych transformację $Y_L U_L V_L$, która opiera się na podstawowej transformacji YC_bC_r [15] wykorzysty-

wanej w takich standardach, jak JPEG czy MPEG [98]. Poniżej przedstawiono wzory transformacji $Y_L U_L V_L$:

$$\begin{aligned} Y_L &= G + \left\lfloor \frac{0{,}299R + 0{,}114B}{0{,}587} \right\rfloor, \\ U_L &= B - \lfloor 0{,}587 Y_L \rfloor, \\ V_L &= R - \lfloor 0{,}587 Y_L \rfloor. \end{aligned} \tag{10.7}$$

Transformacja odwrotna ma następującą postać:

$$\begin{aligned} R &= V_L + \lfloor 0{,}587 Y_L \rfloor, \\ B &= U_L + \lfloor 0{,}587 Y_L \rfloor, \\ G &= Y_L - \left\lfloor \frac{0{,}299R + 0{,}114B}{0{,}587} \right\rfloor. \end{aligned} \tag{10.8}$$

Drugi rodzaj transformacji, $O_1O_2O_3$, pozwala uzyskać nieco lepsze rezultaty, choć wymaga operacji dzielenia przez 3, co w porównaniu z kolejnymi transformacjami, jest wadą w przypadku wymagań stawianych przy projektowaniu prostej realizacji sprzętowej. Poniżej przedstawiono wzory transformacji $O_1O_2O_3$:

$$\begin{aligned} O_1 &= \left\lfloor \frac{R+G+B}{3} + 0{,}5 \right\rfloor, \\ O_2 &= \left\lfloor \frac{R-B+1}{2} \right\rfloor, \\ O_3 &= B - 2G + R. \end{aligned} \tag{10.9}$$

Transformacja odwrotna ma następującą postać (w pracy [56] przedstawiono błędnie wzory, zamieniając składowe R i B):

$$\begin{aligned} R &= O_1 + O_2 + O_3 - \left\lfloor \frac{O_3+1}{2} \right\rfloor - \left\lfloor \frac{O_3}{3} + \frac{1}{2} \right\rfloor, \\ G &= O_1 - \left\lfloor \frac{O_3}{3} + \frac{1}{2} \right\rfloor, \\ B &= O_1 - O_2 + \left\lfloor \frac{O_3+1}{2} \right\rfloor - \left\lfloor \frac{O_3}{3} + \frac{1}{2} \right\rfloor. \end{aligned} \tag{10.10}$$

Trzeci rodzaj transformacji to *Y'I'Q'*. Jej uproszczoną wersję, dającą tę samą efektywność, zaprezentowano w pracy [71] pod nazwą YC_oC_g. Transformacja wymaga wykonywania wyłącznie operacji dodawania, odejmowania oraz przesunięć bitowych. Poniżej przedstawiono wzory transformacji YC_oC_g:

$$
\begin{aligned}
Y &= \left\lfloor \frac{R + 2G + B}{4} \right\rfloor, \\
C_o &= R - B, \\
C_g &= G - \left\lfloor \frac{R + B}{2} \right\rfloor.
\end{aligned}
\tag{10.11}
$$

Transformacja odwrotna ma następującą postać:

$$
\begin{aligned}
B &= Y - \left\lfloor \frac{C_o}{2} \right\rfloor - \left\lfloor \frac{C_g}{2} \right\rfloor, \\
G &= Y + \left\lfloor \frac{C_g + 1}{2} \right\rfloor, \\
R &= B + C_o.
\end{aligned}
\tag{10.12}
$$

Najczęściej wykorzystywaną transformacją jest YD_bD_r, która jest stosowana np. w kodowaniu bezstratnym standardu JPEG2000 pod nazwą RCT (ang. *Reversible Color Transform*) [1]. Również i w tym przypadku obliczenia wymagają jedynie operacji dodawania, odejmowania oraz przesunięć bitowych. Poniżej przedstawiono wzory tej transformacji:

$$
\begin{aligned}
Y &= \left\lfloor \frac{R + 2 \cdot G + B}{4} \right\rfloor, \\
D_b &= B - G, \\
D_r &= R - G.
\end{aligned}
\tag{10.13}
$$

Transformacja odwrotna ma następującą postać:

$$
\begin{aligned}
G &= Y - \left\lfloor \frac{D_b + D_r}{4} \right\rfloor, \\
B &= D_b + G, \\
R &= D_r + G.
\end{aligned}
\tag{10.14}
$$

W tabeli 10.16 przedstawiono, dla omówionych powyżej transformacji, zakresy wartości poszczególnych składowych oraz liczbę bitów niezbędnych do ich reprezentacji. Wynika z niej, że do zapisu składowych transformacji $Y_LU_LV_L$ potrzeba 27 bitów, w przypadku pozostałych trzech transformacji wystarcza 26 bitów.

Tab. 10.16. Zakresy wartości składowych oraz liczba bitów niezbędnych do ich reprezentacji

Transformacja	Składowa	Przedział wartości	Liczba bitów
$Y_LU_LV_L$	Y_L	<0; 434>	9
	U_L	<–225; 227>	9
	V_L	<–178; 180>	9
$O_1O_2O_3$	O_1	<0; 255>	8
	O_2	<–127; 128>	8
	O_3	<–510; 510>	10
YC_oC_g	Y	<0; 255>	8
	C_o	<–255; 255>	9
	C_g	<–255; 255>	9
YD_bD_r	Y	<0; 255>	8
	D_b	<–255; 255>	9
	D_r	<–255; 255>	9

10.5.3. Badania eksperymentalne

W przeprowadzonych badaniach transformacja YD_bD_r dała najlepszy rezultat w obu testach. Pierwszy z nich dotyczył analizy wartości entropii 30 obrazów testowych poddanych poszczególnym transformacjom. Wyniki zaprezentowano w tab. 10.17, uwzględniającej także pomiar sumarycznej entropii dla składowych *RGB*, dla której uzyskano średnią 21,484 bitu na piksel. Użycie transformacji YD_bD_r pozwoliło zmniejszyć tę średnią do 19,654 bitu na piksel, co daje oszczędność 8,52%.

Przy kolejnych pomiarach zestaw obrazów podzielono na dwie kategorie. Pierwsze 27 to obrazy naturalne, druga grupa składa się z trzech obrazów z kategorii „artystycznych", czyli sztucznie wygenerowanych (clegg, frymire, serrano). W badaniach zastosowano modelowanie predykcyjne. Jako wartość przewidywaną wykorzystano predyktor MED użyty w algorytmie JPEG-LS (patrz podrozdział 3.1.1), co dla obrazów naturalnych pozwoliło uzyskać bez użycia transformacji średnią wartość entropii na poziomie 13,161 bitu na piksel (patrz tab. 10.18). W tym przypadku przedstawiono też wyniki kodowania (kolumna ΔRGB) z wykorzystaniem korekcji sygnałów różnicowych opartej na wzorach (10.5) i (10.6), uzyskując średnią 12,009 bitu na piksel, co stanowi poprawę o 8,75%. Również w tym badaniu najlepsze rezultaty otrzymano dla transformacji YD_bD_r, która pozwoliła zmniejszyć średnią do 11,583 bitu na piksel, co dało oszczędność aż o 11,99%. W kategorii obrazów artystycznych zastosowanie transformacji powodowało wzrost średniej wartości entropii, jednak najmniejszą stratę odnotowano dla transformacji YD_bD_r, natomiast najlepsze rezultaty uzyskano przy kodowaniu z wykorzystaniem korekcji sygnałów różnicowych ΔRGB.

Tab. 10.17. Entropia obrazów kolorowych dla różnych rodzajów transformacji barw

Obrazy	Szerokość	Wysokość	*RGB*	$Y_LU_LV_L$	$O_1O_2O_3$	YC_oC_g	YD_bD_r
announcer	512	480	22,306	18,484	18,471	18,473	18,049
barbara	720	576	22,836	19,754	20,119	20,104	20,265
beachgirl	480	512	21,694	20,920	19,934	19,934	20,284
bikes	768	512	22,170	18,745	19,199	19,311	19,373
bluegirl	480	512	22,299	19,779	18,892	18,894	19,161
cablecar	512	480	21,762	19,990	19,538	19,545	19,706
clegg	814	880	21,896	23,353	23,113	23,174	22,864
cornfield	512	480	22,370	20,196	20,352	20,327	20,053
flower	512	480	22,206	19,639	19,670	19,673	19,956
fruits	512	480	21,622	21,336	20,982	21,001	21,005
frymire	1118	1105	13,708	17,461	15,587	15,500	14,922
girl	480	512	22,137	19,660	19,051	19,031	19,119
girl_ldisk	512	480	22,205	18,604	17,743	17,741	18,152
hustler	512	480	22,224	19,522	19,054	19,063	18,992
kids	512	480	22,545	21,492	20,864	20,843	20,757
lena	512	512	21,816	20,788	20,253	20,337	20,606
masuda1	512	480	21,029	18,982	18,767	18,647	18,367
masuda2	512	480	19,567	18,943	18,727	18,579	18,541
mobile	720	576	21,033	19,678	19,768	19,801	19,583
model	512	480	21,684	18,663	18,007	17,990	18,121
monarch	768	512	21,469	20,297	20,322	20,265	20,613
parrots	768	512	22,083	20,550	21,330	21,485	21,107
pens	512	480	22,479	20,119	19,959	19,961	19,611
peppers	512	512	21,912	21,814	21,943	22,083	22,069
sail	768	512	21,381	19,238	18,653	18,639	18,850
serrano	629	794	16,998	20,912	19,359	19,191	17,732
soccer	512	480	22,156	20,256	19,699	19,683	19,785
tanaka	512	480	21,777	19,974	19,053	19,029	19,442
tulips	768	512	22,400	22,425	22,318	22,308	21,824
yacht	512	480	22,770	21,167	20,651	20,650	20,717
Średnia	–	–	21,484	20,091	19,713	19,709	19,654

Na podstawie powyższych badań do systemu Blend-C, kodującego w oparciu o metodę Blend-13 obrazy barwne, została wybrana transformacja YD_bD_r dla trybu bezstratnego, natomiast w trybie prawie bezstratnym całkowicie zrezygnowano z transformacji. Wynika to z faktu, iż użycie wzorów (10.1) i (10.2) do kwantyzacji błędów predykcji poszczególnych składowych YD_bD_r nie gwarantuje dla tych składowych zachowania maksymalnego błędu d po konwersji powrotnej do postaci *RGB*. Problem ten jest trudny do analizy i został naświetlony w pracy [89]. Z kolei zastosowanie wzorów (10.3) i (10.4), dokonujących kwantyzacji kolorów *RGB*, a następnie transformacji i bezstratnej kompresji błędów predykcji przy $d > 1$, daje niższy stopień kompresji niż zastosowanie kompresji prawie bezstratnej bez użycia transformacji.

