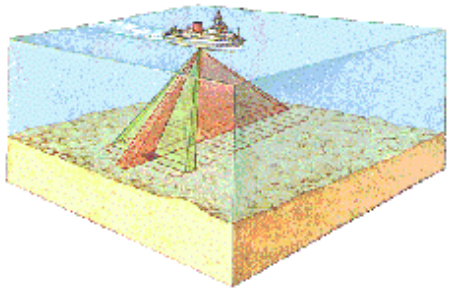
Zachodniopomorski Uniwersytet Technologiczny w Szczecinie
Wydział Informatyki

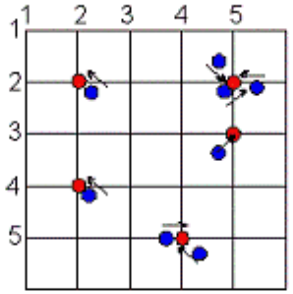
Języki analizy danych

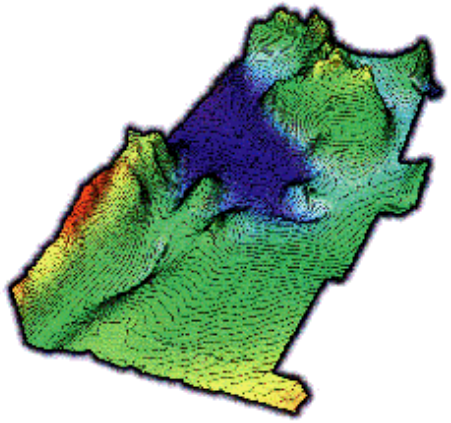
Wykład – Kompresja DTM

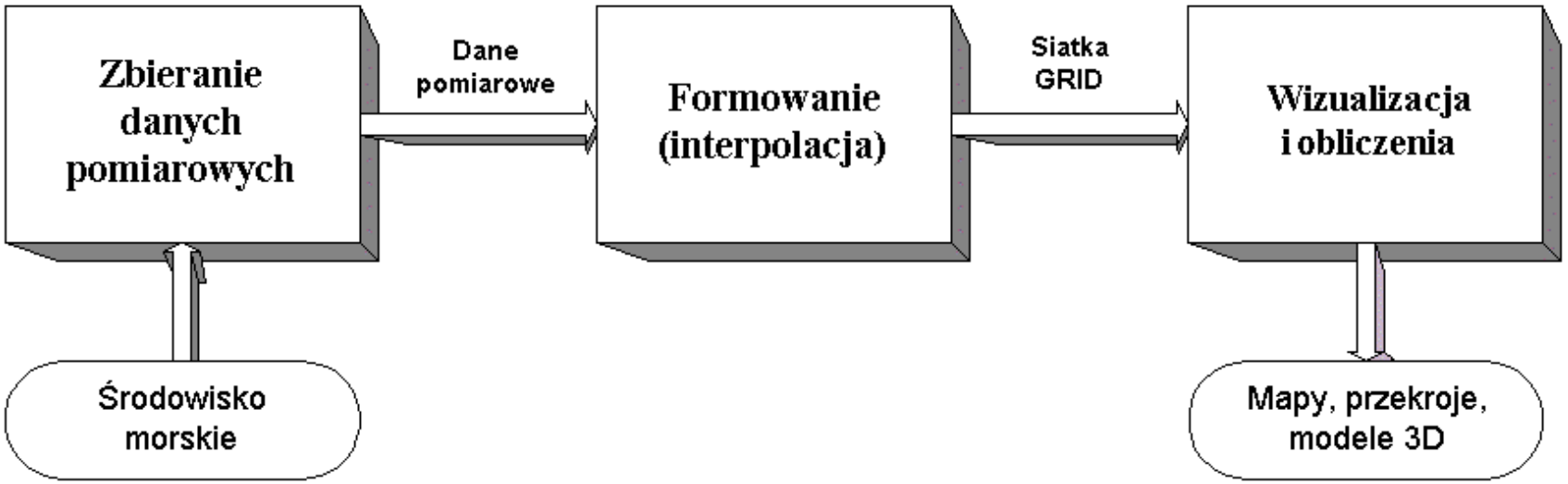
dr hab. inż. Wojciech Maleika
dr inż. Andrzej Łysko
WI ZUT, Szczecin
wmaleika@wi.zut.edu.pl
alysko@zut.edu.pl

Etapy tworzenia modelu dna morza









Grafika: opracowanie własne

Normy w pomiarach hydrograficznych

Tabela 1.2. Minimalne wymagania norm dla pomiarów hydrograficznych wg IHO [IHO98].

Parametr / Klasa	Specjalna	1	2	3
Dokładność horyzontalna (poziom ufności = 95%)	2 m	5 m+5% głębokości	20 m+5% głębokości	150 m + 5% głębokości
Dokładność głębokości dla głębokości skorygowanych (poziom ufności = 95%)	a=0.25, b=0.0075	a=0.5, b=0.013	a=1, b=0.023	a=1, b=0.023
100-procentowe przeszukiwanie dna	obowiązkowe	wymagane w wybranych obszarach	może być wymagane w wybranych obszarach	nie dotyczy
zdolność detekcyjna systemu	obiekty o objętości 1 m^3	obiekty o objętości 2 m^3 do 40 m; 10% głębokości powyżej 40 m	jak klasa 1	nie dotyczy
maksymalne odstępy pomiędzy liniami pomiaru	nie dotyczy (100 % przeszukiwanie)	3 średnie głębokości lub 25 m (zależnie od tego, co jest większe)	3-4 średnie głębokości lub 200 m (zależnie od tego, co jest większe)	4 średnie głębokości

Aby obliczyć wielkość błędu dla pomiaru głębokości, odpowiednie wartości a i b podane w tabeli 1.2 należy wprowadzić do wzoru [IHO98]:

$$e = \pm\sqrt{[a^2 + (b \cdot d)^2]} \qquad (1.1)$$

gdzie: a – stały błąd głębokości, b – wskaźnik błędu zależnego od głębokości, d – głębokość.

