Poradnik

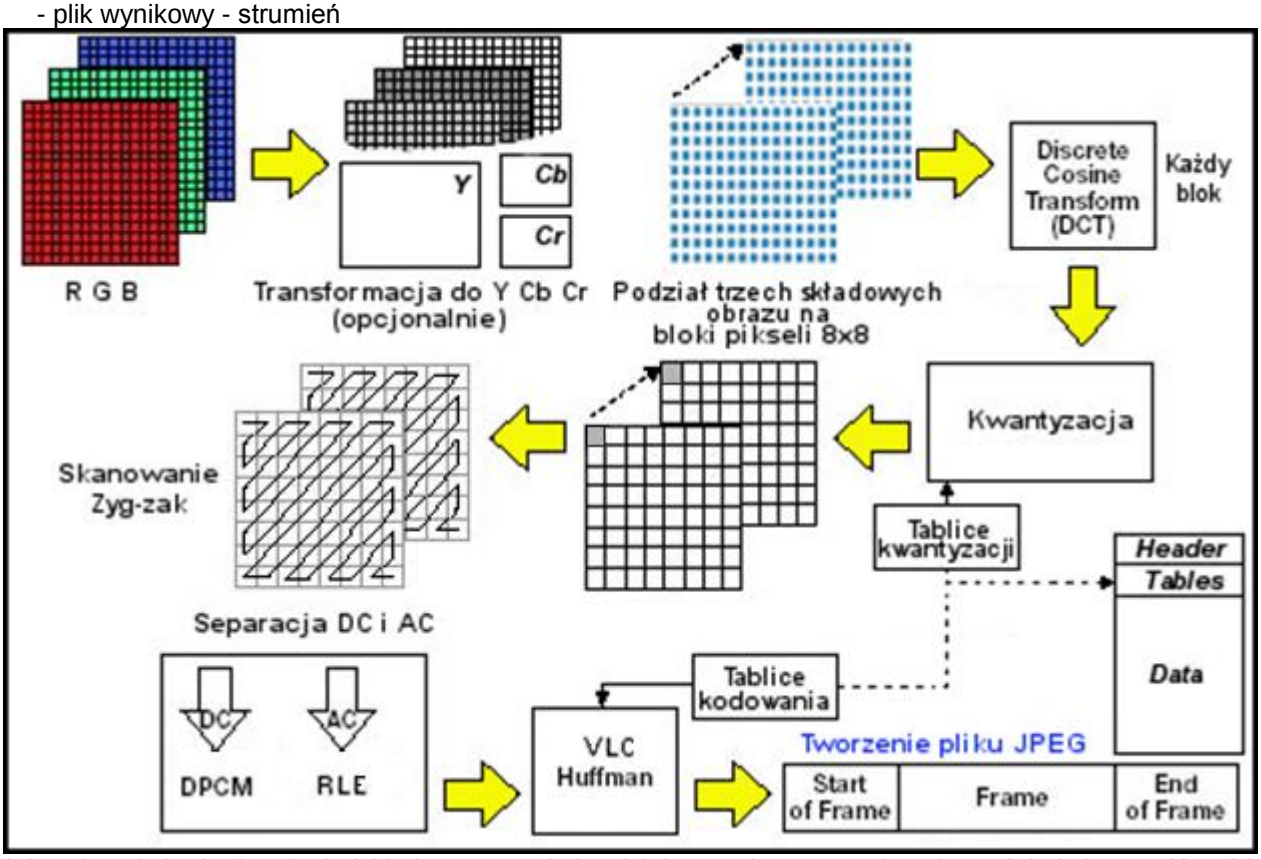
Kompresja JPEG

10-07-2009r

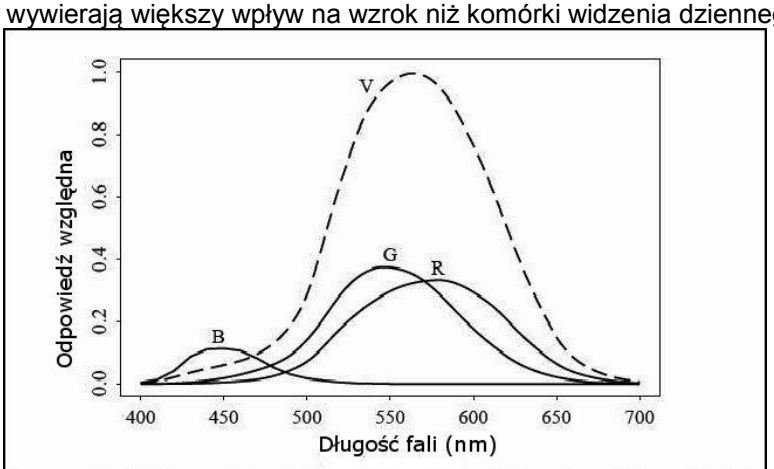
Najłatwiej zrozumieć kompresję JPG, jeśli wyjaśni się ją krok po kroku tak jak robi to enkoder (*kroki dekodera są odwrotne do kroków enkodera*)