Tab. 10.18. Entropia obrazów kolorowych po zastosowaniu predyktora MED dla różnych rodzajów transformacji barw

Obrazy	*RGB*	$Y_LU_LV_L$	$O_1O_2O_3$	YC_oC_g	YD_bD_r	ΔRGB
announcer	12,358	10,161	10,047	9,987	10,004	10,699
barbara	15,907	11,676	11,215	11,175	11,294	12,937
beachgirl	11,547	11,331	11,297	11,241	11,209	11,715
bikes	16,755	12,022	12,120	12,066	12,034	12,434
bluegirl	11,671	11,298	11,316	11,264	11,232	11,709
cablecar	13,930	12,990	12,977	12,944	13,018	13,408
cornfield	15,298	13,847	14,035	14,021	13,894	14,226
flower	12,538	11,844	11,887	11,850	11,859	12,119
fruits	13,056	12,690	12,973	12,900	12,680	12,635
girl	12,802	12,242	12,268	12,212	12,210	12,635
girl_ldisk	11,417	9,451	9,346	9,277	9,261	10,050
hustler	11,621	9,701	9,595	9,533	9,478	10,167
kids	13,688	12,712	12,676	12,658	12,617	12,856
lena	14,408	14,219	14,104	14,166	14,168	14,041
masuda1	10,912	10,736	10,784	10,719	10,649	11,000
masuda2	11,013	10,956	11,025	10,951	10,888	11,334
mobile	15,106	12,594	12,482	12,469	12,557	12,936
model	11,828	11,266	11,269	11,225	11,200	11,602
monarch	12,457	10,041	9,950	9,872	10,008	10,374
parrots	11,713	9,402	9,539	9,478	9,419	9,751
pens	13,384	12,563	12,595	12,561	12,498	12,846
peppers	12,603	10,817	10,691	10,741	10,808	11,154
sail	16,286	11,629	11,398	11,366	11,320	11,866
soccer	14,600	13,458	13,456	13,445	13,391	13,612
tanaka	11,846	11,670	11,652	11,606	11,647	12,261
tulips	13,483	11,412	11,268	11,263	11,178	11,331
yacht	13,128	12,271	12,265	12,243	12,224	12,549
Średnia	13,161	11,667	11,638	11,601	11,583	12,009
clegg	5,799	6,028	5,973	6,012	6,018	5,965
frymire	4,655	5,095	4,674	4,633	4,492	4,215
serrano	4,115	4,679	4,615	4,598	4,156	4,004
Średnia	4,856	5,267	5,087	5,081	4,889	4,728

Tabela 10.19 zawiera porównanie efektywności proponowanej metody Blend-C i systemu sprzętowej realizacji kompresji ELIC (ang. *Enhanced Lossless Image Compression*) firmy GEMAC [30]. W tabeli znajdują się pomiary stopnia kompresji w trybie bezstratnym i prawie bezstratnym dla dwóch zestawów obrazów kolorowych. Z użyciem systemu ELIC uzyskano lepsze rezultaty w kategorii obrazów „artystycznych", choć przewaga zmniejsza się wraz ze wzrostem maksymalnego błędu *d*. Jednak zdecydowanie lepszy stopień kompresji obrazów naturalnych uzyskano metodą Blend-C, a przewaga ta wzrasta jeszcze przy wyższych wartościach *d*.

Tab. 10.19. Stopień kompresji w trybie prawie bezstratnym dla wartości błędu $d = \{0, 1, 2, 4, 8\}$

Obrazy	Blend-C $d = 0$	ELIC $d = 0$	Blend-C $d = 1$	ELIC $d = 1$	Blend-C $d = 2$	ELIC $d = 2$	Blend-C $d = 4$	ELIC $d = 4$	Blend-C $d = 8$	ELIC $d = 8$
lena	1,869	1,33	2,884	1,95	3,788	2,40	5,634	3,14	10,364	4,34
monarch	2,741	1,84	3,841	2,76	5,298	3,34	8,383	4,20	13,515	5,68
peppers	2,502	1,69	3,560	2,57	4,826	3,18	7,530	4,03	14,432	5,44
sail	2,322	1,50	2,444	2,16	3,072	2,58	4,187	3,16	6,233	4,03
tulips	2,472	1,63	3,420	2,44	4,519	2,97	6,572	3,77	10,716	4,89
Średnia	2,381	1,598	3,230	2,376	4,300	2,894	6,461	3,660	11,052	4,876
clegg	1,758	3,88	2,587	4,77	3,187	5,45	4,144	6,51	5,839	8,06
frymire	2,727	6,41	3,516	6,80	4,263	7,38	5,372	8,26	6,983	9,60
serrano	2,889	7,23	4,170	8,21	5,284	8,96	7,064	10,28	10,161	12,28
Średnia	2,458	5,840	3,424	6,593	4,244	7,263	5,527	8,350	7,661	9,980

Kolejne porównanie (tym razem rozwiązań programowych), które znajduje się w tab. 10.20, dotyczy tego samego zestawu ośmiu obrazów testowych z bazy Waterloo ColorSet. Najlepsze rezultaty uzyskano metodą BMF zarówno dla grupy obrazów naturalnych, jak i „artystycznych" (które nie zostały pozyskane z wykorzystaniem obiektywu kamery lub aparatu fotograficznego) [77]. W kategorii obrazów naturalnych, dla której zaprojektowano proponowany tu system kompresji, jeszcze tylko z zastosowaniem programu Rkim, charakteryzującego się wysoką złożonością obliczeniową, zdołano uzyskać niższą średnią niż dla Blend-C. Zaprezentowane w pracach [3] i [9] użycie ulepszonych wersji metody Glicbaws (oznaczonych w tab. 10.20, odpowiednio, jako Glicbawls+ i OVP-12), mimo ich wysokiej złożoności obliczeniowej, nie pozwoliło na uzyskanie tak niskiej średniej bitowej, jak otrzymana dla metody Blend-C proponowanej w tej pracy. W kategorii obrazów „artystycznych" i sztucznie generowanych (np. przez programy z kategorii CAD) znacznie lepiej sprawdzają się metody kompresji wykorzystujące kodowanie słownikowe [98].

Tab. 10.20. Średnia bitowa obrazów kolorowych po zastosowaniu metody Blend-C w porównaniu z innymi implementacjami programowymi

Obrazy	Blend-C	Pngcrusch [77]	LOCOE [77]	Glicbawls [77]	Glicbawls+ [3]	OVP-12 [9]	Rkim [77]	BMF [77]
lena	12,84158	14,51	13,60	12,74	12,51	13,33	12,56	12,28
monarch	8,75736	12,52	11,29	10,14	9,99	8,91	8,37	8,21
peppers	9,59375	12,99	11,75	10,56	9,12	9,36	9,15	9,19
sail	10,33474	16,17	15,61	13,60	10,83	10,56	10,26	10,02
tulips	9,71006	13,85	12,54	10,71	10,20	9,60	9,39	9,31
Średnia	10,24750	14,01	12,96	11,55	10,53	10,35	9,95	9,80
clegg	13,64866	5,41	7,30	15,08	–	–	10,43	4,28
frymire	8,80042	1,63	6,06	12,98	–	–	3,95	1,26
serrano	8,30839	1,71	4,70	11,65	–	–	3,26	1,27
Średnia	10,25249	2,92	6,02	13,24	–	–	5,88	2,27

Podsumowanie

W pracy można wydzielić cztery główne bloki tematyczne. W pierwszym z nich, składającym się z rozdziałów od 1 do 7, omówiono wszystkie podstawowe podejścia dotyczące modelowania predykcyjnego. Przeanalizowano w tych rozdziałach sposoby wyznaczania stałych, statycznych i adaptacyjnych modeli predykcyjnych o różnych poziomach złożoności implementacyjnej.

Drugi blok to rozdział 8, omawiający adaptacyjny koder arytmetyczny z przełączaniem kontekstów, który można określić jako drugi etap bezstratnego kodowania obrazów.

Wykorzystanie całej tej wiedzy pozwoliło na zaprojektowanie i omówienie w trzecim bloku tej pracy (rozdział 9) ostatecznej metody kompresji, która opiera się na możliwości jednoczesnego wykorzystania wielu metod predykcyjnych do ustalenia końcowej przewidywanej wartości piksela. Jest to metoda mieszania predykcyjnego, której różne warianty zademonstrowano w poszczególnych podrozdziałach rozdziału 9. Metody określone mianem Blend-20, Blend-24 oraz Blend-25 spełniają założenie przyjęte w celu pracy i mówiące o tym, aby średnia bitowa dla zbioru testowanych obrazów była niższa od wszystkich pozostałych wyników znanych z literatury.

Ostatni dający się wydzielić w tej pracy blok to opis w rozdziale 10 praktycznych zastosowań uproszczonej wersji mieszania predykcyjnego do bezstratnego i prawie bezstratnego kodowania obrazów i sekwencji wideo.