Normy w pomiarach hydrograficznych

Tabela 1.3. Dopuszczalny błąd głębokości dla powierzchni testowych wg norm IHO.

Powierzchnia	**Dopuszczalny błąd głębokości wg IHO [m]**
Brama torowa	0.258
Kotwicowisko	0.259
Obrotnica	0.258
Wraki	0.251

Źródła błędów przy wyznaczaniu głębokości

Przy ustalaniu dokładności głębokości modelu (dokładności pionowej) należy ocenić źródła indywidualnych błędów, a następnie je połączyć, aby otrzymać całkowity błąd propagacyjny (TPE) [IHO98]. Składowe błędów obejmują:

- błędy systemu pomiarowego i prędkości dźwięku, 5 cm
- błędy pomiaru pływów i modelowania, 2 cm
- błędy przetwarzania danych. ? cm (interpolacja 1 cm)

Kompresja JPG

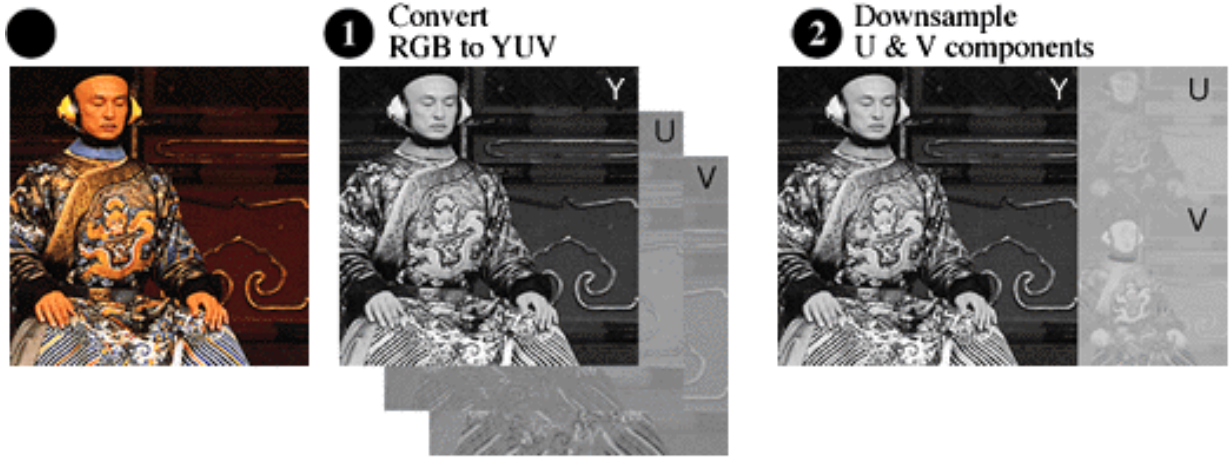
Krok 1- Konwersja do modelu luminancja – chrominancja

Krok ten wykonywany jest jedynie dla obrazów barwnych, zapisanych najczęściej jako RGB. W przypadku, gdy kompresowany jest obraz monochromatyczny, niniejszy krok jest także pomijany.

Jeśli założyć, że punkty obrazu źródłowego opisane są przy pomocy składowych RGB konwersja na model luminacja–chrominancja n.p. model YUV, realizowana jest przez przeliczenie składowych opisujących barwę punktu przy pomocy wzoru:

$$\begin{bmatrix} Y' \\ U \\ V \end{bmatrix} = \begin{bmatrix} 0.2126 & 0.7152 & 0.0722 \\ -0.09991 & -0.33609 & 0.436 \\ 0.615 & -0.55861 & -0.05639 \end{bmatrix} \begin{bmatrix} R \\ G \\ B \end{bmatrix}$$

$$\begin{bmatrix} R \\ G \\ B \end{bmatrix} = \begin{bmatrix} 1 & 0 & 1.28033 \\ 1 & -0.21482 & -0.38059 \\ 1 & 2.12798 & 0 \end{bmatrix} \begin{bmatrix} Y' \\ U \\ V \end{bmatrix}$$

