Huber Drabczyk
Paweł Grochala
Adam Lewandowski

Projekt KODA

Kodek obrazów z predykcją (liniową, nieliniową)

Cel pracy

Celem pracy było zaimplementowanie i przebadanie różnych metod predykcji.

I. Zrealizowane zadania

W ramach projektu wykonaliśmy następujące prace:

1) Implementacja metod predykcji

Zaimplementowaliśmy 12 różnych metod predykcji. W tym osiem znanych modeli: (DPCM rzędu pierwszego, drugiego i trzeciego, model Grahama, DARC, ALCM, MED/MAP oraz GAP). Opracowaliśmy także cztery własne koncepcje, które zostaną omówione w dalszej części pracy. Wśród zaimplementowanych modeli są zarówno metody liniowe jak i nieliniowe.

2) Testowanie metod predykcji

Wszystkie metody predykcji przebadaliśmy pod względem uzyskanej efektywności kompresji a także czasu działania. Badania przeprowadziliśmy dla obrazów o rozmaitej kolorystyce i różnej charakterystyce. Testowaliśmy takie kategorie obrazów jak twarze, krajobrazy, zdjęcia lotnicze, obrazy regularne, zajmowaliśmy się zarówno obrazami kolorowymi jak i czarnobiałymi. Porównaliśmy wyniki uzyskane przez różne modele predykcji.

3) Testowanie innych parametrów wpływających na stopień kompresji.

Przebadaliśmy wpływ innych parametrów na efektywność kompresji w kodowaniu predykcyjnym. Są to:

- Stosowanie kodera słownikowego LZW lub kodowania Huffmana
- Wpływ RLE
- Różne sposoby przeglądania pikseli (po wierszach, kolumnach lub zygzakiem po wierszach)
- Grupowanie pikseli po kolorach lub po wierszach

Dwa ostatnie parametry dotyczą sytuacji w której stosujemy kodowanie słownikowe

Przedstawiliśmy optymalny według nas dobór parametrów, który pozwala uzyskać największy stopień kompresji.

4) Eksperymenty z modelami predykcji DARC , MED i GAP

Eksperymentowaliśmy z ustawieniem współczynników dla znanych metod predykcji. Staraliśmy się dobrać parametry, które dawałyby optymalny wynik.

II. Aplikacja

Interfejs graficzny

Program został napisany w środowisku Visual C++ jako aplikacja typu MDI, oparta na architekturze dokument – widok. W aplikacji wykorzystano biblioteki Microsoft Foundation Classes. Sterowanie działaniem odbywa się za pomocą standardowych mechanizmów spotykanych w systemie Windows – menu, przycisków na pasku narzędzi i okienek dialogowych. W celu przeprowadzenia na obrazku operacji trzeba wybrać odpowiednie pole menu lub nacisnąć skojarzony z nim przycisk na pasku narzędzi.

Jako baza aplikacji wykorzystano istniejące oprogramowanie do przetwarzania obrazów (stworzone w ramach innego projektu na uczelni). Program bazowy pozwala na skalowanie obrazu, wykonanie filtracji różnego typu (splotu przestrzennego, filtrów rankingowych, unsharp mask, filtracji logicznej) i zaznaczanie obszarów. Obsługiwane formaty plików to bitmapy (bmp) oraz własne kompresje: bazująca na redukcji kolorów, kodowanie różnicowe i kodek Huffmana oraz kompresja w dziedzinie częstotliwości. Aktualne prace zwiększają możliwości programu.

Reprezentacja danych wewnątrz programu

Całość logiki związanej z przetwarzaniem obrazu zebrana została w klasie CObrazek. Obiekt tej klasy przechowuje dane obrazka, jego metody operują na przechowywanej bitmapie i realizują wcześniej opisane operacje m.in. kodowanie i dekodowanie predykcyjne.

Obrazek trzymany jest w pamięci w postaci trzech tablic typu *unsigned char*, po jednej na każdą składową koloru: czerwoną, zieloną i niebieską. Globalnie dla obiektu przechowywana jest też informacja o wysokości i szerokości obrazka (jako *int*). Dodatkowo w obiekcie są dostępne trzy tablice robocze odpowiadające typem i rozmiarem zestawowi „głównemu" tablic. Rozwiązanie podyktowane jest tym, że praktycznie każda operacja wymaga drugiego zestawu tablic, do których wpisywane są przetworzone piksele. Stosując dwa zestawy tablic unikamy częstego alokowania dużych obszarów pamięci. Tablice „główne" i „robocze" można w prosty sposób zamienić miejscami przez zamianę wskaźników do nich (a właśnie przez wskaźniki odwołujemy się do tych tablic). Dzięki temu rozwiązaniu unikamy czasochłonnego przepisywania wynikowych wartości kolorów pikseli z tablic „roboczych" to tablic „głównych".