Schemat **sekwencyjnej kompresji bazowej JPEG** przebiega w/g następujących kroków:

- Dane wejściowe są w postaci RGB24,
- Transformacja przestrzeni barw na komponenty **YCbCr** (YUV),
- Proces dyskretnej transformacji kosinusowej,
- Kwantyzacja współczynników transformaty
- Kodowanie długości ciągów (*run-length encoding* - ***RLE***)
- Kodowanie algorytmem VLC Huffmana (2DC i 2AC Tablice) [dla **Baseline Optimized** (4DC i 4AC Tablice)]
- plik wynikowy - strumień



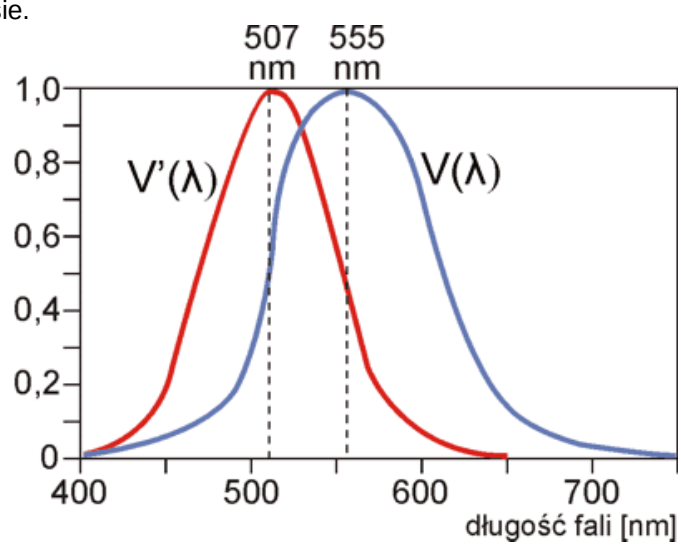
Jak wykazały badania, oko ludzkie jest znacznie bardziej wyczulone na zmianę jasności niż barwy. Komórki widzenia czarno-białego wywierają większy wpływ na wzrok niż komórki widzenia dziennego (kolorowego).



Krzywe R, G i B przedstawiają wrażliwość pręcików oka od długości fali.
Krzywa, V przedstawia odpowiedź oka na zmiany jasności.
Wartość V jest reprezentowana jako V(λ) ~ 2R + G + B / 20.
Rys obrazuje czytelnie, że Oko człowieka jest bardziej wrażliwe na zmiany jasności światła niż zmiany koloru, co jest podstawą podpróbkowania kolorów.

Dlatego standard JPEG bierze pod uwagę ten fakt, że ludzkie oko jest bardziej wrażliwe na jaskrawość koloru niż na jego odcień. Jeśli będziemy przekształcić dane z 3 kanałów RBG tak, aby jasność była przechowywana w jednej składowej, a kolory w pozostałych dwóch, to okaże się, że możemy usunąć większość informacji o kolorze, bez zauważalnego pogorszenia wyglądu obrazu.

Poszczególne osoby posiadają odmienny poziom wrażliwości nie tylko na kolor światła, ale również na jego intensywność. Sposób odbierania światła przez każdego człowieka jest zbliżony w charakterze do przedstawionego na wykresie.



Krzywa czułości względnej oka ludzkiego dla widzenia fotopowego (dziennego) V(λ) [czopki, słaba czułość na kontrast, widzenie barwne] i skotopowego (nocnego) V' (λ) [pręciki, duża czułość na kontrast, brak widzenia barw] według CIE [Widzenie mezopowe (zmierzchowe)]

Progowy model widzenia - definiuje granicę pomiędzy widzialnym i niewidzialnym.