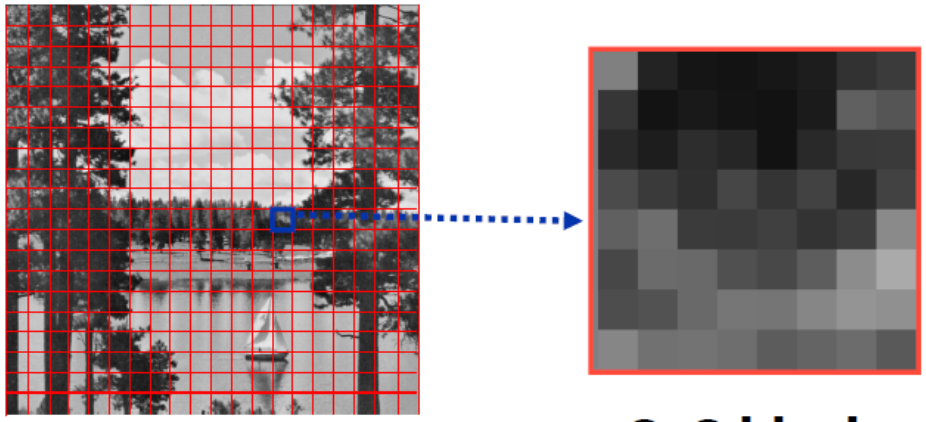


Wstępnie odrzucana jest część pikseli kanałów barwy (UV), ponieważ ludzkie oko ma znacznie niższą rozdzielczość barwy niż rozdzielczość jasności. Można nie redukować wcale, redukować 2:1 lub 4:1

Kompresja JPG

Krok 2- Podział obrazu na bloki

Na wstępie od wszystkich składowych obrazu odejmowana jest liczba całkowita równa połowie zakresu dla jej elementów. Dla przykładu, jeśli składowe YUV zapisane są na 8-bitach to zakres wynosi 256, czyli odejmowana jest liczba 128. Po wykonaniu odejmowania obraz dzielony jest na tak zwane bloki (mniejsze obrazy) o rozmiarach 8x8 punktów (bloki są tablicami). Następuje w ten sposób rozpad obrazu na części, które będą dalej przetwarzane już całkowicie niezależnie. Jeśli obraz jest monochromatyczny, blok opisywany jest jedną tablicą, w przypadku obrazu barwnego trzema tablicami zawierającymi odpowiednio składowe YUV.



Kompresja JPG

Krok 3 - Obliczenie transformaty kosinusowej dla bloków

Dla każdego bloku obliczana jest transformata kosinusowa zadana wzorami 3 i 5:

$$F(k,l)=\frac{C(k)C(l)}{4}\sum_{i=0}^{7}\sum_{j=0}^{7}f(i,j)\cos\left(\frac{2i+1}{16}k\pi\right)\cos\left(\frac{2j+1}{16}l\pi\right) \qquad (3)$$

$$f(i,j)=\frac{1}{4}\sum_{k=0}^{7}\sum_{l=0}^{7}C(k)C(l)F(k,l)\cos\left(\frac{2i+1}{16}k\pi\right)\cos\left(\frac{2j+1}{16}l\pi\right) \qquad (4)$$

$$C(k)=\begin{cases}1/\sqrt{2} & dla \quad k=0\\ 1 & dla \quad k\neq 0\end{cases} \quad i \quad C(l)=\begin{cases}1/\sqrt{2} & dla \quad l=0\\ 1 & dla \quad l\neq 0\end{cases} \qquad (5)$$

Jak widać przekształcenie polega na obliczeniu dla tablicy f o rozmiarze 8×8, innej tablicy F, o elementach F(k,l) również o rozmiarze 8×8. Wzór (4) opisuje natomiast tak zwaną transformatę odwrotną. Pozwala ona z tablicy F odtworzyć z powrotem tablicę, f, czyli blok danych opisujących obraz. Transformata odwrotna będzie wykorzystana w dekoderze.

Kompresja JPG

Krok 4 - Kwantyzacja współczynników transformaty kosinusowej

Elementy tablicy F (po DCT) są liczbami rzeczywistymi. Przekształcenie zwane kwantyzacją ma dwa zadania. Po pierwsze usunięcie z danych, informacji opisującej szczegóły obrazu praktycznie niewidoczne dla oka ludzkiego. Po drugie zastąpienie liczb rzeczywistych przybliżeniami całkowitoliczbowymi. Kwantyzacja wykonywana jest według zależności opisanych wzorami (6 i 7) dla Y, (6 i 8) dla U i V. Wyliczenie nowych tablic sprowadza się do podzielenia współczynnika transformaty kosinusowej z tablicy F przez odpowiedni element tablicy kwantyzacji Q, a następnie do zaokrągleniu wyniku do najbliższej liczby całkowitej.

$$F^{Q}(k,l) = Integer\,Round\left(\frac{F(k,l)}{Q(k,l)}\right) \qquad (6)$$

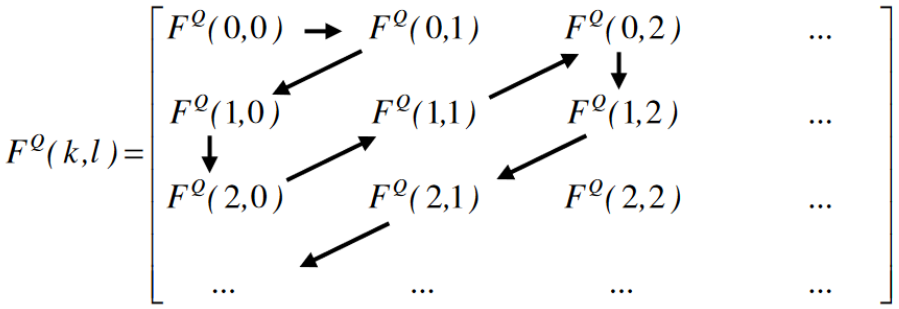
$$Q(k,l) = \begin{bmatrix} 16 & 11 & 10 & 16 & 24 & 40 & 51 & 61 \\ 12 & 12 & 14 & 19 & 26 & 58 & 60 & 56 \\ 14 & 13 & 16 & 24 & 40 & 57 & 69 & 56 \\ 14 & 17 & 22 & 29 & 51 & 87 & 80 & 62 \\ 18 & 22 & 37 & 56 & 68 & 109 & 103 & 77 \\ 24 & 35 & 55 & 64 & 81 & 104 & 113 & 92 \\ 49 & 64 & 78 & 87 & 103 & 121 & 120 & 101 \\ 72 & 92 & 95 & 98 & 112 & 100 & 103 & 99 \end{bmatrix} \qquad (7)$$