Zaproponowane przez autora pracy indywidualnie opracowane techniki i najistotniejsze udoskonalenia istniejących rozwiązań polegają na:

– wprowadzeniu redukcji zakresu odcieni jako wstępnego etapu modelowania danych (określając warunki opłacalności zastosowania tej techniki) – patrz podrozdział 1.5;

– opracowaniu metody selekcji współczynników predykcji (uzyskano rezultaty lepsze niż metodą wykorzystującą MMSE z rzędem $r \leq 4$) – patrz podrozdział 2.3.2;

– opracowaniu algorytmu genetycznego z odpowiednio dobranymi parametrami, służącego do wyznaczania liniowych modeli predykcyjnych, co pozwoliło na przedstawienie nowego spojrzenia na minimalizację funkcji średniego błędu predykcji (wykorzystanie odległości Minkowskiego) – patrz podrozdział 2.5;

– opracowaniu metody systematycznego przeszukiwania wybiórczego o złożoności wielomianowej; dzięki tej technice można wyznaczać modele predykcyjne pozwalające uzyskać lepszą efektywność w porównaniu z klasyczną metodą MMSE – patrz podrozdział 2.4.3;

– udoskonaleniu metod GBSW oraz FLBP – patrz podrozdziały 3.1.3 oraz 3.1.4;

– zaprojektowaniu uogólnionej metody wielokontekstowej – patrz podrozdział 3.1.5;

– wprowadzeniu nowatorskiej metody podziału kontekstowego (konteksty główne) pozwalającej na wzrost efektywności wielu metod predykcyjnych znanych z literatury – patrz podrozdział 3.2;

– opracowaniu mieszanej metody korekcji skumulowanego błędu predykcji; w metodzie tej mieszanie odbywa się z użyciem ośmiu różnych podejść do wyznaczania składowej stałej – patrz podrozdział 4.3;

– zaprojektowaniu autorskiej metody szybkiego wyznaczania współczynników predykcji wysokiego rzędu ($r > 10$) – patrz podrozdział 4.5;

– udoskonaleniu prostych adaptacyjnych metod predykcyjnych ALCM$_+$ i CoBALP$_+$ oraz propozycji ich połączenia w szybką metodę M-LMS, która pozwala uzyskać efektywność zbliżoną do siedmiokrotnie wolniejszej metody GLICBAWLS – patrz podrozdziały 5.3, 5.4 oraz 5.5;

– udoskonaleniu złożonych metod adaptacyjnych RLS$_+$, OLS, AVE-WLS1 i AVE-WLS2 – patrz podrozdziały 6.1, 6.2 oraz 6.4;

– wprowadzeniu dodatkowego bloku NLMS pozwalającego na dalszą poprawę jakości przewidywania aktualnie kodowanych pikseli – patrz podrozdział 6.5;

– udoskonaleniu metody predykcyjnej AdNN$_+$ opartej na sieciach neuronowych – patrz podrozdział 7.3;

– opracowaniu kontekstowego kodera arytmetycznego z adaptacją krótkoterminową oraz długoterminową, który wykorzystuje odpowiednio zaprojektowany schemat kwantyzacji błędów predykcji – patrz punkty 8.2, 8.3;

– wprowadzeniu do kodera arytmetycznego wielu tabel kwantyzacji dostosowanych do odpowiednich rozkładów sygnału błędów predykcji – patrz podrozdział 9.8.5;

– zaprojektowaniu kodera arytmetycznego służącego do kompresji bitu znaku błędu predykcji – patrz podrozdział 8.4;

– opracowaniu nowych odmian mieszania predykcyjnego (patrz wzory (9.1), (9.2), (9.11), (9.15)) oraz użyciu mieszania dwupoziomowego – patrz podrozdziały 9.3, 9.7 oraz 9.8;

– wprowadzeniu w metodzie mieszania predykcyjnego wag i zróżnicowanej wielkości otoczeń dla poszczególnych subpredyktorów – patrz podrozdział 9.3;

– zaprojektowaniu własnej metody predykcji wykorzystującej zasadę dopasowania teksturowanego – patrz podrozdział 9.5 i jej rozszerzenie w podrozdziale 9.8.2;

– opracowaniu metod predykcyjnych o zróżnicowanym poziomie złożoności implementacyjnej opartych na mieszaniu predykcyjnym – patrz rozdział 9;

– wprowadzeniu selektywnej aktywacji bloków NLMS oraz korekcji skumulowanego błędu predykcji (aktywacja została wprowadzona na podstawie eksperymentów mających na celu zwiększenie efektywności kompresji) – patrz rozdział 9;

– zaprojektowaniu uproszczonej wersji mieszania predykcyjnego przystosowanej do sprzętowej realizacji kompresji obrazów i sygnałów wideo w trybie bezstratnym i prawie bezstratnym, w tym obrazów kolorowych (o trzech składowych *RGB*) – patrz rozdział 10;

– uzyskaniu przewagi metody Blend-24 nad JPEG2000 (z punktu widzenia miar PSNR oraz MSSIM) w przypadku stratnego kodowania obrazów przy średnich bitowych wyższych od 1 bitu na piksel (tryb *near-lossless* dla obrazów w odcieniach szarości – patrz punkt 10.2).

Celem niniejszej pracy było opracowanie metody, która umożliwi kompresję obrazów cyfrowych uzyskującą średnią efektywność na poziomie wyższym od efektywności klasycznych rozwiązań znanych z literatury, przy czym odnosi się to do kategorii obrazów naturalnych pozyskanych z aparatów i kamer cyfrowych**.**

Założony cel trwających ponad osiem lat badań został osiągnięty, co potwierdzają wyniki zawarte w tabelach od 9.9 do 9.12.

Ważnym aspektem przedstawionym w tej pracy było wykazanie, że osiągnięcie postawionego celu, którym jest wzrost efektywności metody kompresji, nie musi oznaczać nieproporcjonalnego wzrostu złożoności już istniejących rozwiązań. Wykazano to, co można przedstawić na przykładzie porównania najbardziej efektywnej (spośród pojedynczych metod) techniki AVE-WLS-NLMS+ z autorską metodą Blend-20, której czas kodowania obrazu Lennagrey był 4,26 razy krótszy, a rezultaty uzyskane w testach osiągnęły prawie identyczny średni wynik efektywności. Próby stosowania metod wykorzystujących sieci neuronowe, w których można rozbudowywać blok kodowania kosztem dużego wzrostu czasu kodowania, również nie pozwoliły osiągnąć tak wysokiej efektywności, jaką oferują metody mieszane.

W pracy sformułowano hipotezę, że dzięki mieszaniu predykcyjnemu możliwe jest jednoczesne efektywne wykorzystanie wielu metod łączonych ze sobą zarówno w sposób szeregowy, jak i równoległy. Pozwoliło to jednocześnie na możliwość elastycznego dopasowywania efektywności projektowanych odmian metody do założeń dotyczących ograniczeń czasowych, a sama hipoteza została potwierdzona wynikami uzyskanymi dzięki metodom Blend-20 i Blend-24. W technikach tych wykorzystano zbiór równolegle działających bloków, z których każdy zawierał odrębny subpredyktor z opcjonalnie szeregowo połączonymi blokami NLMS oraz korekcji skumulowanego błędu predykcji.

Praca ta może stać się dla grona badaczy inspiracją do projektowania kolejnych metod kompresji, zwłaszcza że postęp technologiczny w najbliższych latach pozwoli na możliwości praktycznego testowania rozwiązań o jeszcze wyższej złożoności implementacyjnej. Należy zatem nakreślić **kierunki dalszych badań**.

Niewątpliwie narzędzie, jakim jest metoda mieszania predykcyjnego, może odegrać kluczową rolę, a jej rozbudowa może nastąpić przez dobór jeszcze efektywniejszych subpredyktorów.

Zarówno metody OLS, jak i jej rozwinięcie WLS opierają się na lokalnym oknie uczącym, na podstawie którego z wykorzystaniem metody MMSE wyznacza się model predykcyjny. Wnioski płynące z pracy [26] pozwalają pójść o krok dalej i stworzyć metodę AVE-WLS, czyli wykorzystać średnią arytmetyczną zbioru predyktorów różnych rzędów wyznaczonych metodą WLS. Przeprowadzona analiza literatury, poszerzona o własne badania, doprowadziła do zaproponowania metod AVE-WLS1 i AVE-WLS2 jako najefektywniejszych (i najbardziej czasochłonnych) adaptacyjnych metod predykcji kodowanego piksela.

Biorąc pod uwagę drugą ścieżkę, którą podążają badacze w celu uzyskania wyższych efektywności, należy zaznaczyć, iż metoda minimalizacji błędu średniokwadratowego nie jest najefektywniejszym rozwiązaniem, co pokazano w pracy [42], w której jako alternatywę zaproponowano minimalizację błędu absolutnego MMAE. Ze względu na wysoką złożoność algorytmu MMAE metody tej nie wykorzystywano dotychczas do adaptacyjnego kodowania, w którym dla każdego kolejno kodowanego piksela ustala się niezależny model predykcyjny,

wykorzystując pewne sąsiedztwo (okno treningowe), tak samo jak w metodach OLS i WLS. Zamiana z kryterium MMSE na MMAE wydaje się najbardziej obiecującym kierunkiem dalszych badań, przy czym warto rozwinąć ten pomysł o wagową minimalizację błędu absolutnego (jak w rozwinięciu metody OLS do WLS), co wymaga dalszej modyfikacji istniejących rozwiązań obliczania modelu predykcyjnego.

Kolejny krok powinien być wzorowany na podejściu, jakie zastosowano, rozwijając metodę WLS do AVE-WLS, a to wiąże się z dalszym wzrostem złożoności implementacyjnej sugerowanego rozwiązania. Jednocześnie biorąc pod uwagę badania przeprowadzone w podrozdziale 2.5.3, warto zaznaczyć, że jeszcze lepszy rezultat od MMAE można uzyskać, gdy użyjemy minimalizacji średniego błędu w sensie wzoru na odległość Minkowskiego z potęgą $M = 0{,}75$. Jednak jak sugerują autorzy pracy [133], algorytmy minimalizujące taką funkcję przy $M < 1$ wymagają jeszcze większego nakładu obliczeniowego w porównaniu z MMAE. Jednoczesne wykorzystanie tych wszystkich kroków może pozwolić na uzyskanie jeszcze efektywniejszego sposobu przewidywania wartości kodowanego piksela. Kilka odmian tego pomysłu może stanowić podstawę do poszerzenia liczby subpredyktorów w metodzie mieszania, którą zaprezentowano w rozdziale 9.

Nie należy też zapominać o samym kodowaniu błędów predykcji. Przedstawione w tej pracy rozwiązanie może zostać poszerzone o większą liczbę iteracji optymalizujących, a ponadto o ewentualne nowe parametry (wyznaczane w zautomatyzowany sposób przy małej liczbie pomiarów) zapisywane w nagłówku pliku wynikowego.