Zależność najmniejszej dostrzegalnej różnicy luminancji od luminancji adaptacji

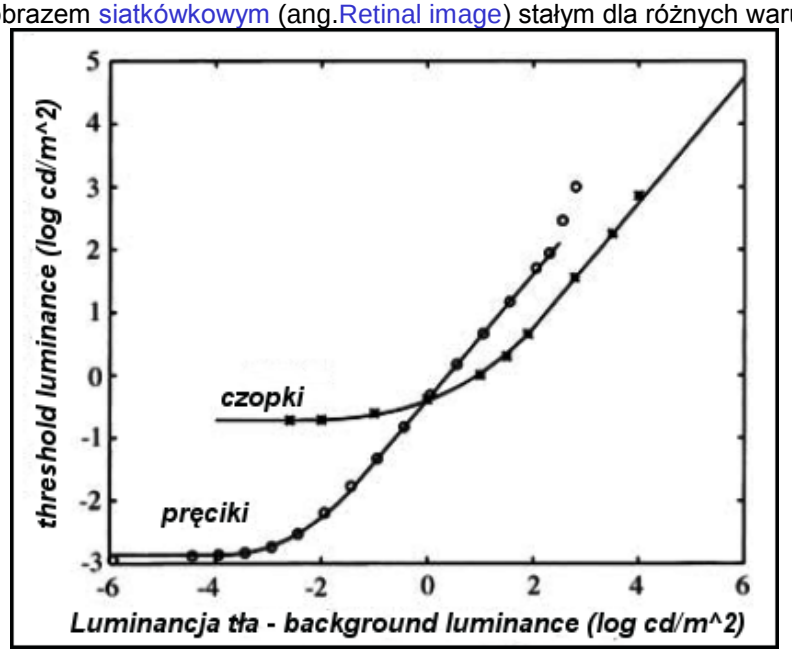
$$\Delta L = k^*L$$

Prawo Weber'a - dla dużych zakresów luminancji adaptacji zależność luminancji adaptacji od najmniejszego rozróżnialnego kontrastu jest funkcją liniową (w skali logarytmicznej).

Definicja kontrastu według Webera:

$$contrast = \Delta L / L$$

Obraz kontrastów jest obrazem siatkówkowym (ang.Retinal image) stałym dla różnych warunków oświetlenia.



Ze wzrostem luminancji tła (adaptacji) maleje czułość ludzkiego oka na kontrast (różnice luminancji).

Zależności są prawdziwe dla prostych scen (scen nie zawierających wielu częstotliwości).

Charakterystyka oka jest nieliniowa, o przebiegu czułości zbliżonym do logarytmicznego (znormalizowana czułość oka ludzkiego). Mechanizm luminancji uczestniczy w zadaniach związanych z wysoką częstotliwością przestrzenną (np. trudno odczytać żółty tekst na białym tle, gdyż różnica w luminancji jest wówczas bardzo mała. Wysoka rozdzielczość przestrzenna zależy od luminancji i jest niezależna od barwy.) ma szersze pasmo (w kodowaniu obrazów kolorowych należy poświęcić większe pasmo na kodowanie luminancji, aby osiągnąć wyższy stosunek komresji nie tracąc na jakości obrazu) mechanizm widzenia kolorów jest bardziej wrażliwy na częstotliwości przestrzenne (małe punkty „tracą" kolor i wydają się być achromatyczne, kolory na większych powierzchniach wydają się być bardziej nasycone i intensywne na większych powierzchniach

Transformacja przestrzeni barw na komponenty YCbCr (YUV)

W transformacji przestrzeni barw następuje dekorelacja składowych, które są kodowane niezależnie od siebie, umożliwia to kodowania różnych składowych z różną jakością (kompresją).

Transformacja ta rozkłada obraz w kolorach **RGB** na składową luminancyjną (popr. **Luma**) - jasność (**Y**) i dwie składowe różnicowe chrominancyjne [*zabarwienia* - antyniebieska niebieski / żółty **Cb** (**C**hroma**b**lue)**–** i składowa antyczerwona czerwony / zielony **Cr** (**C**hroma**r**ed) - wartości odpowiadające za nasycenie barwy, oznaczają odpowiednio jak bardzo niebieski i czerwony jest dany kolor - wartości te wskazują, jak wiele jest w tym kolorze, odpowiednio niebieskich i czerwonych]. Komponenty obrazu możemy uważać za warstwy, na które dzielimy obraz, by uprościć późniejszy proces kodowania, a które po zdekodowaniu i złożeniu dadzą pierwotny obraz. (terminologia w artykule: http://www.poynton.com/PDFs/YUV_and_luminance_harmful.pdf)

Założeniem dla poprawnej transformacji jest, aby wszystkie składowe barw były liczbami nieoznaczonymi całkowitymi o długości minimum ośmiu bitów (typ *byte* lub *unsigned char* ***Typ znakowy*** w praktyce powinno być używane tam, gdzie operujemy na bajtach i oczekujemy wartości {0, 1, 2, ..., 255}).

