Kompresja transformacyjna. Opis standardu JPEG.

Algorytm JPEG powstał w wyniku prac prowadzonych przez grupę ekspertów (ang. Joint Photographic Expert Group). Prace te zakończyły się w 1991 roku, kiedy organizacja ISO (ang. International Standard Organisation) przyjęła nowy standard kompresji statycznych obrazów cyfrowych. Standard JPEG wykorzystuje transformacyjną metodę kompresji opartą o transformatę kosinusową DCT. Dla naturalnych obrazów kolorowych metoda ta pozwala na redukcję średniej ilości bitów na piksel z 24 do 0.5 przy akceptowalnej jakości, lub do 1 bitu przy dobrej jakości.

Według raportu ISO z 1988 roku algorytm JPEG składa się z następujących kroków:

1. Podzielenie obrazu na rozłączne bloki U o rozmiarze 8x8;

2. Wykonanie transformacji DCT dla każdego bloku U: $V = CUC^T$;

$$C = \left[c_{i,j}\right], \quad i, j = 0, \ldots, 7$$

$$c_{i,j} = \begin{cases} \frac{1}{\sqrt{8}}, & i = 0 \\ \sqrt{\frac{2}{8}} \cos \frac{pi(j+0.5)}{8}, & i > 0 \end{cases}$$

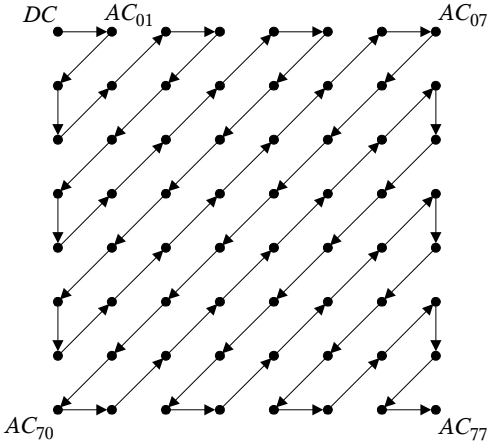
3. Skwantowanie elementów bloku V według schematu równomiernego:

$$K_{i,j} = \left\lfloor V_{i,j} / Q_{i,j} + 0.5 \right\rfloor ,$$

gdzie długość przedziału kwantyzacji jest dana w macierzy Q. Celem kwantyzacji jest dalsze zwiększenie stopnia kompresji poprzez reprezentację współczynników DCT z precyzją nie większą niż wymagana dla otrzymania rekonstruowanego obrazu o założonej jakości. Mówiąc inaczej kwantyzacja jest procesem, który eliminuje informację wizualnie nieznaczącą. Macierz kwantyzacji może być dostarczona przez użytkownika lub może być jedną z macierzy zalecanych przez JPEG (wynik eksperymentów psychowizualnych). Można w szczególności przyjąć, że wszystkie elementy tej macierzy są sobie równe. Podejście to ze względu na swoją prostotę i dobre wyniki jest zalecane w trakcie niniejszego projektu. Macierze Q używane w procesie kwantyzacji można znaleźć np. w [Ska93].

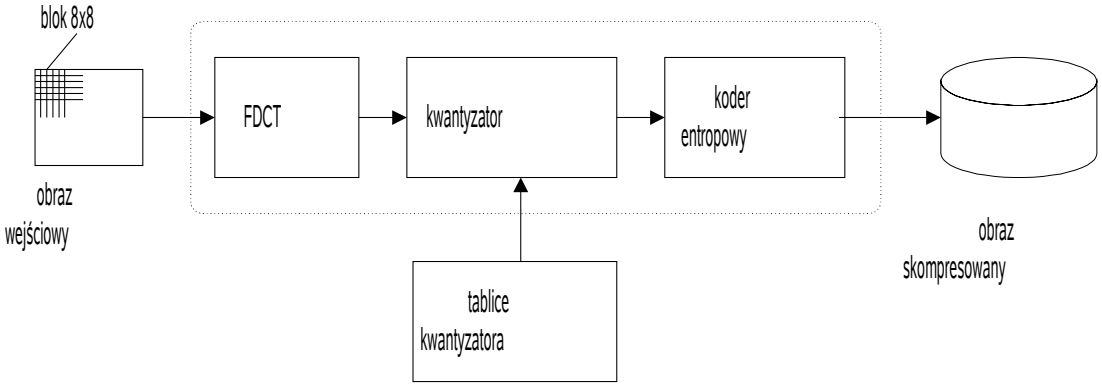
4. Współczynniki otrzymane w wyniku kwantyzacji dzielone są na dwie grupy: *DC* i *AC*. Współczynniki *DC* ($K_{0,0}$) niosą informację o średniej jasności bloku 8x8 pikseli. Pomiędzy współczynnikami *DC* z sąsiednich bloków występuje zazwyczaj silna korelacja, dlatego