$$Q(k,l) = \begin{bmatrix} 17 & 18 & 24 & 47 & 24 & 40 & 51 & 61 \\ 18 & 21 & 26 & 66 & 26 & 58 & 60 & 56 \\ 24 & 26 & 56 & 99 & 99 & 99 & 99 & 99 \\ 47 & 99 & 99 & 99 & 99 & 99 & 99 & 99 \\ 99 & 99 & 99 & 99 & 99 & 99 & 99 & 99 \\ 99 & 99 & 99 & 99 & 99 & 99 & 99 & 99 \\ 99 & 99 & 99 & 99 & 99 & 99 & 99 & 99 \\ 99 & 99 & 99 & 99 & 99 & 99 & 99 & 99 \end{bmatrix} \qquad (8)$$

Kompresja JPG

Krok 4 - Zamiana tablicy współczynników na wektor

Następnym krokiem algorytmu jest zamiana tablicy liczb całkowitych FQ = FQ(k, l) na wektor. Zamiana tablicy na wektor polega na zapisaniu elementów tablicy FQ w odpowiedniej kolejności określonej przez tak zwany algorytm zig-zag. Schemat wyjaśniający ideę algorytmu zig-zag pokazano poniżej.



(9)

$$F^Q = [DC \quad AC_1 \quad AC_2 \quad AC_3 \quad \dots \quad AC_{63}] \tag{10}$$

Dla wygody dalszego opisu dogodnie jest wprowadzić konwencje zapisu elementów tablicy FQ jak w (10). Pierwszy element wektora współczynników DC = FQ(0, 0) nosi nazwę składowej stałej, pozostałe są składowymi zmiennymi.

Przekształcenie tablicy w wektor według schematu zig-zag ma sens, dlatego że zazwyczaj duża liczba zer znajdująca się w prawym dolnym rogu tablicy, zostanie przekształcona po zapisie w postaci wektora w długi ciąg zer. Taki ciąg jest obiektem stosunkowo prostym do oszczędnego zakodowania

Kompresja JPG

Krok 4 - Kodowanie entropijne wektora współczynników

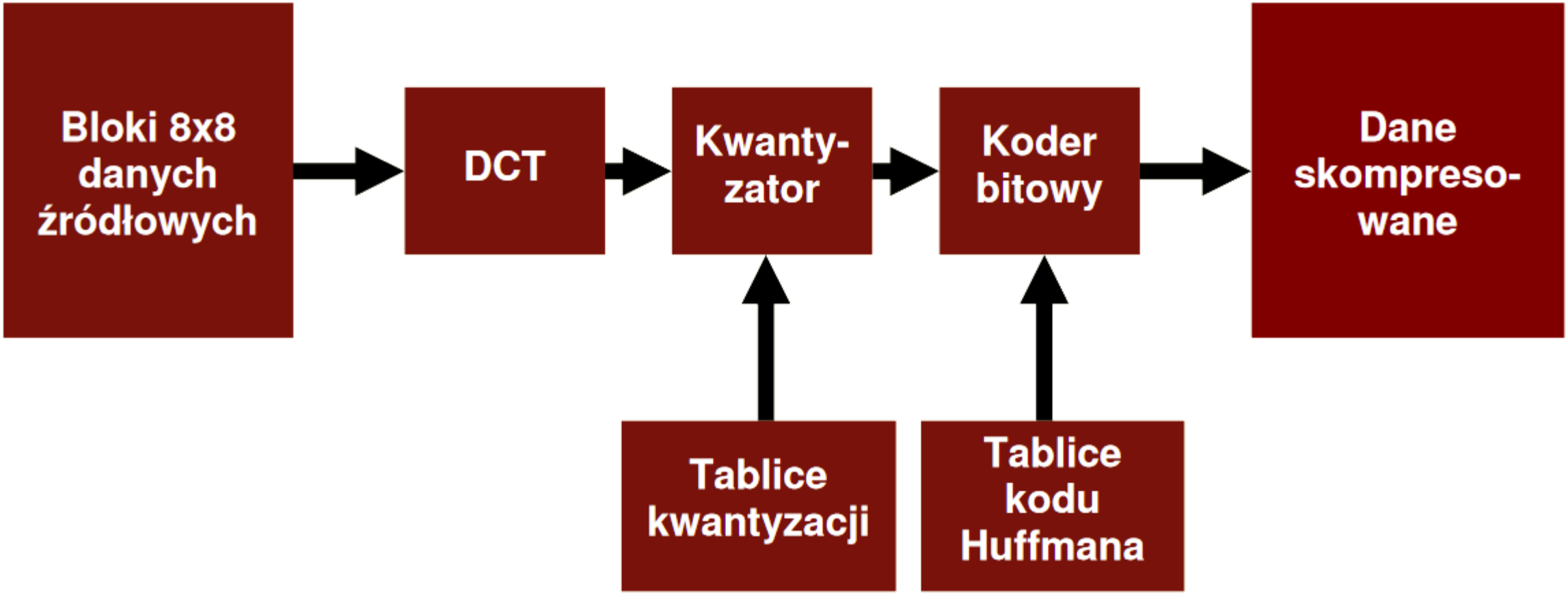
Ostatnim krokiem algorytmu kompresji jest kodowanie entropijne wektorów FQ zapisanych w postaci (10). Idea kodowania entropijnego polega na tym, że długość słowa kodowego służącego do zapisania kodowanego elementu jest powiązania z prawdopodobieństwem pojawienia się tego elementu (czyli kodowanie Huffmanna). Żeby zminimalizować średnią długość ciągu kodowego (ciągu słów kodowych opisujących kolejne elementy), te elementy, które występują częściej kodowane są przy pomocy słów krótszych niż słowa kodowe przypisywane elementom występującym rzadziej. Kodowanie entropijne wektorów FQ wykonywane jest osobno dla składowych stałych DCi składowych zmiennych ACi.