Składowe oblicza się według wzorów: http://www.jpeg.org/public/jfif.pdf http://en.wikipedia.org/wiki/YUV

JFIF (JPEG File Interchange Format) -Y'CbCr (601) from "digital 8-bit R'G'B'

$$Y = 0{,}299\ R + 0{,}587\ G + 0{,}114\ B$$
$$Cb = -0{,}1687\ R - 0{,}3313\ G + 0{,}5\ B + 128$$
$$Cr = 0{,}5\ R - 0{,}4187\ G - 0{,}0813\ B + 128$$

Przeliczenie odwrotne (przeprowadza dekoder, zawarty np. w przeglądarce obrazu lub w GIMP-ie) realizowane jest za pomocą wzorów:

$$R = Y + 1.402\ (C_r\text{-}128)$$
$$G = Y - 0.34414\ (C_b\text{-}128) - 0.71414\ (C_r\text{-}128)$$
$$B = Y + 1.772\ (C_b\text{-}128)$$

Podane wyżej przekształcenia są nieodwracalne (ze względu na zmiennoprzecinkową reprezentację składowych). Poniżej przedstawiono odwracalną wersję przekształcenia RGB na YCbCr:

$$Y = [0{,}25\ R + 0{,}5\ G + 0{,}25\ B]$$
$$Cb = (B - G)$$
$$Cr = (R - G)$$

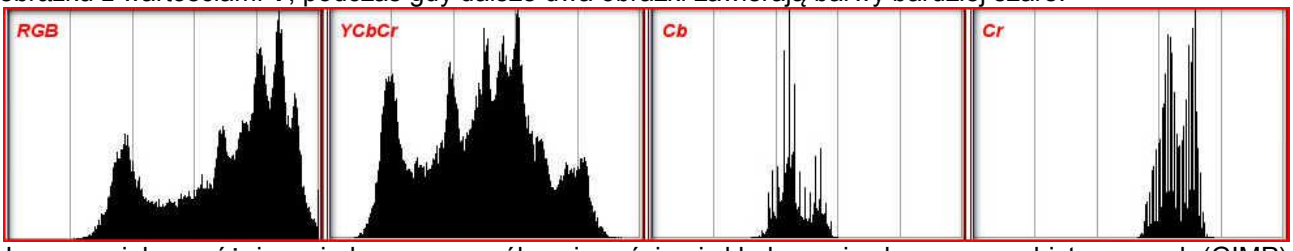
oraz YCbCr na RGB:

$$G = Y - [0{,}25\ Cb + 0{,}25\ Cr]$$
$$R = Cr + G,$$
$$B = Cb + G.$$

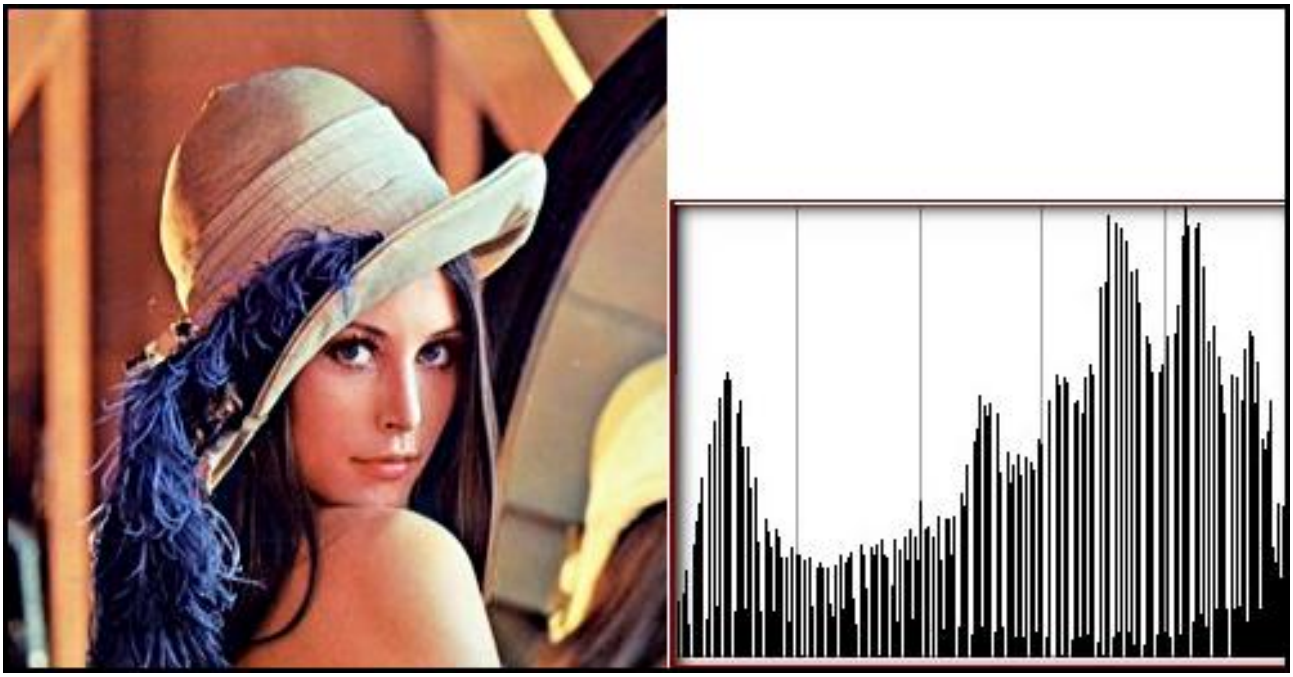
http://www.couleur.org/index.php?page=download
RGBCube Version 1.1.1 for Windows NT/2000/XP: RGBCube-1.1.1.exe (size 7.5MB) licence.txt
ColorSpace Version 1.1.1 for Windows NT/2000/XP: ColorSpace-1.1.1.exe (size 8.8MB)
http://colantoni.nerim.net/download/colorspacemanual-1.0.pdf
ColorSpace Data Viewer 0.70 for Windows NT/2000/XP: CDV-0.70.exe (size 7.5MB)