Ponadto omówiony w rozdziale 8 adaptacyjny koder arytmetyczny nie jest jedynym skutecznym podejściem wykorzystywanym w kodowaniu obrazów. Druga gałąź badań skupia się na dopasowywaniu rozkładów prawdopodobieństw (skojarzonych z danym kontekstem) do kodowanych błędów predykcji, przy czym w sposób parametryczny informacje o tych rozkładach są umieszczane w nagłówku pliku. Badania takich rozwiązań przedstawiono w pracach [77, 86, 115, 151].

Do dziś nie rozstrzygnięto, które z tych dwóch podejść pozwala uzyskać lepsze rezultaty w kodowaniu arytmetycznym. Warto tu też zaznaczyć, że ze względu na złożoność implementacyjną, nikt nie powtórzył ani nie kontynuował pomysłu użytego w TMW [86], aby wyznaczać dla każdego kolejno kodowanego piksela mieszanie rozkładów (podobnie jak wykonuje się mieszanie wartości przewidywanych na podstawie zbioru subpredyktorów) skojarzonych z odrębnymi subpredyktorami. Poszerzenie tych badań o dobór większej liczby rozmaitych rozkładów, w połączeniu z przedstawioną w tej pracy metodą mieszania predykcyjnego, może stać się drogą do dalszego wzrostu efektywności bezstratnej kompresji obrazów.

Warto tu także zaznaczyć, iż rozwiązania zaprezentowane w pracy mogą być również wykorzystane do bezstratnej kompresji dźwięku. Wiele z omówionych tu pomysłów doskonale sprawdza się także w odniesieniu do kompresji dźwięku [118, 119]. Będą one wykorzystane w trwających obecnie (w latach 2011–2014) pracach nad projektem „Bezstratny kodek dźwięku z automatyczną identyfikacją i klasyfikacją danych akustycznych”. Kierownikiem tego projektu jest autor niniejszej pracy.

Podsumowując, **znaczący wkład autora w rozwój dyscypliny naukowej**, jaką jest bezstratna kompresja obrazów, polegał na opracowaniu metody, która różni się od innych istniejących w literaturze tym, że wykorzystuje:

– nowe techniki mieszania predykcyjnego, pozwalające na skuteczne łączenie wielu metod znanych z literatury;

– nowatorską metodę podziału kontekstowego, która może być przydatna do zwiększenia efektywności niektórych znanych metod predykcyjnych;

– nieznaną wcześniej w literaturze koncepcję mieszania wielu metod korekcji skumulowanego błędu predykcji;

– rozwiązanie stosujące w sposób efektywny kaskadową technikę NLMS+;

– kontekstowy koder arytmetyczny służący do kompresji bitu znaku błędów predykcji.

Dopiero wszystkie te cechy, w połączeniu z opracowanymi na nowo i udoskonalonymi wieloma rozwiązaniami znanymi z literatury, pozwoliły na osiągnięcie postawionego celu.

Literatura

1. Acharya T., Ray A.K., *Image processing, principles and applications*, Hoboken, New Jersey, John Wiley & Sons, Inc. 2005.
2. Adjeroh D., Bhupathiraju K.V., On lossless image compression using the Burrows-Wheeler Transform, in: *Proceedings of 18th International Conference on Image Processing (ICIP)*, Brussels, Belgium, 11–14 Sept. 2011, [b.w.] 2011, s. 1997–2000.
3. Aiazzi B., Alparone L., Baronti S., Context modeling for near-lossless image coding, *IEEE Signal Processing Letters*, March 2002, vol. 9, no. 3, s. 77–80.
4. Aiazzi B., Baronti S., Alparone L., Near-lossless image compression by relaxation-labeled prediction, *Signal Processing*, November 2002, vol. 82, no. 11, s. 1619–1631.
5. Alecu A., Munteanu A., Cornelis J.P.H., Wavelet-based Scalable L-infinity-oriented Compression, *IEEE Transactions On Image Processing*, September 2006, vol. 15, no. 9, s. 2499–2512.
6. Andriani S., Calvagno G., Lossless Compression of Colour Video Sequence using Optimal Prediction Theory – Octopus, in: *Proceedings of DCC'07*, Snowbird, Utah, 27–29 March 2007, [b.m.], The Printig Houses 2007, s. 375.
7. Andriani S., Calvagno G., Erseghe T., Mian G.A., Durigon M., Rinaldo R., Knee M., Walland P., Koppetz M., Comparison of lossy to lossless compression techniques for digital cinema, in: *Proceedings of International Conference on Image Processing ICIP'04*, Singapore, 24–27 Oct. 2004, vol. 1, [b.w.], 2004, s. 513–516.
8. Andriani S., Calvagno G., Mian G.A., A lossless image coding technique exploiting spectral correlation on the RGB Space, in: *Proceedings of 12th European Signal Processing Conference (EUSIPCO)*, Wien, 6–10 Sept. 2004. Wiedeń, EURASIP 2004, s. 1305–1308.
9. Andriani S., Calvagno G., Mian G.A., Lossless Image Compression using Vector Prediction based on Spectral Correlation, in: *Proceedings of International Conference on Image Processing ICIP'05*, Genova, Italia, 11–14 Sept. 2005, vol. 2, [b.w.] 2005, s. 277–280.
10. Andriani S., Calvagno G., Mian G.A., Lossless Video Compression using a Spatio-Temporal Optimal Predictor, in: *Proceedings of 13th European Signal Processing Conference EUSIPCO-05* [płyta CD], Antalya, Turkey, 4–8 September 2005, [b.w.] 2005.
11. 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 (extended version).
12. Avramovic A., Lossless compression of medical images based on gradient edge detection, In: *Proceedings of 19th Telecommunications Forum (TELFOR)* [płyta CD], Serbia, Belgrade, 22–24 Nov. 2011, [b.m.] 2011, IEEE, s. 1199–1202.
13. Bao P., Wu X., L-infinity constrained near-lossless image compression using weighted finite automata encoding, *Comput. & Graphics* 1998, vol. 22, no. 2–3, s. 217–223.
14. Bekkouche H., Barret M., Adaptive multiresolution decomposition: applications to lossless image compression, in: *Proceedings of IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP'02)*, Orlando, Florida, USA, May 2002, vol. 4, [b.w.] 2002, s. IV_3533–3536.
15. Bhaskaran V., Konstantinides K., *Image and video compression standards – algorithms and architectures*, wyd. 2, Norwell, MA, USA, Kluwer Academic Publishers 1997.
16. Boulgouris N.V., Zaharos S., Strintzis M.G., Adaptive Decorrelation and Entropy Coding for Context-based Lossless Image Compression, in: *First Balkan Conference on Signal Processing, Communications, Circuits, and Systems*, Turkey, June 2000, Istanbul, [b.w.] 2000, s. 1–4.

17. Carotti E.S.G., De Martin J.C., Meo A.R., Backward-Adaptive Lossless Compression of Video Sequences, in: *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP 2002)*, Orlando, Florida, May 2002, vol. 4, [b.w.] 2002, s. 3417–3420.
18. Carpentieri B., Weinberger M.J., Seroussi G., Lossless compression of continuous-tone images, *Proceedings of the IEEE*, November 2000, vol. 88, no. 11, s. 1797–1809.
19. Chang C.-C., Chen G.-I., Enhancement algorithm for nonlinear context-based predictors, *IEE Proceedings Vision, Image and Signal Processing* 2003, vol. 150, no. 1, s. 15–19.
20. Charith G., Abhayaratne K., Modifying Integer Wavelet Transforms for Scalable Near-Lossless Image Coding, *Visual Communications and Image Processing 2003, Proceedings of the SPIE* 2003, vol. 5150, s. 1697–1708.
21. Chen X. i in., Lossless Compression for Space Imagery in a Dynamically Reconfigurable Architecture, in: Proceedings of International Workshop on Applied Reconfigurable Computing (ARC2008), *LNCS*, March 2008, vol. 4943, s. 336–341.
22. Chung Y.-S., Kanefsky M., On 2-D recursive LMS algorithms using ARMA prediction for ADPCM encoding of images, *IEEE Transactions on Image Processing*, vol. 1, no. 3, s. 416–422.
23. Daaboul A., Local prediction for lossless image compression, in: *Proceedings of the Prague Stringology Club Workshop '98*, Czechy, [b.w.] 1998, s. 44–50.
24. Danesh A.S., Rad R.M., Attar A., A novel predictor function for lossless image compression, in: *Proceedings of 2nd International Conference on Advanced Computer Control (ICACC)*, Shenyang, Liaoning, China, 27–29 March 2010, [b.w.] 2010, s. 527–531.
25. Deng G., Adaptive predictor combination for lossless image coding, in: *Proceedings of 10th European Conference on Signal Processing EUSIPCO*, Tampere, Finland, Sept. 2000, [b.m.], EURASIP 2000, s. 1141–1144.
26. Deng G., Transform domain LMS-based adaptive prediction for lossless image coding, *Signal Processing Image Communication*, February 2002, vol. 17, no. 2, s. 219–229.
27. Deng G., Ye H., A general framework for the second-level adaptive prediction, in: *Proceedings of ICASSP '03*, April 2003, vol. 3, [b.w.] 2003, s. III_237–240.
28. Deng G., Ye H., Lossless image compression using adaptive predictor combination, symbol mapping and context filtering, in: *Proceedings of IEEE 1999 International Conference on Image Processing*, Kobe, Japan, Oct. 1999, vol. 4, [b.w.] 1999, s. 63–67.
29. Deng G., Ye H., Marusic S., Tay D., A method for predictive order adaptation based on model averaging, in: *Proceedings IEEE International Conference on Image Processing ICIP'03*, Barcelona, Catalonia, Spain, 14–17 Sept. 2003, vol. 2, [b.w.] 2003, s.189–192.
30. Dittrich C., FPGAs for lossless image/video and universal data compression, *ECE Magazine*, April 2005, s. 42–44.
31. Domański M., *Obraz cyfrowy*, wyd. 1, Warszawa, WKŁ 2010, ISBN 978-83-206-1795-5.
32. Dony R.D., Haykin S., Neural network approaches to image compresion, *Proceedings of the IEEE*, February 1995, vol. 83, no. 2, s. 288–303.
33. Drost G.W., Bourbakis N.G., A hybrid system for real-time lossless image compression, *Microprocessors and Microsystems*, 15 March 2001, vol. 25, no. 1, s. 19–31.
34. Drozdek A., *Elements of Data Compression*, Pacific Grove, CA, Thomson Brooks/Cole 2001, ISBN 053438448X.
35. Dziurzański P., Ulacha G., System-level perspective of computation and communication refinement for a subpredictor-blending-based compression system, *Elektronika, Konstrukcje, Technologie, Zastosowania* 2007, no. 12, s. 86–89.
36. Estrakh D.D., Mitchell H.B., Schaefer P.A., Mann Y., Peretz Y., "Soft" median adaptive predictor for lossless picture compression, *Signal Processing*, September 2001, vol. 81, no. 9, s. 1985–1989.
37. Gallager R.G., Variations on a theme by Huffman, *IEEE Transactions on Information Theory*, November 1978, vol. 24, no. 6, s. 668–674.