Tab. 1. Tablica kodowania dla składowej stałej

Wartość Δ_k	*size*	Kod Huffmana dla *size*	Bity dodatkowe
0	0	00	-
-1, 1	1	010	0,1
-3,-2, 2, 3	2	011	00,01,10,11
-7,…,-4,4,…7	3	100	000,…,011,100,…,111
-15,…,-8,8,…,15	4	101	0000,…,0111,1000,…,1111
…	…	…	…
-2047,…-1024,1024,…,2047	11	1 1111 1110	000 0000 0000,…,111 1111 1111

(*runlength, size*)	kod Huffmana	(*runlength, size*)	kod Huffmana
(0, 1)	00	(0, 6)	1111000
(0, 2)	01	(1, 3)	1111001
(0, 3)	100	(5, 1)	1111010
EOB	1010	(6, 1)	1111011
(0, 4)	1011	(0, 7)	11111000
(1, 1)	1100	(2, 2)	11111001
(0, 5)	11010	(7, 1)	11111010
(1, 2)	11011	(1, 4)	111110110
(2, 1)	11100	…	…
(3, 1)	111010	ZRL	11111111001
(4, 1)	111011	…	…

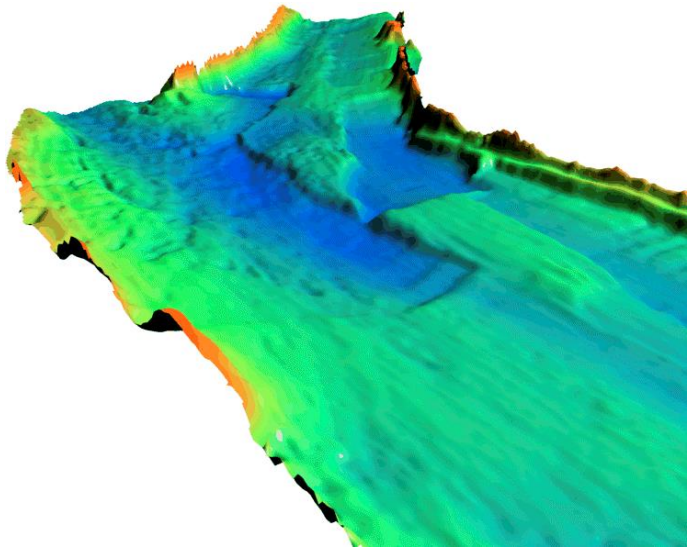
Kompresja JPG

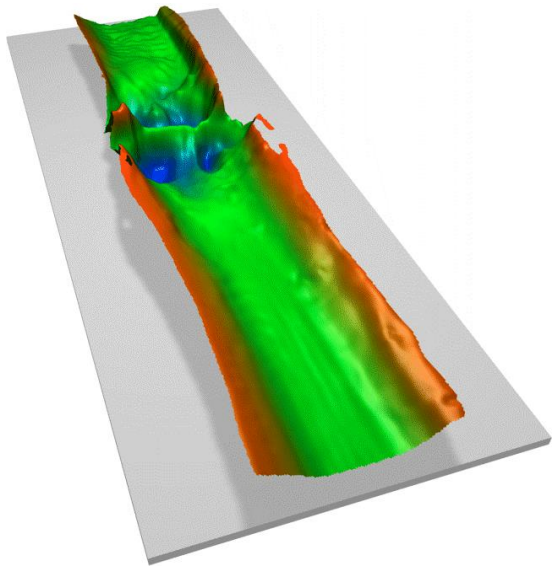


Kompresja DCT DTM

Krok 0 – Wymagania:

- ✓ Jedna macierz.
- ✓ Liczby rzeczywiste a nie całkowite.
- ✓ Wymagana zadana dokładność rekonstrukcji, wartość określona przez operatora, np. 5 cm.
- ✓ Nie wszystkie wartości muszą być liczbą, mogą występować *NaN*, powierzchnia może mieć nieregularny kształt. Problem należy rozwiązać, ale nie będzie przedmiotem naszych zajęć.