Przykłady podane poniżej pokazują w jaki sposób przebiega transformacja obrazu z przestrzeni typu TrueColor RGB do przestrzeni barwnej YCbCr.
Posłużę się istniejacym w sieci http://en.wikipedia.org/wiki/Standard_test_image popularnym obrazkiem Lena (Lenna).

Po transformacji do przestrzeni barw **YCbCr** możemy zauważyć, że najwięcej informacji zawartych jest w obrazku z wartościami **Y**, podczas gdy dalsze dwa obrazki zawierają barwy bardziej szare.



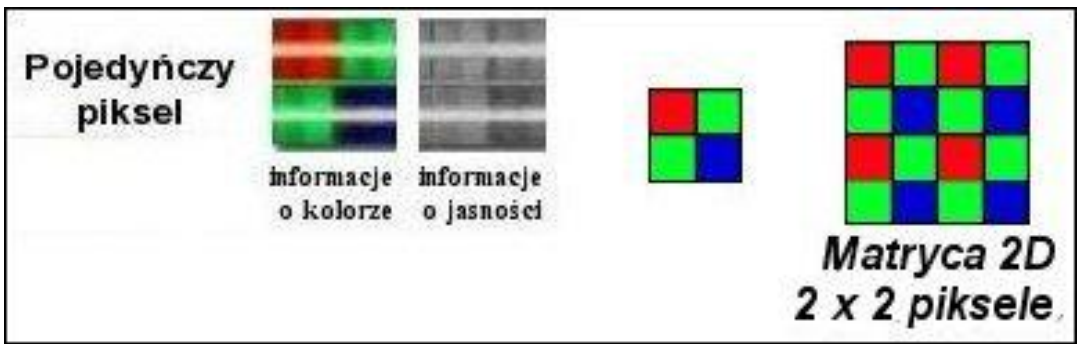
Jeszcze większe różnice między poszczególnymi częściami składowymi zobaczymy na histogramach (GIMP). Wyjaśnienie tej dysproporcji jest logiczne – pierwotny obraz zawierał małą ilość barw, wzgl. barwy nie były rozmieszczone równomiernie. Możemy jednak spróbować w GIMP-ie dokonać poprawy kontrastu obrazu modyfikując histogram. Do tego celu używa się transformacji jednego rozkładu wartości do drugiego rozkładu wartości pikseli przy pomocy Krzywych. W wyniku tej operacji powstaje nowy histogram.