38. Golchin F., Paliwal K.K., A lossless image coder with context classification, adaptive prediction and adaptive entropy coding, in: *Proceedings of IEEE International Conference on Acoustics, Speech and Signal Processing*, Seattle, Washington, USA, May 1998, [b.w.] 1998, s. 2545–2548.
39. Golchin F., Paliwal K.K., Classified adaptive prediction and entropy coding for lossless coding of images, in: *Proceedings of International Conference on Image Processing*, Santa Barbara, USA, 26–29 October 1997, vol. 3, [b.w.] 1997, s. 110–113.
40. Goldberg D.E. *Genetic Algorithms in Search, Optimization, and Machine Learning*, Maassachusetts, USA, Addison-Wesley Publishing Company, Inc., Reading 1989.
41. Hartenstein H., Herz R., Saupe D., A comparative study of L_∞-distortion limited image compression algorithms, in: *Proceedings of Picture Coding Symposium*, Berlin, Sep. 1997, [b.w.] 1997, s. 1–5.
42. Hashidume Y., Morikawa Y., Lossless Image Coding Based on Minimum Mean Absolute Error Predictors, in: *Proceedings of SICE Annual Conference 2007*, Kagawa University, Japan, 17–20 September 2007, [b.m.], Society of Instrument and Control Engineers 2007, s. 2832–2836.
43. Hsieh F.-Y., Wang C.-M., Lee C.-C., Fan K.-C., A Lossless Image Coder Integrating Predictors and Block-Adaptive Prediction, *Journal of Information Science and Engineering*, September 2008, vol. 24, no. 5, s. 1579–1591.
44. Hong G., Hall G., Terrell T., Prediction by back-propagation neural network for lossless image compression, in: *Proceedings of 3rd International Conference on Signal Processing*, 14–18 October 1996, vol. 2, [b.w.] 1996, s. 1026–1030.
45. Huang H., Fränti P., Huang D., Rahardja S., Cascaded RLS-LMS prediction in MPEG-4 lossless audio coding, *IEEE Trans. on Audio, Speech and Language Processing*, March 2008, vol. 16, no. 3, s. 554–562.
46. Iordache R., Tabus I., Astola J., Fixed-slope near-lossless context-based image compression, in: *Proceedings of 1998 International Conference on Image Processing*, 4–7 October 1998, vol. 1, [b.w.] 1998, s. 512–515.
47. Itani A., Das M., Adaptive Switching Linear Predictor for Lossless Image Compression, *Lecture Notes in Computer Science* 2005, vol. 3804, s. 718–722.
48. Jakhetiya V., Jaiswal S.P., Tiwari A.K., A novel predictor coefficient interpolation approach for lossless compression of images, in: *Proceedings of Instrumentation and Measurement Technology Conference*, Binjiang, China, 10–12 May 2011, [b.w.] 2011, s. 1–4.
49. Jiang J., Image compression with neural networks – a survey, *Signal Processing: Image Communication* 1999, vol. 14, s. 737–760.
50. Jiang J., Grecos C., Towards an improvement on prediction accuracy in JPEG-LS, *Optical Engineering, SPIE* 2002, vol. 41, no. 2, s. 335–341.
51. Karimi N., Samavi S., Shirani S., Lossless compression of high-throughput RNAi images, in: *Proceedings of 10th IEEE International Conference on Information Technology and Applications in Biomedicine (ITAB)*, Corfu, Greece, 3–5 Nov. 2010, [b.w.] 2010, s. 1–4.
52. Kassim A.A., Pingkun Yan, Wei Siong Lee, Sengupta K., Motion compensated lossy-to-lossless compression of 4-D medical images using integer wavelet transforms, *IEEE Transactions on Information Technology in Biomedicine*, March 2005, vol. 9, no. 1, s. 132–138.
53. Kau L.-J., Lin Y.-P., Least squares-adapted edge-look-ahead prediction with run-length encodings for lossless compression of images, in: *Proceedings of International Conference on Acoustics, Speech and Signal Processing ICASSP 2008*, Las Vegas, Nevada, U.S.A., 31 March–4 April 2008, [b.w.] 2008, s. 1185–1188.
54. Kau L.-J., Lin Y.-P., Lossless image coding using a switching predictor with run-length encodings, in: *Proceedings of IEEE International Conference on Multimedia and Expo*, June 2004, vol. 2, [b.w.] 2004, s. 1155–1158.
55. Kau L.-J., Lin Y.-P., Lin C.-T., Lossless image coding using adaptive, switching algorithm with automatic fuzzy context modelling, *IEE Proceeding Vision, Image and Signal Processing*, October 2006, vol. 153, no. 5, s. 684–694.

56. Khelifi F., Bouridane A., Kurugollu F., Efficient Lossless Colour Image Coding With Speck, in: *Proceedings of 13th European Signal Processing Conference EUSIPCO-05* [płyta CD], Antalya, Turkey, 4–8 September 2005, [b.m.], EURASIP 2005.
57. Kiely A., Klimesh M., *The ICER Progressive Wavelet Image Compressor*, The Interplanetary Network Progress Report 42, no. 155, November 15, 2003.
58. Knezovic J., Kovac M., Gradient based selective weighting of neighboring pixels for predictive lossless image coding, in: *Proceedings of the 25th International Conference on Information Technology Interfaces ITI 2003*, Cavtat, Croatia, 16–19 June 2003, [b.w.] 2003, s. 483–488.
59. Knezovic J., Kovac M., Mlinaric H., A New Adaptive Blending Predictor for Lossless Image Compression, in: *Proceedings of ITI 4th International Conference on Information & Communications Technology ICICT '06*, Cairo, 10–12 December 2006, [b.w.] 2006, s. 1–10.
60. Kobayashi M., Noma M., Hiratsuka S., Kamata S., Lossless compression for RGB color still images, in: *Proceedings of International Conference on Image Processing ICIP'99*, 1999, vol. 4, [b.w.] 1999, s. 73–77.
61. Krivoulets A., A method for progressive near-lossless image compression, in: *Proceedings of International Conference on Image Processing ICIP'03*, Barcelona, Catalonia, Spain, 14–18 September 2003, vol. 2, [b.w.] 2003, s. 185–188.
62. Kuroki Y., Ueshige Y., Ohta T., An estimation of the predictors implemented by shift operation, addition, and/or substraction, in: *Proceedings of International Conference on Image Processing 2001*, [b.w.] 2001, s. 474–477.
63. Kwon M., Kim H.J., Lee C.W., Lee S.U., A lossless image coder with context-based minimizing MSE prediction and entropy coding, in: *Proceedings of International Symposium on Circuits and Systems 1999*, Orlando, Florida, USA, 30 May–2 June 1999, vol. 4, [b.w.] 1999, s. 479–482.
64. Lastri C., Aiazzi B., Alparone L., Baronti S., Virtually Lossless Compression of Astrophysical Images, *EURASIP Journal on Applied Signal Processing*, vol. 2005, no. 15, s. 2521–2535.
65. Lee C.-H., Lai W.-Y., Chen C.-C., Lossless image coding via adaptive Takagi-Sugeno fuzzy neural network predictor, in: *Proceedings of International Conference on Networking, Sensing and Control 2004*, Taipei, Taiwan, 21–23 March 2004, vol. 1, [b.w.] 2004, s. 565–570.
66. Lee W.S., Edge-adaptive prediction for lossless image coding, in: *Proceedings of Data Compression Conference DCC'99*, Snowbird, Utah, 29–31 March 1999. [b.m.], The Printing House 1999, s. 483–490.
67. Leon D., Balkir S., Sayood K., An evolvable predictor for lossless image compression, in: *Proceedings of International Symposium on Circuits and Systems ISCAS 2002*, Scottsdale, Arizona, USA, 26–29 May 2002, vol. 4, [b.w.] 2002, s. IV-731–IV-734.
68. *Lossless Data Compression. Recommendation for Space Data System Standards*, 121.0-B-1. Blue Book. Issue 1, Washington, D.C., CCSDS, May 1997.
69. *Lossless Data Compression. Recommendation for Space Data System Standards*, 120.1-G-1. Green Book. Issue 1, Washington, D.C., CCSDS, June 2007.
70. Maeda H., Minezawa A., Matsuda I., Itoh S., Lossless Video Coding Using Multi-Frame MC and 3D Bi-Prediction Optimized for Each Frame, in: *Proceedings of 14th European Signal Processing Conference EUSIPCO'06* [płyta CD], Florence, Italy, 4–8 September 2006, [b.w.] 2006.
71. Malvar H., Sullivan G., YCoCg-R: *A Color Space with RGB Reversibility and Low Dynamic Range*, JVT-document JVT-I014, Trondheim, Norway, JVT, July 2003.
72. Marcellin M., Gormish M., Bilgin A., Boliek M., An Overview of JPEG2000, in: *Proceedings Data Compression Conference*, Snowbird, Utah, March 2000. [b.m.], The Printing House 2000, s. 523–541.
73. Marusic S., Deng G., A neural network based adaptive non-linear lossless predictive coding technique, in: *Proceedings of the 5th Int. Symposium on Signal Processing and Its Applications ISSPA '99*, Brisbane, Australia, 22–25 August, 1999,[b.w.] 1999, s. 653–656.
74. Marusic S., Deng G., A study of two new adaptive predictors for lossless image compression, in: *Proceedings of IEEE 1997 International Conference on Image Processing*, Oct. 1997, vol. 2, [b.w.] 1997, s. 286–289.