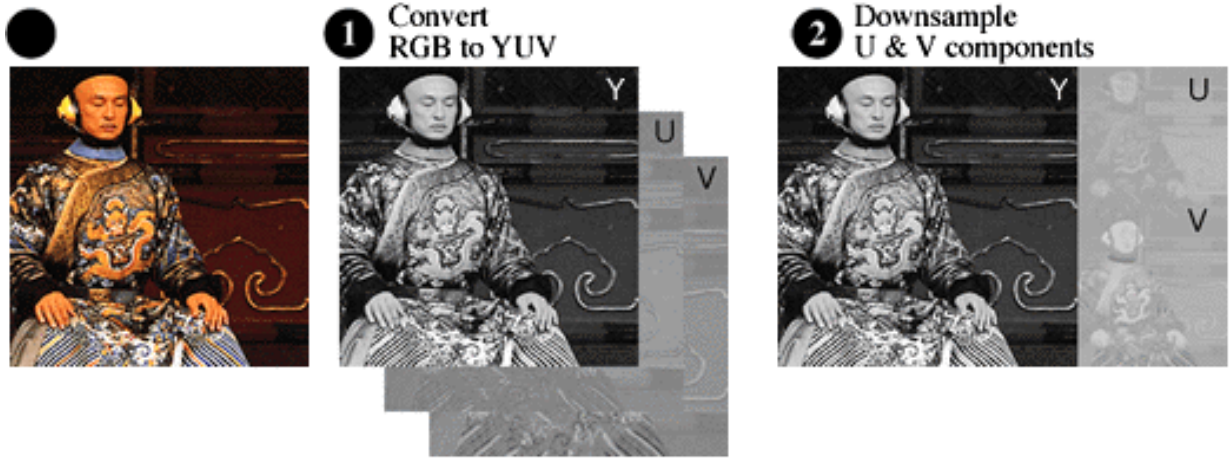
Kompresja DCT DTM

Krok 1- Konwersja do modelu luminancja – chrominancja?

- ✓ Nie, bo nie mamy danych RGB czy YUV a jedną tablicę
- ✓ Nasze dane nie są całkowite a rzeczywiste
- ✓ Aby zachować wysoką dokładność nie dokonujemy redukcji rozmiaru tablicy
- ✓ Ten etap właściwie nie występuje

$$\begin{bmatrix} Y' \\ U \\ V \end{bmatrix} = \begin{bmatrix} 0.2126 & 0.7152 & 0.0722 \\ -0.09991 & -0.33609 & 0.436 \\ 0.615 & -0.55861 & -0.05639 \end{bmatrix} \begin{bmatrix} R \\ G \\ B \end{bmatrix}$$

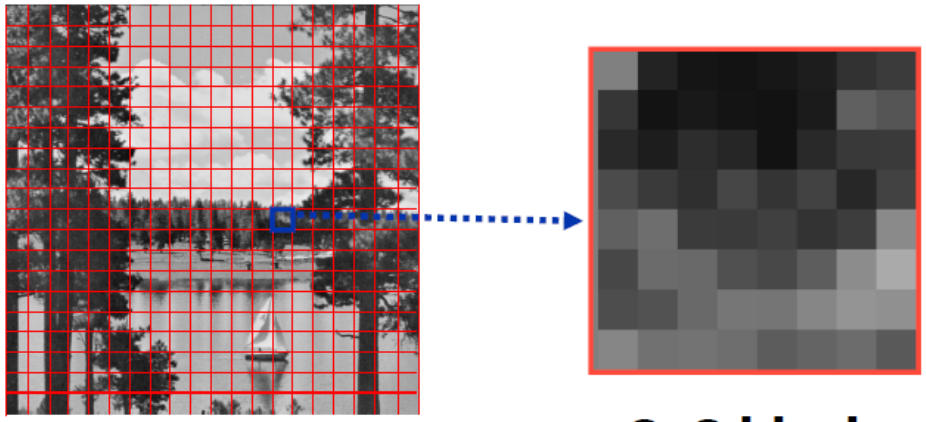
$$\begin{bmatrix} R \\ G \\ B \end{bmatrix} = \begin{bmatrix} 1 & 0 & 1.28033 \\ 1 & -0.21482 & -0.38059 \\ 1 & 2.12798 & 0 \end{bmatrix} \begin{bmatrix} Y' \\ U \\ V \end{bmatrix}$$



Kompresja DCT DTM

Krok 2- Podział obrazu na bloki?

- ✓ Tak, bo taka jest ogólna zasada w kompresji z użyciem DCT
- ✓ Czy bloki 8x8 są właściwe? – jakie częstotliwości (szybkości zmian) dominują?
- ✓ Może warto przetestować jaki rozmiar bloku byłby optymalny?



Kompresja DCT DTM

Krok 3 - Obliczenie transformaty kosinusowej dla bloków?

- ✓ Oczywiście, to przecież podstawa tej metody kompresji
- ✓ Wzory są uniwersalne i działają i dla całkowitych i dla rzeczywistych danych
- ✓ Transformata kosinusowa dwuwymiarowa – DCT i IDCT (python)

$$F(k,l)=\frac{C(k)C(l)}{4}\sum_{i=0}^{7}\sum_{j=0}^{7}f(i,j)\cos\left(\frac{2i+1}{16}k\pi\right)\cos\left(\frac{2j+1}{16}l\pi\right) \qquad (3)$$

$$f(i,j)=\frac{1}{4}\sum_{k=0}^{7}\sum_{l=0}^{7}C(k)C(l)F(k,l)\cos\left(\frac{2i+1}{16}k\pi\right)\cos\left(\frac{2j+1}{16}l\pi\right) \qquad (4)$$

$$C(k)=\begin{cases}1/\sqrt{2} & dla \quad k=0\\ 1 & dla \quad k\neq 0\end{cases} \quad i \quad C(l)=\begin{cases}1/\sqrt{2} & dla \quad l=0\\ 1 & dla \quad l\neq 0\end{cases} \qquad (5)$$

Kompresja DCT DTM

Krok 4 - Kwantyzacja współczynników transformaty kosinusowej?