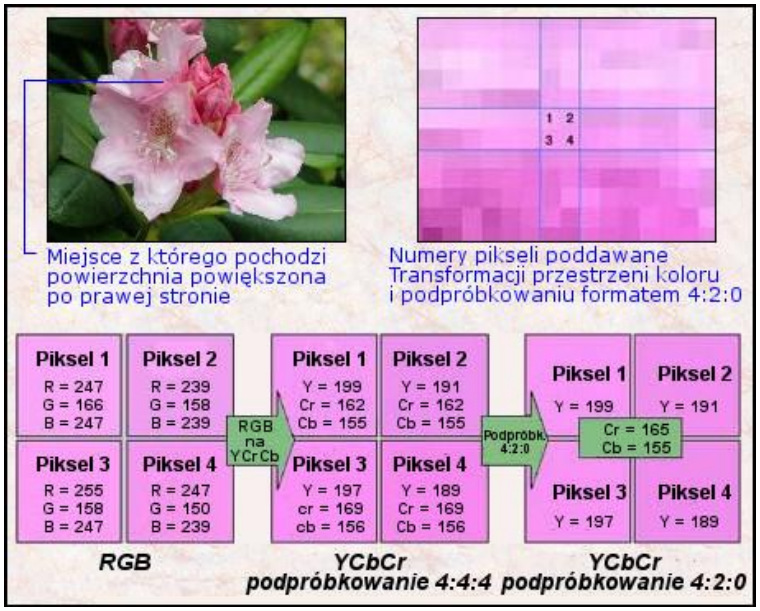


Powyżej obrazek Lena poprawiony krzywymi i jego histogram.
Przykład transformacji mojego obrazka do przestrzeni YCbCr:

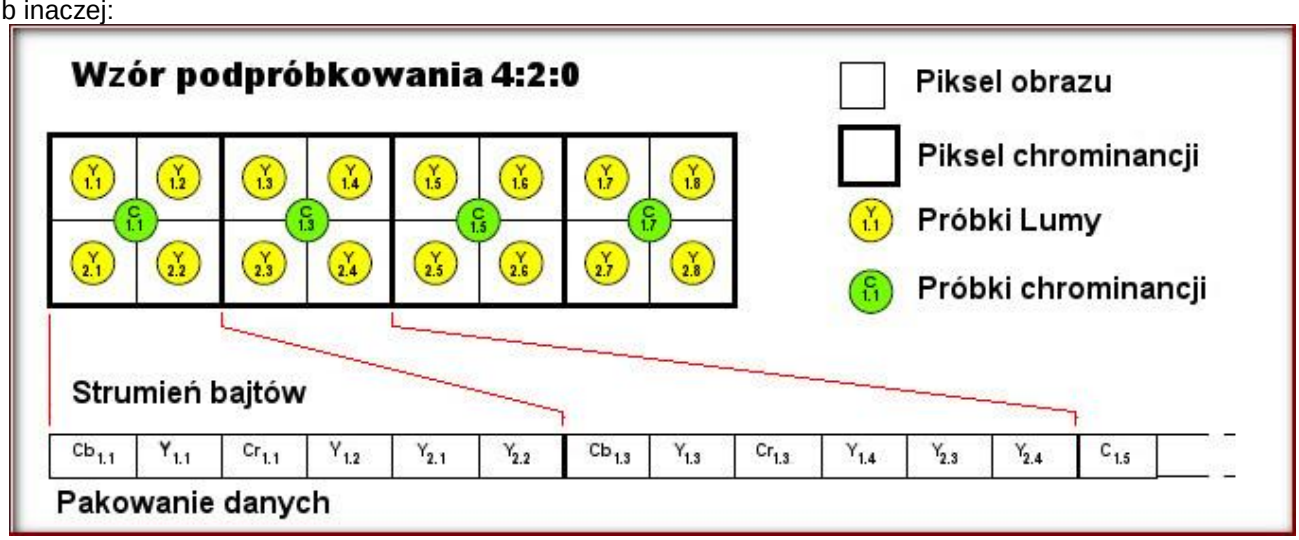
Używanie transformacji barw nie jest zalecane, jeżeli zależy nam na oryginalnej jakości obrazu.
W modelu YCbCr dominującym czynnikiem jest **Y**; Cb i Cr mają mniejszy wpływ na wygląd obrazu. W przeciwieństwie do tego modelu, w RGB wszystkie składniki są równe, tzn. mają jednakowy wpływ na jakość obrazu.
Ta transformacja pomiędzy dwójką przestrzeni barwnych, jest bezstratna, tj. nie może przy niej dojść do żadnej straty informacji o obrazie.