kodowanie binarne sekwencji elementów *DC* wykonuje się techniką predykcji liniowej rzędu jeden. Otrzymane różnice są kodowane entropowo przy użyciu kodu Huffmana zgodnie z aktualnymi częstościami poziomów kwantyzacji występującymi w obrazie. Warto tutaj wspomnieć, że współczynniki *DC* zawierają zazwyczaj znaczną część energii całego obrazu. Ostatecznie wszystkie współczynniki po kwantyzacji ustawiane są w ciąg według uporządkowania *zygzak* (rys. 1). To uporządkowanie, poprzez umieszczenie współczynników związanych z funkcjami bazowymi o niższych częstotliwościach przed tymi o częstotliwościach większych, prowadzi do zwiększenia skuteczności kodowania entropowego.

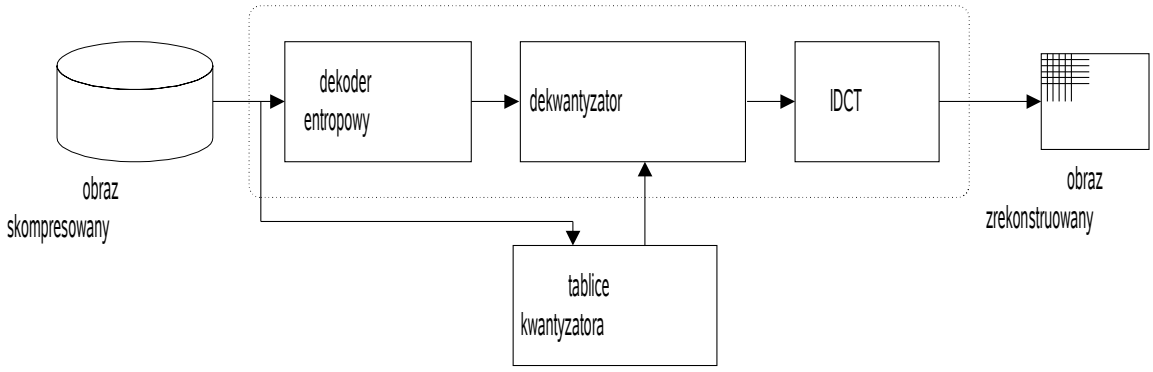


Rys 1 Schemat przeglądania typu *zygzak* poprzedzającego kodowanie entropowe.

5. Kodowanie entropowe (bezstratne) elementów *AC*, tzn. $K_{i,j}$, $i \neq 0$ lub $j \neq 0$. Proces kodowania można podzielić na dwie fazy. W fazie pierwszej ciąg współczynników uzyskany w punkcie 4 (po kwantyzacji i uporządkowaniu *zygzak*) jest dzielony na sekwencje zer zakończonych elementem niezerowym. W fazie drugiej koduje się te sekwencje przy wykorzystaniu kodowania ze słowem o zmiennej długości *VLC* (ang. variable length coding). Najpierw koduje się metodą Huffmana zintegrowane obiekty typu: długość sekwencji zer, liczba bitów w kodzie *VLC* elementu niezerowego. Następnie koduje się wartości niezerowych elementów *AC* według prostej reguły: jeśli *AC* ma wartość *k* i zapis binarny modułu liczby *k itob*(|*k*|) ma *b* cyfr, to dla *k*>0 emituje się *itob*(|*k*|), w przeciwnym razie emituje się *b* bitów otrzymanych z *itob*(|*k*|) przez negację wszystkich bitów, tzn. emituje się ~*itob*(|*k*|). Sekwencje zer dłuższe niż 15 są dzielone na porcje przy użyciu kodu Huffmana dla obiektu (15,0). Proces ten wymaga tablicy kodowej Huffmana. Tablica taka może być dostarczona przez użytkownika lub zaczerpnięta z opisu standardu JPEG.



Rys. 2a Schemat blokowy kodera JPEG



Rys. 2b Schemat blokowy dekodera JPEG

6. Umieszczenie otrzymanego ciągu bitów w pliku lub wysłanie go przez kanał transmisyjny zgodnie z formatem zdefiniowanym w dokumencie JPEG.

Na rysunkach 2a i 2b przedstawiono schemat blokowy kodera opartego o transformatę DCT działającego wg powyżej przedstawionego standardu JPEG.