Zapis i odczyt obrazków z plików

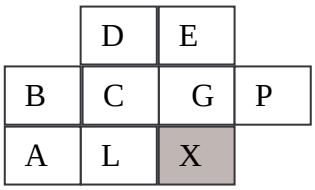
Program pozwala na zapis i odczyt plików typu **.tra, *.trb, *.trc*. Są to pliki we własnym formacie aplikacji zawierające obrazki skompresowane wcześniej opisanymi metodami kompresji. Wynikiem kompresji z użyciem metod predykcyjnych dodanych w ramach aktualnego projektu jest format **.trd*, identyczny dla wszystkich metod. Nagłówek pliku zawiera informacje o zastosowanej metodzie predykcji, przeglądania danych, użytej kompresji. Dodatkowo program pozwala na ładowanie i zapisywanie bitmap (**.bmp*).

III. ZASTOSOWANE METODY PREDYKCJI

Metody predykcyjne

W naszym projekcie zaimplementowaliśmy 8 różnych metod predykcji. Są wśród nich metody liniowe, nieliniowe a także metody statyczne jak i adaptacyjne. Poniżej przedstawiony jest szczegółowy opis tych metod.

Wszystkie metody mają różny kontekst przewidywania danych. To znaczy na podstawie różnych pikseli przewidują wartość bieżącego pola. Poniżej na rysunku przedstawiono oznaczenia, które stosujemy przy wzorach. X oznacza piksel dla którego obliczamy wartość funkcji predykcji.



1) Predykcja „poprzedni z lewej”

Jest to predykcja liniowa rzędu pierwszego. Wartości dla kolejnych pikseli przewidujemy jako wartość piksela z lewej strony.

Wzór predykcji: $X = L$

Kształt kontekstu:

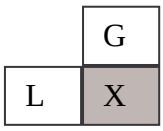
Od wartości rzeczywistej odejmujemy wartość tak przewidzianej danej i powstaje nam strumień błędów kodowych do zakodowania. Predykcje wykonujemy oddzielnie dla każdej składowej RGB.

2) Predykcja „średnia z lewej i z góry”

Jest to predykcja liniowa rzędu drugiego. Wartość piksela obliczana jest na podstawie średniej wartości piksela leżącego z lewej i powyżej piksela badanego. Dla pikseli leżących w pierwszym wierszu i pierwszej kolumnie stosujemy predykcję różnicową.

Wzór predykcji: $X = 0.5\ L + 0.5\ G$

Kształt kontekstu:



Od wartości rzeczywistej odejmujemy wartość tak przewidzianej danej i powstaje nam strumień błędów predykcyjnych do zakodowania.

3) Predykcja „liniowa rzędu trzeciego”

Jest to predykcja liniowa rzędu trzeciego. Wartość piksela jest obliczana jako kombinacja liniowa trzech pikseli: leżącego po lewej stronie, leżącego powyżej, oraz piksela leżącego po skosie lewo-góra. Podobnie jak w ostatniej metodzie pierwszy wiersz oraz kolumna kodowane są różnicowo.

Wzór predykcji: $X = 0.75\ L + 0.75\ G - 0.5\ C$

Kształt kontekstu:

C	G
L	X

4) Model Grahama

Jest to model predykcji nieliniowej. Jest to jeden z prostszych modeli predykcji przełączanej. Charakteryzuje się on tym, że w zależności od sytuacji bierzemy albo piksel leżący z lewej strony albo piksel leżący powyżej. Pierwsza kolumna oraz wiersz kodowane są różnicowo.

Wzór predykcji: $$X = \begin{cases} G, & |C - G| > |G - C| \\ L, & wpp \end{cases}$$

Kształt kontekstu:

	G
L	X

5) „Kodowanie DARC”

Jest to model predykcji nieliniowej. Jest to też model adaptacyjny, w której do wyznaczania wartości przewidywanej stosujemy metodę oszacowania gradientów ΔH oraz ΔV.

Oszacowanie gradientów: $\Delta H = |C - G|$
$\Delta V = |C - L|$

Wzór predykcji: $X = \alpha * C - (1 - \alpha) * G$,

Gdzie współczynnik α wyznaczamy następująco $$\alpha = \frac{\Delta V}{\Delta V + \Delta H}$$

Kształt kontekstu:

C	G
L	X

6) Kodowanie ALCM

ALCM jest to adaptacyjny model predykcji nieliniowej. Każdemu pikselowi wchodzącemu w skład kontekstu przypisana jest waga z jaką bierzemy dane pole do przewidywania kolejnej wartości. Zastosowana tu jest tak zwana adaptacja wstecz.
Po każdorazowej predykcji piksela te wagi są odpowiednio modyfikowane, ale skutki tej adaptacji będą dotyczyły dopiero kolejnych danych przewidywanych. Kształt kontekstu tworzą dwa piksele po lewej stronie oraz piksele leżące powyżej a także leżące po skosie w kierunku lewo – góra oraz prawo – góra. Piksele w pierwszym rzędzie i pierwszych dwóch kolumnach kodowane są różnicowo.