Podpróbkowanie (decymacja) składowych chrominancji CbCr- redukcja rozdzielczości (opcjonalnie)
http://en.wikipedia.org/wiki/Downsampling http://en.wikipedia.org/wiki/Chroma_subsampling
Kiedy już mamy składowe YCbCr stosuje się "podpróbkowanie chrominancji".
Jak wspomniano powyżej, oko ludzkie jest znacznie bardziej wyczulone na zmianę jasności niż barwy. Komórki widzenia czarno-białego wywierają większy wpływ na wzrok niż komórki widzenia dziennego (kolorowego). Dlatego standard JPEG bierze pod uwagę ten fakt, że ludzkie oko jest bardziej wrażliwe na jaskrawość koloru niż na jego odcień. Jeśli będziemy przekształcić dane z 3 kanałów RBG tak, aby jasność była przechowywana w jednej składowej, a kolory w pozostałych dwóch, to okaże się, że możemy usunąć większość informacji o kolorze, bez zauważalnego pogorszenia wyglądu obrazu. Tutaj odbywa się transformacja koloru **RGB** do **YCbCr** (lub inaczej **YUV** dla analogowego systemu kodowania) i obniżenie rozdzielczości.
Składowe chrominancyjne mogą zostać poddane próbkowaniu ze zmniejszoną rozdzielczością. Polega to na uśrednieniu wartości składowych chrominancji dla bloku **2 x 2** piksele. Na każde **2 x 2** punkty luminancji *(Lumy)* **Y** przypada zaledwie jeden punkt różnicowy chrominancji **Cb** i jeden punkt różnicowy chromacji **Cr**.
Uwaga: Przy obrazach w gradacji szarości wykorzystuje się tylko składową Y (brak podpróbkowania).