- ✓ Trudno powiedzieć czy ją zastosować
- ✓ Obniża dokładność, ale może nie aż tak mocno
- ✓ Znacznie zwiększa kompresję
- ✓ Czy stosować kwantyzację? Należy to sprawdzić.

$$F^{Q}(k,l) = Integer\,Round\left(\frac{F(k,l)}{Q(k,l)}\right) \qquad (6)$$

$$Q(k,l) = \begin{bmatrix} 16 & 11 & 10 & 16 & 24 & 40 & 51 & 61 \\ 12 & 12 & 14 & 19 & 26 & 58 & 60 & 56 \\ 14 & 13 & 16 & 24 & 40 & 57 & 69 & 56 \\ 14 & 17 & 22 & 29 & 51 & 87 & 80 & 62 \\ 18 & 22 & 37 & 56 & 68 & 109 & 103 & 77 \\ 24 & 35 & 55 & 64 & 81 & 104 & 113 & 92 \\ 49 & 64 & 78 & 87 & 103 & 121 & 120 & 101 \\ 72 & 92 & 95 & 98 & 112 & 100 & 103 & 99 \end{bmatrix} \qquad (7)$$

$$Q(k,l) = \begin{bmatrix} 17 & 18 & 24 & 47 & 24 & 40 & 51 & 61 \\ 18 & 21 & 26 & 66 & 26 & 58 & 60 & 56 \\ 24 & 26 & 56 & 99 & 99 & 99 & 99 & 99 \\ 47 & 99 & 99 & 99 & 99 & 99 & 99 & 99 \\ 99 & 99 & 99 & 99 & 99 & 99 & 99 & 99 \\ 99 & 99 & 99 & 99 & 99 & 99 & 99 & 99 \\ 99 & 99 & 99 & 99 & 99 & 99 & 99 & 99 \\ 99 & 99 & 99 & 99 & 99 & 99 & 99 & 99 \end{bmatrix} \qquad (8)$$

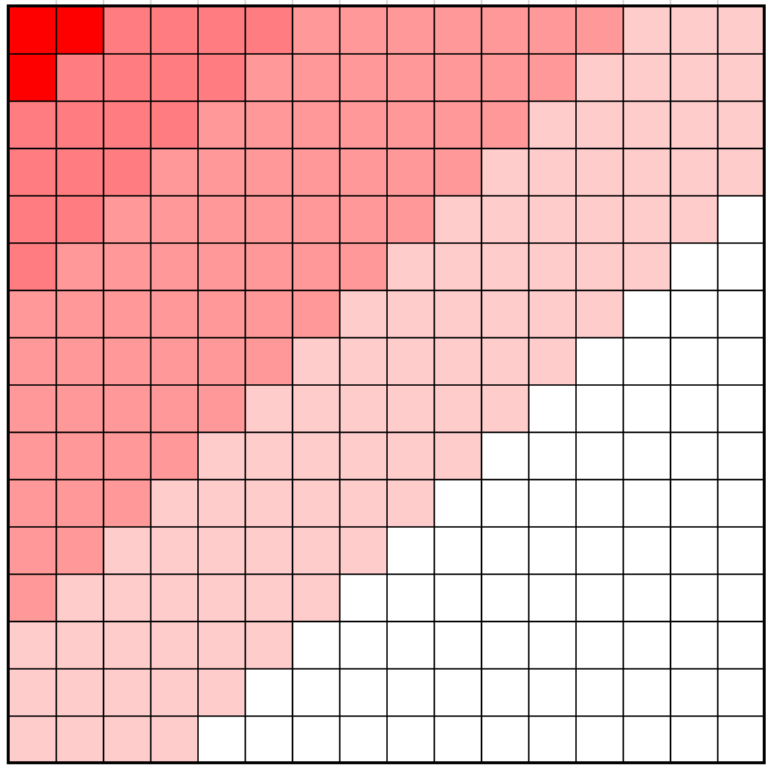
Kompresja DCT DTM

Krok 4 - Zamiana tablicy współczynników na wektor?

- ✓ Raczej tak, dzięki temu uporządkujemy dane od najważniejszych do mniej istotnych.
- ✓ Czy można zastosować inną metodę – może zapis tylko fragmentu trójkąta?

$$F^Q(k,l) = \begin{bmatrix} F^Q(0,0) \rightarrow F^Q(0,1) & F^Q(0,2) & \dots \\ \swarrow \quad \nearrow & \quad \downarrow & \\ F^Q(1,0) \quad F^Q(1,1) & F^Q(1,2) & \dots \\ \downarrow \quad \nearrow & \quad \swarrow & \\ F^Q(2,0) \quad F^Q(2,1) & F^Q(2,2) & \dots \\ \swarrow & & \\ \dots \quad \dots & \dots & \dots \end{bmatrix} \tag{9}$$

$$F^Q = [DC \quad AC_1 \quad AC_2 \quad AC_3 \quad \dots \quad AC_{63}] \tag{10}$$



Kompresja DCT DTM?