Wzór predykcji: $X = W_A * A + W_L * L + W_C * C + W_G * G + W_P * P$
Gdzie W_A, W_L, W_C, W_G, W_P są to wagi odpowiednich pikseli

Modyfikowanie wag:

Jeżeli X > wartość rzeczywista
- Waga piksela o najwyższej wartości zwiększana o 1
- Waga piksela o najmniejszej wartości zmniejszana o 1

Jeżeli X < wartość rzeczywista piksela
- Waga piksela o najwyższej wartości zmniejszana o 1
- Waga piksela o najmniejszej wartości zwiększana o 1

Kształt kontekstu:

	C	G	P
A	L	X	

7) Model predykcji MED/MAP

Model MED jest nieliniowym modelem adaptacyjnym. Do predykcji wartości piksela wybierana mediana z trzech możliwych wartości przewidywanych: piksela z lewej, z góry lub liniowej kombinacji pikseli z lewej, z góry i po skosie lewy-górny. Metoda ta wykrywa krawędź w najbliższym sąsiedztwie kodowanego piksela, a następnie do predykcji wykorzystywana jest wartość bardziej różniąca się od piksela po skosie lewa-górna.

Wzór predykcji:
$$X = \begin{cases} \min(L,G), & C \geq \max(L,G) \\ \max(L,G), & C \leq \min(L,G) \\ L+G-C, & wpp \end{cases}$$

Kształt kontekstu:

C	G
L	X

8) Model predykcji z gradientem GAP

Model GAP jest modelem predykcji przełączanej. W metodzie tej GAP model predykcji jest dostosowywany zgodnie z lokalnymi gradientami. W zależności od wykrytej krawędzi (ostra krawędź pozioma, krawędź pozioma, słaba krawędź pozioma, ostra krawędź pionowa, krawędź pionowa, słaba krawędź pionowa) wybierana jest wartość funkcji predykcji. Wartości progów umożliwiających wykrywanie krawędzi w najbliższym sąsiedztwie kodowanego piksela dobrano eksperymentalnie.

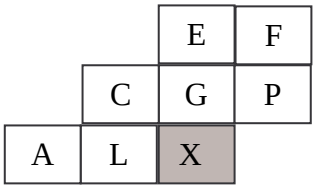
$$\nabla_v = |L - A| + |G - C| + |G - P|$$
$$\nabla_h = |L - C| + |G - E| + |P - F|$$

Wzór predykcji:

$$X = \begin{cases} L, & \nabla_v - \nabla_h > 80 \\ \lfloor (L+Z)/2 \rfloor, & 80 \geq \nabla_v - \nabla_h > 32 \\ \lfloor (L+3Z)/4 \rfloor, & 32 \geq \nabla_v - \nabla_h > 8 \\ G, & \nabla_v - \nabla_h < -80 \\ \lfloor (G+Z)/2 \rfloor, & -80 \leq \nabla_v - \nabla_h < -32 \\ \lfloor (G+3Z)/4 \rfloor, & -32 \leq \nabla_v - \nabla_h < -8 \\ Z & wpp \end{cases}$$

$$gdzie\ Z = \lfloor (L+G)/2 \rfloor + \lfloor (P+C)/2 \rfloor$$

Kształt kontekstu:



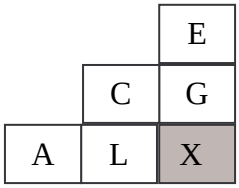
9) Model predykcji „medianowy”

Model „medianowy” został on stworzony w ramach projektu. W metodzie tej do predykcji używana jest wartość mediany spośród wartości piksela po skosie i liniowej kombinacji dwóch pikseli u góry bądź dwóch pikseli po lewej stronie. Wartości pikseli u góry i z lewej są wyrażone z wagami. Do predykcji używana jest wartość środkowa po to, aby za każdym razem popełniać jak najmniejszy błąd.