Kompresja obrazów cyfrowych wg standardu <u>JPEG-like</u>

Wykonać punkty algorytmu JPEG: 1, 2, 3 ze stałą macierzą kwantyzacji tj.

$$Q_{i,j} = cr, i, j = 0, \ldots, 7$$

wykonać punkt 4, ale zamiast kodowania Huffmana wprowadzić kodowanie VLC. Wykonać punkt 5, ale bez kodowania Huffmana. Przy kodowaniu VLC będziemy zawsze przeznaczać 4 bity na zapisywanie długości rozwinięcia dwójkowego kodowanej liczby, potem zaś będzie występować jej rozwinięcie, np. liczba 7 w kodzie dwójkowym ma zapis 111, a w proponowanym tu kodzie VLC 0011 111. W przypadku liczb ujemnych zapisujemy negację ich bitów, np. liczbę -7 zapisujemy w kodzie VLC jako 0011 000

Przykład kodowania:

Dane są współczynniki dwóch kolejnych bloków (mają po 64 elementy) już po kwantyzacji:

$$\begin{bmatrix} 520 & 50 & 0 & 0 & 0 & 0 & 0 \\ 20 & 0 & 0 & 0 & 0 & 0 & \\ 0 & 0 & 0 & 0 & -12 & & \\ 0 & 0 & 0 & 0 & & & \\ 0 & 0 & 0 & & & & \\ 0 & 0 & & & & & \\ 0 & & & & & & \\ & & & & & & 0 \end{bmatrix} \begin{bmatrix} 510 & 30 & 0 & 0 & 0 & & \\ 47 & 0 & 0 & 0 & & & \\ 0 & 0 & 0 & & & & \\ 12 & 0 & & & & & \\ 0 & & & & & & \\ & & & & & & \\ & & & & & & \\ & & & & & & 0 \end{bmatrix}$$

Ustawiamy elementy tych bloków w ciąg zgodnie z uporządkowaniem *zygzak*:

blok 1: 520,50,20,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,-12,0,...0.

blok 2: 510,30,47,0,0,0,0,0,0,12,0,...0.

Zgodnie z punktem 5 tworzymy pary (liczba zer bezpośrednio poprzedzających wartość niezerową, wartość niezerowa). Para (0,0) oznacza - dalej same zera już do końca bloku. Otrzymujemy więc:

blok(1): 520, (0,50), (0,20), (15,0), (6,-12), (0,0)

blok(2): 510, (0,30), (0,47), (6,12), (0,0)

Uwaga para (15,0) oznacza ciąg 15 zer + zero t.j. ciąg 16 zer, para (6,-12) oznacza ciąg 6 zer zakończony liczbą -12.

kodujemy blok 1:

Ponieważ jest to pierwszy blok jako predykcję wartości DC przyjmujemy 0, czyli kodujemy DC=520 jako 520-0 = 520. Wartość 520 kodujemy jako ciąg binarny (w kodzie VLC) :
1010 1000001000. Pierwsze cztery bity informują, że do zakodowania wartości 520 potrzeba 10 bitów, potem 10 bitów stanowiących reprezentację binarną liczby 520.

Przy kodowaniu par AC przyjmujemy, że na **pierwszy element pary** przeznaczamy **4 bity** (jeśli występuje więcej zer niż 15 np. 22, to kodujemy parę jako dwie pary: (15,0) i (6,x)), potem zapisujemy wartość **drugiego elementu pary w kodzie VLC**.

(0,50) kodujemy jako: 0000 0110 110010
(0,20) kodujemy jako: 0000 0101 10100
(15,0) kodujemy jako: 1111 0000

(7,-12) kodujemy jako: 0111 0100 0011 //W tym przypadku pierwszy element pary - 0111: oznacza ciąg 7 zer, potem kodujemy drugi element pary liczbę -12: na zakodowanie liczby |-12| potrzebne są 4 bity stąd kod: 0100, potem zapisujemy **zanegowaną** reprezentację binarną liczby 12: 0011 = ~(1100)

AC=(0,0): 0000 0000 //do końca bloku już same zera

kodujemy blok 2:

DC=510: 510-520 = -10 kodujemy jako ciąg binarny 0100 ~(1010) = 0100 0101

pierwsze cztery bity informują, że do zakodowania wartości |-10| potrzeba 4 bity, potem 4 bitów stanowiących **zanegowaną** reprezentację binarną liczby 10. Ze względu na predykcyjne kodowanie współczynników DC jako prognozę bieżącej wartości t.j. 510 przyjmujemy wartość poprzednią czyli 520 i zapisujemy tylko różnicę.

W przypadku realizacji projektu w Matlabie przydatne funkcje to:

imgread – wczytanie obrazu z dysku do pamięci

COLORMAP – ustawienie aktywnej palety kolorów (zwracana przez funkcję imgread przy ładowaniu obrazu)

image – wyświetlenie obrazu

double – konwersja do typu zmiennoprzecinkowego, przydatna przy operacjach matematycznych wykonywanych na wczytanym obrazie (obraz jest typu uint8)