75. Marusic S., Deng G., Adaptive prediction for lossless image compression, *Signal Processing: Image Communications*, May 2002, vol. 17, s. 363–372.
76. 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.
77. Matsuda I., Kaneko N.T., Minezawa A., Itoh S., Lossless Coding of Color Images using Block-Adaptive Inter-Color Prediction, in: *Proceedings of International Conference on Image Processing ICIP 2007*, San Antonio, Texas, USA, 16–19 September 2007, vol. 2, [b.w.] 2007, s. II_329–332.
78. Matsuda I., Ozaki N., Umezu Y., Itoh S., Lossless coding using Variable Blok-Size adaptive prediction optimized for each image, in: *Proceedings of 13th European Signal Processing Conference EUSIPCO-05* [płyta CD], September 2005, [b.w.] 2005.
79. Matsuda I., Shiodera T., Itoh S., Lossless Video Coding Using Variable Block-Size MC and 3D Prediction Optimized for Each Frame, in: *Proceedings of 12th European Signal Processing Conference EUSIPCO 2004*, September 2004. [b.m.], EURASIP 2004, s. 1967–1970.
80. Matsuda I., Shirai N., Itoh S., Lossless Coding Using Predictors and Arithmetic Code Optimized for Each Image, *Lecture Notes in Computer Science* 2003, vol. 2849, s. 199–207.
81. Memon N.D., Sayood K., An Asymmetric lossless image compression technique, in: *Proceedings of the 1995 International Conference on Image Processing*, 23–26 October 1995, vol. 3, [b.w.] 1996, s. 97–100.
82. Memon N.D., Sayood K., Lossless image compression: a comparative study, *Proceedings SPIE* 1995, vol. 2418, s. 8–20.
83. Memon N.D., Wu X., Recent Developments in Context-Based Predictive Techniques for Lossless Image Compression, *The Computer Journal* 1997, vol. 40, s. 127–136.
84. Meyer B., Tischer P., Extending TMW for near lossless compression of greyscale images, in: *Proceedings of Data Compression Conference 1998*, Snowbird, Utah, [b.m.], The Printing House 1998, s. 458–470.
85. Meyer B., Tischer P., GLICBAWLS – Grey Level Image Compression by Adaptive Weighted Least Squares, in: *Proceedings of Data Compression Conference 2001*, Snowbird, Utah, The Printing House 2001, s. 503.
86. Meyer B., Tischer P., TMW – a new method for lossless image compression, in: *Proceedings of International Picture Coding Symposium (PCS97)*, Berlin, Germany, September 1997, [b.w.] 1997, s. 533–538.
87. Meyer B., Tischer P., TMW^{Lego} – An Object Oriented Image Modelling Framework, in: *Proceedings of Data Compression Conference 2001*, Snowbird, Utah, The Printing House 2001, s. 504.
88. Motta G., Storer J. A., Carpentieri B., Improving the performance of adaptive linear prediction coding (ALPC) via least square minimization, in: *Proceedings of 12th International Workshop on Systems, Signals & Image Processing – IWSSIP 2005*, Chalkida, Greece 2005, [b.w.] 2005, s. 335–338.
89. Nakachi T., Fujii T.; Suzuki J., Lossless and near-lossless compression of still color images, in: *Proceedings of International Conference on Image Processing ICIP'99*, vol. 1, [b.w.] 1999, s. 453–457.
90. Nutes P.R.R.L., Segmented optimal linear prediction applied to lossless image coding, in: *Proceedings of International Telecommunications Symposium*, 3–6 September 2006, [b.w.] 2006, s. 524–528.
91. Park S.-G., Delp E. J., Adaptive lossless video compression using an integer wavelet transform, in: *Proceedings of International Conference on Image Processing ICIP'04*, Singapore, 24–27 October 2004, vol. 4, [b.w.] 2004, s. 2251–2254.
92. Pinho A.J., On the impact of histogram sparseness on some lossless image compression techniques, in: *Proceedings of International Conference on Image Processing*, Thessaloniki 2001, [b.w.] 2001, s. 442–445.
93. Przelaskowski A., *Kompresja danych: podstawy, metody bezstratne, kodery obrazów*, wyd. 1, Warszawa, Wydawnictwo BTC 2005, ISBN 83-60233-05-5.

94. Rizvi S.A., Nasrabadi N.M., Lossless image compression using modular differential pulse code modulation, in: *Proceedings of International Conference on Image Processing ICIP 1999*, Kobe, Japan, 24–28 October 1999, vol.1, [b.w.] 1999, s. 440–443.
95. Said A., Pearlman W.A., A New, Fast, and Efficient Image Codec Based on Set Partitioning in Hierarchical Trees, *IEEE Transactions on Circuits and Systems for Video Technology 6*, June 1996, no. 3, s. 243–250.
96. Salami M., Iwata M., Higuchi T., Lossless Image Compression by Evolvable Hardware, in: *Proceedings of The Fourth European Conference on Artificial Life*, Brighton, UK, 28–31 July 1997. [b.m.], MIT Press 1997, s. 407–416.
97. Sanchez V., Nasiopoulos P., Abugharbieh R., Efficient 4D motion compensated lossless compression of dynamic volumetric medical image data, in: *Proceedings of IEEE International Conference on Acoustics, Speech and Signal Processing ICASSP 2008*, Las Vegas, Nevada, USA., 31 March–4 April 2008, [b.w.] 2008, s. 549–552.
98. Sayood K., *Introduction to Data Compression*, wyd. 2, San Francisco, Morgan Kaufmann Publ. 2002, ISBN 1-55860-558-4.
99. Sayood K., *Lossless Compression Handbook*, San Diego, Academic Press 2003, ISBN 0-12--620861-1.
100. Scharcanski J., Lossless and Near-Lossless Compression for Mammographic Digital Images, in: *Proceedings of International Conference on Image Processing ICIP'06*, Atlanta, GA, USA, 8–11 October 2006, [b.w.] 2006, s. 2253–2256.
101. Schuller G., Bin Yu; Dawei Huang, Lossless coding of audio signals using cascaded prediction, in: *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing 2001*, Salt Lake City, Utah, USA, 2001, vol. 5, [b.w.] 2001, s. 3273–3276.
102. Seemann T., Tisher P., *Generalized locally adaptive DPCM*, Department of Computer Science Technical Report CS97/301, Monash University, Australia, 1997, s. 1–15.
103. Seemann T., Tisher P., Meyer B., History-Based Blending of Image Sub-Predictors, in: *Proceedings of Picture Coding Symposium*, Berlin, Germany, 1997, [b.w.] 1997, s. 147–151.
104. Skarbek W., *Metody reprezentacji obrazów cyfrowych*, Warszawa, Akademicka Oficyna Wydawnicza PLJ 1993, ISBN 83-7101-059-1.
105. Skarbek W. i in., *Multimedia algorytmy i standardy kompresji*, Warszawa, Akademicka Oficyna Wydawnicza PLJ 1998, ISBN 83-7101-385-X.
106. Strom J., Cosman P., Medical image compression with lossless regions of interest, *Signal Processing*, June 1997, vol. 59, no. 2, s. 155–171.
107. Strutz T., Context-Based Adaptive Linear Prediction for Lossless Image Coding, in: *4th Proceedings of International ITG Conference on Source and Channel Coding*, Berlin, Germany, 28–30 January, 2002, [b.w.] 2002, s. 105–109.
108. Takamura S., Matsumura M., Yashima Y., A study on an evolutionary pixel predictor and its properties, in: *Proceedings of International Conference on Image Processing ICIP'09*, Egypt, Cairo, 7–11 November 2009, [b.w.] 2009, s. 1921–1924.
109. Takizawa K., Takenouchi S., Aomori H., Otake T., Tanaka M., Matsuda I., Itoh S., Lossless image coding by cellular neural networks with minimum coding rate learning, in: *Proceedings of 20th European Conference on Circuit Theory and Design (ECCTD)*, Linköping, Sweden, 29–31 Aug. 2011, [b.w.] 2011, s. 33–36.
110. Taquet J., Labit C., Near-lossless and scalable compression for medical imaging using a new adaptive hierarchical oriented prediction, in: *Proceedings of International Conference on Image Processing ICIP 2010*, Hong Kong, China, 26–29 Sept. 2010, [b.w.] 2010, s. 481–484.
111. Tian-Hu Yu, A fuzzy logic-based predictor for predictive coding of images, *IEEE Transactions on Fuzzy Systems*, February 1998, vol. 6, no. 1, s. 153–162.
112. Tiwari A.K., Kumar R.V.R., A switched adaptive predictor for lossless compression of high resolution images, in: *Proceedings of International Conference on Communications ICC 2005*, 16–20 May 2005, vol. 2, [b.w.] 2005, s. 1097–1101.