Wzór predykcji:

$$X = mediana(w_1, w_2, C)$$
$$gdzie: \quad w_1 = 0.8L + 0.2A$$
$$w_2 = 0.8G + 0.2E$$

Kształt kontekstu:



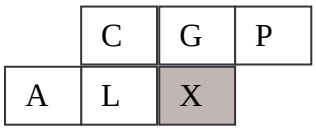
10) Model połączony (ALCM + MED + Metoda Grahama)

W metodzie tej wyliczamy wartość przewidywaną jako kombinację wartości uzyskiwanych przez ALCM, MED oraz M. Grahama. Chcemy, aby nasz model był jak najbardziej zbliżony do tej metody, która generuje najmniejszy błąd predykcyjny. W związku z tym każdej metodzie przypisujemy wagę, która jest odpowiednio modyfikowana w zależności od tego jak dobrze dana funkcja przewiduje. Jeśli błąd predykcji jest duży to waga mała, jeśli zaś błąd mały to współczynnik duży. W efekcie końcowym chcemy stworzyć model uzyskujący wyniki zbliżone do najlepszej funkcji przewidującej.

Najpierw każdej metodzie przypisujemy wagę W_{ALCM}, W_{MED} oraz W_{Graham}. Na początku są one sobie równe. Dla każdego piksela obliczamy następnie wynik predykcji dla metody ALCM: X_A, MED X_M oraz modelu Grahama X_G. Potem bierzemy każdą z powyższych wartości z odpowiednim dla danego modelu współczynnikiem i tak uzyskujemy wynik predykcji połączonej. Po każdym zakodowanym pikselu współczynniki są modyfikowane, tak, aby waga była odwrotnością modułu błędów jakie dana metoda uzyskała od początku predykcji. W związku z tym metoda, której suma dotychczasowych błędów jest najmniejsza będzie brana z największym współczynnikiem.

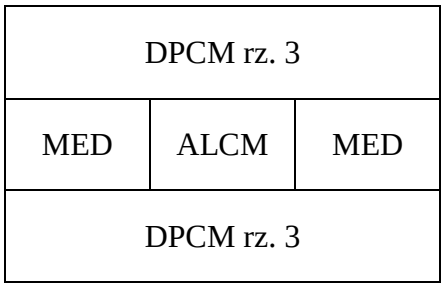
Wzór predykcji: $X = (X_A * W_{ALCM} + X_M * W_{MED} + X_G * W_{Graham}) / (W_{ALCM} + W_{MED} + W_{Graham})$

Kształt kontekstu:
(Odpowiada metodzie ALCM, gdyż ona ma największy kontekst ze stosowanych metod)



11) Kodowanie z podziałem na obszary 1

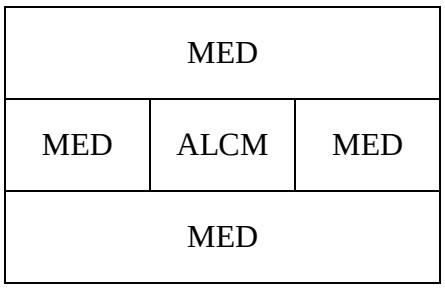
Jest to model stworzony na potrzeby projektu. Idea tej metody wzięła się z obserwacji, że zwykle najwięcej szczegółów jest zawarte w centrum obrazka, często tam też jest umieszczona twarz. Chcemy więc aby środkowa część była kodowana za pomocą metody najlepiej przewidującej. Natomiast do fragmentów o większej ilości obszarów gładkich, gdzie nie potrzebujemy tak bardzo dokładnej predykcji stosujemy nieco szybsze metody.
W tym modelu obszar jest podzielony na 9 równych części. Cały górny i dolny pasek kodowany jest metodą DPCM rzędu trzeciego. Natomiast środkowa część jest podzielona na trzy obszary. Lewa i prawa część kodowane są metodą MED, natomiast środkowa metodą dającą najlepsze wyniki, czyli ALCM. Podział zilustrowany jest na rysunku 1.
Kontekst odpowiada kontekstom metod użytych do odpowiednich fragmentów.



Rysunek 1

12) Kodowanie z podziałem na obszary 2

Jest to model podobny do poprzedniego. Główna myśl jest taka sama. Chcemy użyć jak najlepszej metody do kodowania fragmentu o największej liczbie szczegółów czyli centrum obrazka a do części leżących bliżej krawędzi, gdzie zwykle przeważają obszary gładkie użyć nieco gorszej metody.
Podobnie jak poprzednio dzielimy obrazek na dziewięć równych obszarów. Metodą najdokładniejszą czyli ALCM kodujemy sam środek obrazka. Natomiast modelu MED, który daje troszkę gorsze wyniki, ale za to jest szybszy używamy do pozostałych części. Podział jest zobrazowany na rysunku 2.
Podobnie jak to miało miejsce w poprzedniej metodzie kontekst odpowiada kontekstom metod użytych do odpowiednich fragmentów.



Rysunek 2