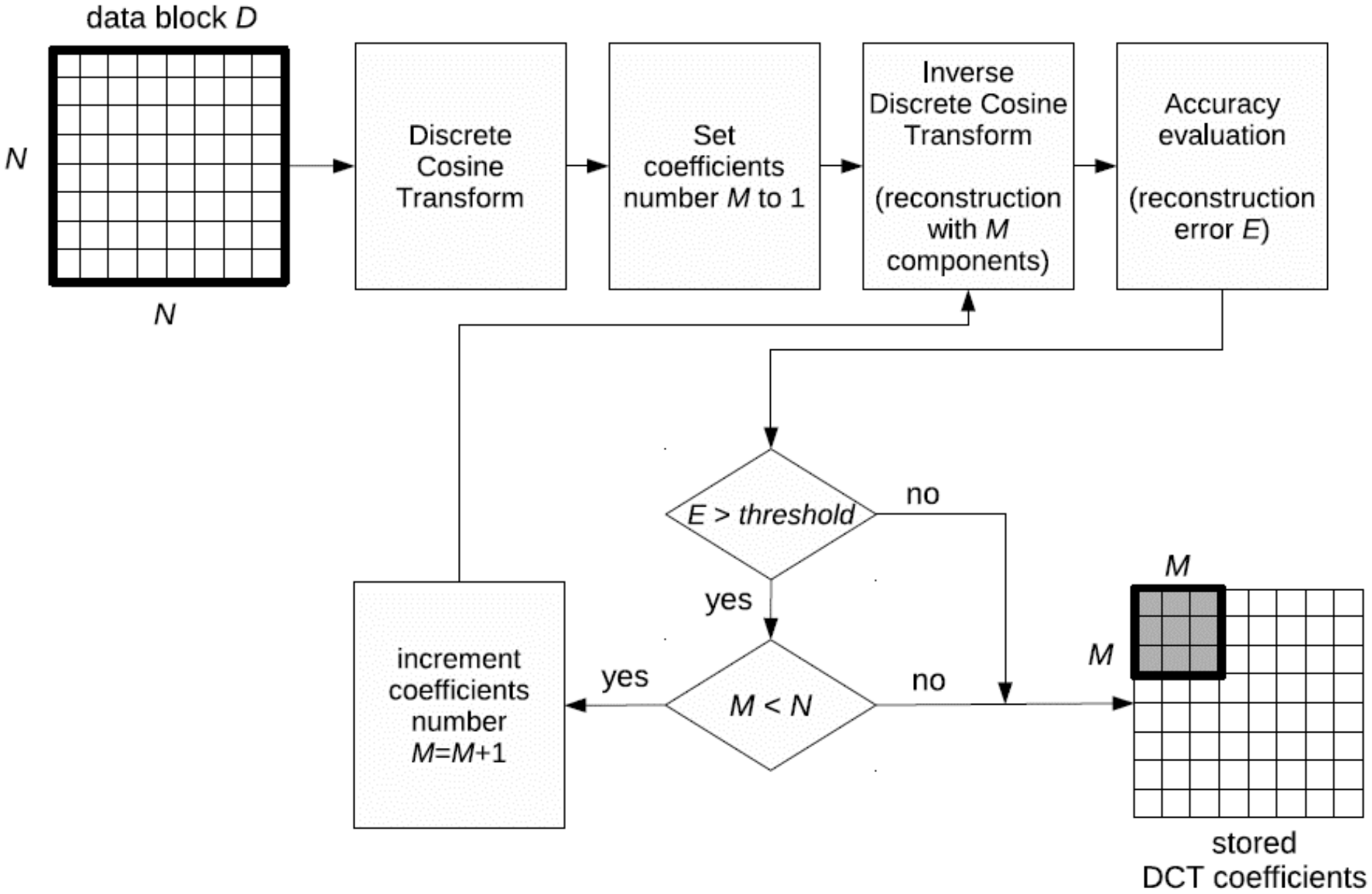
Krok 4 - Kodowanie entropijne wektora współczynników ?

- ✓ Czy można je zastosować?
- ✓ Czy przyniesie tam podobne efekty / kompresję?
- ✓ Czy możemy zastosować coś innego? (LZW, RAR)

Tab. 1. Tablica kodowania dla składowej stałej

Wartość Δ_k	*size*	Kod Huffmana dla *size*	Bity dodatkowe
0	0	00	-
-1, 1	1	010	0,1
-3,-2, 2, 3	2	011	00,01,10,11
-7,…,-4,4,…7	3	100	000,…,011,100,…,111
-15,…,-8,8,…,15	4	101	0000,…,0111,1000,…,1111
…	…	…	…
-2047,…-1024,1024,…,2047	11	1 1111 1110	000 0000 0000,…,111 1111 1111

(*runlength, size*)	kod Huffmana	(*runlength, size*)	kod Huffmana
(0, 1)	00	(0, 6)	1111000
(0, 2)	01	(1, 3)	1111001
(0, 3)	100	(5, 1)	1111010
EOB	1010	(6, 1)	1111011
(0, 4)	1011	(0, 7)	11111000
(1, 1)	1100	(2, 2)	11111001
(0, 5)	11010	(7, 1)	11111010
(1, 2)	11011	(1, 4)	111110110
(2, 1)	11100	…	…
(3, 1)	111010	ZRL	11111111001
(4, 1)	111011	…	…


data block D
N
N
Discrete Cosine Transform
Set coefficients number M to 1
Inverse Discrete Cosine Transform
(reconstruction with M components)
Accuracy evaluation
(reconstruction error E)
E > threshold
no
yes
increment coefficients number M=M+1
yes
M < N
no
M
M
stored DCT coefficients

THE END