113. Tiwari A.K., Kumar R.V.R., Least squares based optimal switched predictors for lossless compression of images, in: *Proceedings of IEEE International Conference on Multimedia and Expo*, Hannover, Germany, 23–26 April 2008, [b.w.] 2008, s. 1129–1132.
114. Topal C., Gerek Ö.N., Pdf sharpening for multichannel predictive coders, in: *Proceedings of 14th European Signal Processing Conference EUSIPCO-06 CD* [płyta CD], September 2006, [b.w.] 2006.
115. Ueno H., Morikawa Y., A New Distribution Modeling for Lossless Image Coding Using MMAE Predictors, in: *Proceedings of the 6th International Conference on Information Technology and Applications*, Hanoi, Vietnam, 9–12 November 2009, [b.w.] 2009, s. 249–254.
116. Ulacha G., Efektywna metoda kompresji obrazów w trybie bezstratnym i prawie bezstratnym, *Elektronika, Konstrukcje, Technologie, Zastosowania* 2009, no. 7, s. 172–176.
117. Ulacha G., Method of Lossless and Near-Lossless Color Image Compression, *Journal Of Information Science And Engineering* 2011, no. 2, s. 621–642.
118. Ulacha G., Metoda wyznaczania współczynników predykcji liniowej z minimalizacją wartości entropii, *Przegląd Elektrotechniczny* 2012, no. 8, s. 161–165.
119. Ulacha G., Zastosowanie adaptacyjnego wielokontekstowego kodera arytmetycznego do bezstratnej kompresji dźwięku, *Biuletyn Wojskowej Akademii Technicznej* 2009, no. 4, s. 67–78.
120. Ulacha G., Dziurzański P., Lossless and near-lossless image compression scheme utilizing blending-prediction-based approach, in: Proceedings of the International Conference on Computer Vision and Graphics ICCVG 2008, *LNCS*, 21 May 2009, vol. 5337, s. 208–217.
121. Ulacha G., Stasiński R., A New Context-based Lossless Image Coding Method, in: *Proceedings of 13th International Conference on Systems, Signals and Image Processing IWSSIP 2006*, Budapest, Hungary 2006, [b.w.] 2006, s. 215–218.
122. Ulacha G., Stasiński R., A New Simple Context Lossless Image Coding Algorithm Based on Adaptive Context Arithmetic Coder, in: *Proceedings of 15th International Conference on Systems, Signals and Image Processing IWSSIP 2008*, Bratislava, Slovak Republic, 25–28 June 2008, [b.w.] 2008, s. 45–48.
123. Ulacha G., Stasiński R., A System for Real-Time Lossless and Near-Lossless Video Compression, in: *Proceedings of 16th International Conference on Systems, Signals and Image Processing IWSSIP 2009*, Chalkida, Greece, 18–20 June 2009, [b.w.] 2009, s. 1–4.
124. Ulacha G., Stasiński R., A Time-Effective Lossless Coder Based on Hierarchical Contexts and Adaptive Predictors, in: *Proceedings of 14th IEEE Mediterranean Electrotechnical Conference MELECON'08*, Ajaccio, France, 5–7 May 2008, [b.w.] 2008, s. 829–834.
125. Ulacha G., Stasiński R., Context based lossless coder based on RLS predictor adaptation scheme, in: *Proceedings of International Conference on Image Processing ICIP 2009*, Egypt, Cairo, 7–11 November 2009, [b.w.] 2009, s. 1917–1920.
126. Ulacha G., Stasiński R., Effective Context Lossless Image Coding Approach Based on Adaptive Prediction, *World Academy of Science, Engineering and Technology*, September 2009, vol. 57, s. 63–68.
127. Ulacha G., Stasiński R., New simple context-based predictive technique for lossless image compression, in: *Proceedings of 13th European Signal Processing Conference EUSIPCO-07*, Poznań, Poland, September 2007, [b.m.], EURASIP 2007, s. 990–993.
128. Ulacha G., Stasiński R., On context-based predictive techniques for lossless image compression, in: *Proceedings of 12th International Workshop on Systems, Signals & Image Processing – IWSSIP 2005*, Chalkida, Greece, 2005, [b.m.], Inderscience Enterprises Ltd 2005, s. 345–348.
129. Ulacha G., Stasiński R., Parameter choice for predictor blending and application in lossless image coding, *Pomiary, Automatyka, Kontrola* 2006, no. 7bis, s. 95–97.
130. Ulacha G., Stasiński R., Performance Optimized Predictor Blending Technique For Lossless Image Coding, in: *Proceedings of The 36th International Conference on Acoustics, Speech and Signal Processing ICASSP'11*, Prague, Czech Republic, 22–27 May 2011, [b.w.] 2011, s. 1541–1544.

131. Ulacha G., Stasiński R., Predictor Blending Technique for Lossless and Near-Lossless Image Coding, in: *Proceedings of Int. Conf. on Signals and Electronic Systems (ICSES'08)*, Kraków, Poland 2008, [b.w.] 2008, s. 185–188.
132. Wang H., Zhang D., A linear model and its application in lossless image coding, *Signal Processing: Image Communication* 2004, vol. 19, s. 955–958.
133. Wang X., Wu X., On Design of Linear Minimum-Entropy Predictor, in: *Proceedings of IEEE 9th Workshop on Multimedia Signal Processing, MMSP 2007*, Crete, 1–3 October 2007, [b.w.] 2007, s. 199–202.
134. Wang Z., Bovik A. C., Sheikh H. R., Simoncelli E. P., Image Quality Assessment: From Error Visibility to Structural Similarity, *IEEE Transactions on Image Processing*, April 2004, vol. 13, no. 4, s.600–612.
135. Weinberger M.J., Seroussi G., Sapiro G., LOCO-I: A low complexity, context-based, lossless image compression algorithm. in: *Proceedings of DCC'96*, Snowbird, Utah, March–April 1996, [b.m.], The Printing House 1996, s. 140–149.
136. Weinberger M.J., Seroussi G., Sapiro G., LOCO-I: Lossless Image Compression Algorithm: Principles and Standardization into JPEG-LS, *IEEE Trans. on Image Processing*, August 2000, vol. 9, no. 8, s. 1309–1324.
137. William P.E., Hoffman M.W., Error Entropy and Mean Square Error Minimization for Lossless Image Compression, in: *Proceedings of International Conference on Image Processing ICIP'06*, Atlanta, Georgia, USA, 8–11 October 2006, [b.w.] 2006, s. 2261–2264.
138. Wu X., Lossless compression of continuous-tone images via context selection, quantization, and modeling, *IEEE Transactions on Image Processing*, May 1997, vol. 6, no. 5, s. 656–664.
139. Wu X., Bao P., L-infinity-constrained high-fidelity image compression via adaptive context modeling, *IEEE Trans. on Image Processing* 2000, vol. 9, no. 4, s. 536–542.
140. Wu X., Barthel K.U., Piecewise 2D autoregression for predictive image coding, in: *Proceedings of International Conference on Image Processing ICIP'98*, Chicago, USA, 4–7 Oct. 1998, vol. 3, [b.w.] 1998, s. 901–904.
141. 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.
142. Wu X., Memon N.D., Sayood K., A context-based, adaptive, lossless/nearly-lossless coding scheme for continuous-tone images, in: *ISO/IEC SC29/WG 1/N256*, Epernay, France, 1996.
143. Wu X., Zhai G., Yang X., Zhang W., Adaptive Sequential Prediction of Multidimensional Signals With Applications to Lossless Image Coding, *IEEE Trans. on Image Processing* 2011, vol. 20, no. 1, s. 36–42.
144. Xiang Xie, GuoLin Li, ZhiHua Wang, A Near-Lossless Image Compression Algorithm Suitable for Hardware Design in Wireless Endoscopy System, *EURASIP Journal on Advances in Signal Processing*, vol. 2007, s. 48–61.
145. Xiaohui Xue, Prediction Based On Backward Adaptive Recognition Of Local Texture Orientation And Poisson Statistical Model For Lossless/near-Lossless Image Compression, in: *Proceedings of IEEE International Conference on Acoustics, Speech, and Signal Processing*, March 1999, vol. 6, [b.w.] 1999, s. 3137–3140.
146. Yang K.H., Faryar A.F., A contex-based predictive coder for lossless and near-lossless compression of video, in: *Proceedings of International Conference on Image Processing ICIP'2000*, Vancouver, BC, Canada, 10–13 Sept. 2000, vol. 1, [b.w.] 2000, s. 144–147.
147. Ye H., *A study on lossless compression of greyscale images*, PhD thesis, Department of Electronic Engineering, La Trobe University, October 2002.
148. Ye H., Deng G., Devlin J.C., A weighted least squares method for adaptive prediction in lossless image compression, in: *Proceedings Picture Coding Symposium*, Saint-Malo, France, 2003, [b.w.] 2003, s. 489–493.
149. Ye H., Deng G., Devlin J.C., Adaptive linear prediction for lossless coding of greyscale images, in: *Proceedings of IEEE Int. Conf. on Image Processing*, [płyta CD], Vancouver, Canada, September 2000, [b.w.] 2000.

150. Ye H., Deng G., Devlin J.C., Least squares approach for lossless image coding, in: *Proceedings of the Fifth International Signal Processing and Its Applications ISSPA'99*, Queensland, Australia, 22–25 August 1999, [b.w.] 1999, s. 63–66.
151. Ye H., Deng G., Devlin J.C., Parametric Probability Models for Lossless Coding of Natural Images, in: *Proceedings of 11th European Signal Processing Conference EUSIPCO-02*, Toulouse, France, 3–6 September 2002, [b.m.], EURASIP 2002, s. 514–517.
152. Yung-Gi Wu, Differential pulse code modulation predictor design procedure using a genetic algorithm, *Optical Engineering*, June 2003, vol. 42, no. 6, s.1649–1655.
153. Zieliński T.P. *Cyfrowe przetwarzanie sygnałów. Od teorii do zastosowań*, wyd. 2, Warszawa, WKŁ 2007, ISBN 978-83-206-1640-8.

Lossless Image Compression Based on Blending Predictors
Summary

The main topic of the work is lossless data compression. Two methods of classification based on encoding and decoding time are presented, and solutions known from the literature in connection with these classifications are described.

The aim of the work was to develop a method which is capable of digital image compression with a higher average efficiency than the solutions published to date.

In chapter 2, the basic rules for designing constant and static predictive models are described, along with the problem of discrepancies between the classic minimization of mean-square error and the minimization criterion based on the entropy function in regards to prediction model design. Moreover, a number of author approaches to the selective search for prediction coefficients, including genetic algorithms, are described.

In chapter 3, basic and modified prediction methods using context switching are described, along with an original method for a division into main and additional contexts, which became the basis for developing the predictive methods described in further chapters.

In chapter 4, the rules regarding an adaptive method for removing a constant coefficient, known also as a prediction error correction associated with an appropriate context, are outlined. Using various solutions, an original method is presented, which utilizes a blend of 8 different values in order to obtain the final value of the constant coefficient for the pixel being encoded.

Chapter 5 presents methods for fast prediction coefficient adaption, describing basic methods and extended methods created by the author: LMS, ALCM, CoBALP. This chapter is finalized with a proposal for a rapid and effective method, M-LMS, which is a combination of the extended methods $ALCM_+$ and CoBALP+. In this chapter, an attempt is also made to answer the question of why the classic LMS method is characterized by low efficiency when applied to digital images.

Chapter 6 includes adaptive methods for determining prediction coefficients characterized by high implementation complexity. The RLS_+, OLS, WLS (and its variant AVE-WLS) methods are described, with particular focus on an analysis of their complexity. An original approach consisting of including the NLMS method as a second block of predictive modeling is proposed; such a solution has never before appeared in the literature on lossless digital image compression.

In chapter 7, a method for determining prediction values with neural networks is presented. Existing solutions are described and, based on these, an effective method based on a multi-layer perceptron network is proposed, while simultaneously analyzing the influence of parameters into compression effectiveness.

The second block of this work is chapter 8, in which an adaptive arithmetic encoder with context switching is described. This is treated as the second stage of the lossless image compression which takes place after data decomposition. A fully-adaptive solution with a minimal number of

initializing parameters is chosen in this original method. The encoder is designed in such a way that the sign bit and the absolute value of the prediction error are encoded separately.

Applying all the knowledge from the the first eight chapters allowed the design and discussion of the final compression method in Chapter 9, which is based on the possibility of the simultaneous usage of multiple methods to determine the expected value of the pixel. This is a prediction-blending method, whose various versions are demonstrated in the subchapters of Chapter 9. The methods referred to as to Blend-20, Blend-24 and Blend-25 meet the hypothesis from the work's aim that the bit average for a set of test images is lower than all other results known in the literature.

Chapter 10 describes the practical applications of a simplified version of the predictor blending for image and video sequence encoding. During video sequence encoding, along with spatial correlation between adjacent pixels in a single frame, there is also a temporal correlation, i.e., similarities between adjacent frames [7, 79, 98], and spectral dependences between the color components R, G and B [77]. Thus, in chapter 10 some aspects of color encoding and video sequences encoding are described. The application of an original Blend-24 method for near-lossless mode is also presented, where for bit-averages higher than 1 bit per pixel (for grayscale image) the method gives better results than the lossy codec JPEG2000.

Chapter 11 includes a conclusion together with a list of the author's original solutions described in the book. Also proposed are some potential areas of research, the results of which, according to the author, can lead to achieving further gains in efficiency in the solutions already developed for lossless and near-lossless compression of digital images.

In summary, the author's substantial contribution to the domain of lossless image compression consists of the development of a method, which, in contrast to others known from the literature, includes:

– new techniques for blending predictors, which are capable of combining many methods known from the literature;

– a novel method of contextual splitting, which may be useful for increasing the effectiveness of some known predictive methods;

– a concept previously unknown in the literature of blending a large number of cumulative error prediction correction methods;

– a solution utilizing in an efficient manner the cascade technique NLMS+;

– a contextual arithmetic encoder for compression of the sign bit of prediction error.

All the above mentioned features, only when combined with the new and numerous improved solutions known from the literature resulted in the achievement of the aim.

Verlustfreie Komprimierung von Digitalbildern mit Voraussagemischung
Zusammenfassung

Die Arbeit berührt das Thema der verlustfreien Komprimierung von Digitalbildern und stellt zwei Klassifizierungsverfahren dar, die sich am Kriterium der Kodierungs- und Dekodierungszeit orientieren. Die Klassifizierungen werden dabei anhand der in der Literatur vorhandenen Lösungen näher gebracht.

Das Ziel der Arbeit ist es, eine innovative Methode der Digitalbildkomprimierung zu erarbeiten, die bisher veröffentliche Ansätze mit ihrer Effizienz weit übertreffen würde.

Im 2. Kapitel werden grundlegende Prinzipien der Gestaltung fester und statischer Voraussagemodelle behandelt. Hierbei wird das Problem der Diskrepanz zwischen einer Lehrbuchsuggestion der Minimierung des Mittelquadratfehlers und dem Minimierungskriterium definiert, das auf der Entropiefunktion in Bezug auf Gestaltung der Voraussagemodelle basiert. Darüber hinaus werden mehrere eigenständige Versuche einer selektiven Suche nach Voraussagekoeffizienten unternommen, u. a. mit Hilfe genetischer Algorithmen.

Im Kapitel 3 werden grundlegende und modifizierte Voraussagemethoden präsentiert, in welchen die Regeln des Context-Switching befolgt werden, sowie die eigens ausgearbeitete Methode der Unterteilung in Haupt- und Ergänzungskontexte, die zur Basis für die Entwicklung der in den nächsten Kapiteln beschriebenen Voraussagemethoden wird.

Im 4. Kapitel werden die Regeln der adaptiven Methode zur Entfernung der Konstanten klargestellt. Die Methode wird auch als Korrektur des auf entsprechenden Kontext bezogenen Voraussagefehlers bezeichnet. Auf unterschiedlichen Ausführungsformen basierend, wurde eine originelle Methode entwickelt, in welcher ein Mix acht verschiedener Zwischenwerte zur Ermittlung des endgültigen Wertes der Konstanten für den jeweils kodierten Pixel benutzt wird.

Im Kapitel 5 werden die Methoden der Schnellanpassung des Voraussagekoeffizienten dargestellt. Dabei werden auch die grundlegenden und durch den Verfasser erweiterten Arbeitsmethoden LMS, ALCM, CoBALP besprochen. Das Kapitel endet mit dem Vorschlag einer schnellen und effektiven M-LMS Methode als Kopplung der Methoden $ALCM_+$ und $CoBALP_+$. Es wird auch nach der Antwort auf die Frage gesucht, warum sich die klassische LMS-Methode in Bezug auf Digitalbilder als ineffektiv erweist.

Das 6 Kapitel enthält Anpassungsmethoden der Voraussagekoeffizientermittlung, die sich durch eine hohe Implementierungskomplexität auszeichnen. Die Methoden RLS_+, OLS, WLS und deren Varianten AVE-WLS werden dargestellt, wobei sich der Autor auf der Analyse ihrer Komplexität konzentriert und einen eigenen Ansatz hervorbringt, indem er die NLMS-Methode als zweiten Block der Voraussagemodellierung einführt, was der bisherigen Literatur über verlustfreie Digitalbildkomprimierung unbekannt war.

Im Kapitel 7 wird eine Methode zur Ermittlung des Voraussagewertes dank Verwendung von Neuronennetzen präsentiert, indem vorhandene Lösungen beschrieben werden und auf dieser Basis eine effektive, auf dem vielschichtigen Perzeptronennetz stützende Methode erarbeitet wird. Dabei wird der Einfluss einzelner Netzparameter auf die jeweilige Kompressionsstufe untersucht.

Der zweite Teil der Arbeit besteht aus dem Kapitel 8, wo ein arithmetischer Anpassungskodierer mit Kontextumschaltung besprochen wird, der als zweite Etappe der verlustfreien Bildkodierung gilt und nach der Datendekompositionsetappe auftritt. In seiner Methode entscheidet sich der Autor für eine komplette Anpassungslösung mit geringster Zahl der Initialkenndaten. Der Kodierer wurde so geschaffen, dass Zeichenbits und absolute Werte der Voraussagefehler separat komprimiert werden.

Anhand der in den 8 ersten Kapiteln beschriebenen Kenntnsse wurde im Kapitel 9 (Teil III der Arbeit) die ultimative Komprimiermethode formuliert und erklärt. Die Methode basiert auf der Möglichkeit, viele Voraussagemethoden zugleich zu benutzen, um den voraussichtlichen Pixel--Endwert zu ermitteln. Es ist eine Voraussagemischungsmethode, deren diverse Varianten in den einzelnen Punkten des 9. Kapitels beschrieben werden. Die Methoden Blend-20, Blend-24 und Blend-25 erfüllen das Kriterium, welches dem Ziel der Arbeit zugrundeliegt, nämlich, dass der Bit--Durchschnitt für den Testbildsatz niedriger als sämtliche aus der Literatur bekannten Werte sein soll.

Im 10. Kapitel befindet sich der letzte Teil der Arbeit, wo praktische Anwendungen der vereinfachten Version der Voraussagemischung zur Kodierung von Bildern und Videosequenzen beschrieben wurden. Außer der gegenseitigen räumlichen Abhängigkeit der nebeneinander liegenden Pixel des kodierten Bildes besteht bei der Kodierung von Videosequenzen ebenfalls eine zeitliche Korrelation (eng. temporal correlation), d. h. eine Ähnlichkeit der nacheinander liegenden Bildersequenzen [7, 79, 98], und auch eine spektrale Abhängigkeit zwischen den Farbbestandteilen *R*, *G* und *B* [77]. Deshalb wurden im Kapitel 10 ergänzend auch die Aspekte der Kodierung von Farben und Videosequenzen beschrieben. Hier wurde ebenfalls die Verwendung einer eigenständigen Methode (Blend-24) für einen fast verlustfreien Modus dargestellt, wo bei einem Bitdurchschnitt größer als 1 Bit pro Pixel (für Bilder im grauen Farbton) die Methode zu besseren Ergebnissen im Vergleich zum Coder JPEG2000 führt.

Im 11. Kapitel befindet sich eine Zusammenfassung mit dem Verzeichnis eigener Lösungen des Autors, die in der Arbeit beschrieben werden. Es werden auch potenzielle Forschungsrichtungen aufgezeigt, durch welche sich der Autor eine weitere Steigerung der Effektivität der bisher erarbeiteten Lösungen auf dem Gebiet der verlustfreien und fast verlustfreien Digitalbildkomprimierung verspricht.

Die Bedeutung des durch den Verfasser geleisteten Beitrags zur verlustfreien Bildkomprimierung besteht in der Ausarbeitung der Methode, die sich von den anderen, in der Literatur beschriebenen wie folgt unterscheidet:

– sie verfügt über neue Techniken der Voraussagemischung, die es erlauben, viele in der Literatur beschriebenen Methoden zu koppeln,

– sie ist eine innovative Methode der Kontextaufteilung, die zur Steigerung der Effektivität mancher bekannten Voraussagemethoden geeignet ist,

– sie verbindet ein bisher in der Literatur unbekanntes Mischungskonzept vieler Methoden, die der Korrektur eines kumulierten Voraussagefehlers dienen,

– sie verfügt über eine Lösung zur effektiven Ausnutzung der Kaskade Technik NLMS+,

– sie basiert auf einem kontextabhängigen arithmetischen Coder zur Komprimierung der Voraussagefehler-Bitzeichen.

All diese Eigenschaften haben erst in Verbindung mit neu erarbeiteten und alten, aber verbesserten, aus der Literatur bekannten Lösungen ermöglicht, das Ziel zu erreichen.

ISBN 978-83-7663-156-1