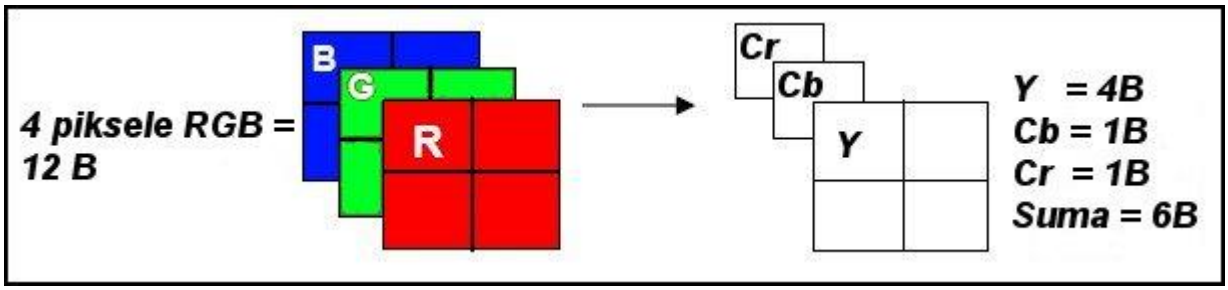


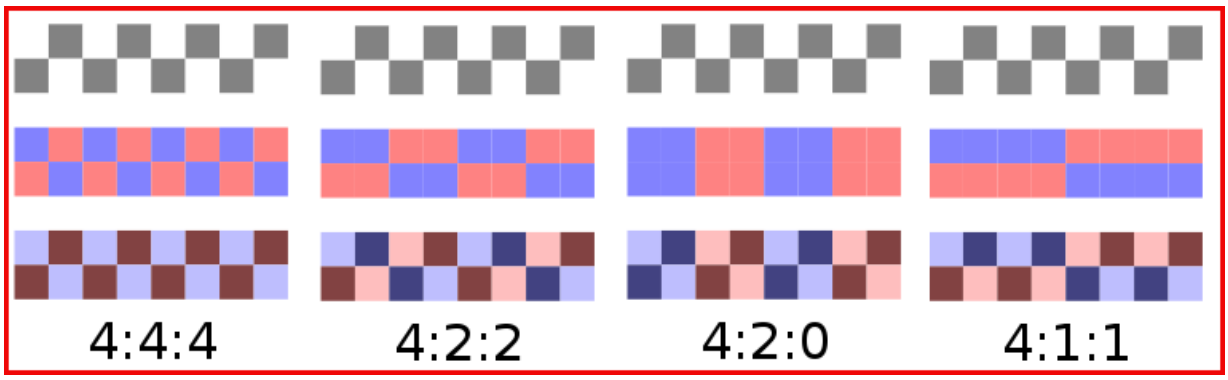


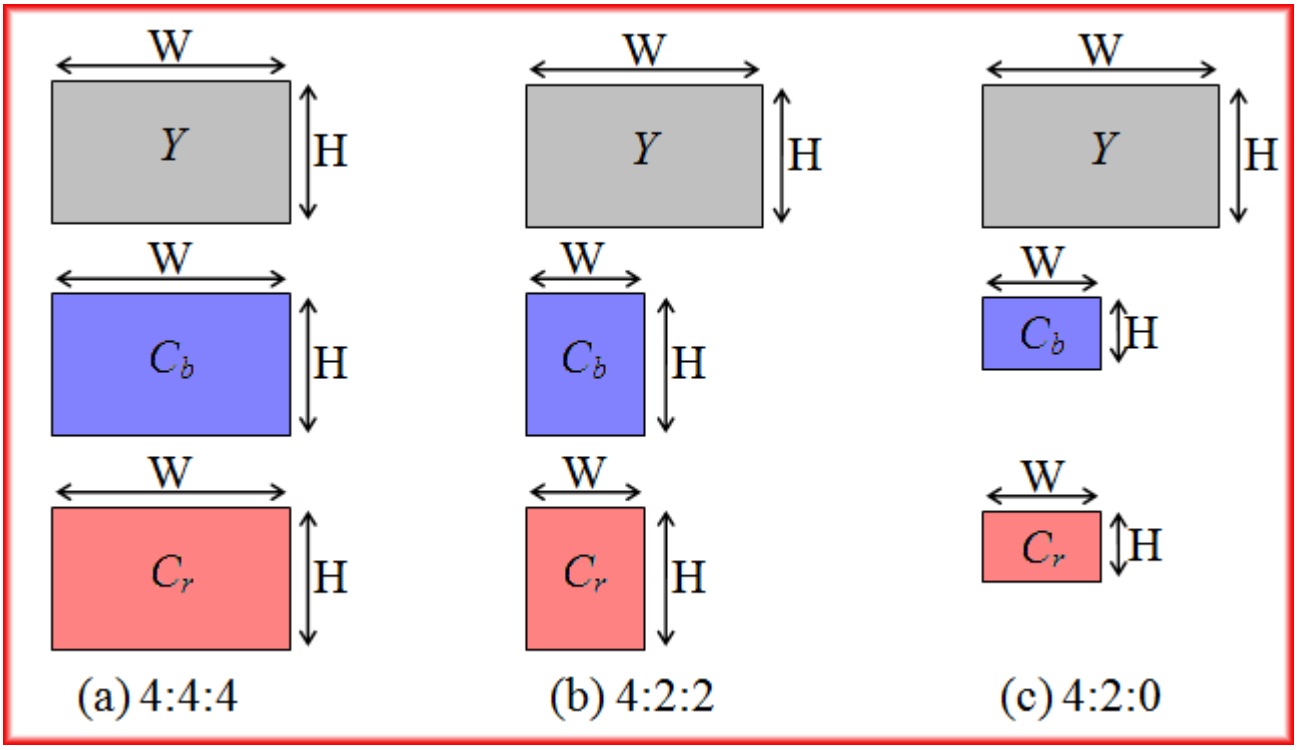
lub inaczej:



Czy:

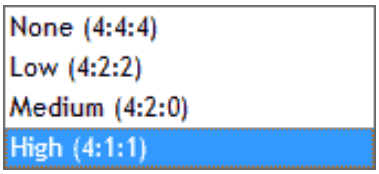






Obraz w kolorach **RGB** *można* transformować na składową luminancyjną (popr. **Luma**) - jasność (**Y**) i dwie składowe różnicowe chrominancyjne [*zabarwienia* - ***antyniebieska*** [niebieski / żółty] **Cb** (**C**hroma**b**lue) **–** i składowa ***antyczerwona*** [czerwony / zielony] **Cr** (**C**hroma**r**ed) - wartości odpowiadające za nasycenie barwy, oznaczają odpowiednio jak bardzo niebieski i czerwony jest dany kolor - wartości te wskazują, jak wiele jest w tym kolorze, odpowiednio niebieskich i czerwonych.
Jest ona definiowana dla dwóch barw (Cr) i (Cb) - w stosunku do jasności kanał (Y) obrazu przez zmniejszaniu obrazu o stały czynnik w kierunku x wyłącznie, albo w obu kierunkach x i y.
W zależności od wymaganej jakości, składowe chrominancyjne **CbCr** mogą zostać poddane ***podpróbkowaniu*** (istnieją różne systemy notacji):

Podpróbkowanie HxV	Opis
1 x 1	Bez podpróbkowania
2 x 1	Podpróbkowanie poziome
1 x 2	Podpróbkowanie pionowe
2 x 2	Podpróbkowanie poziome i pionowe



1.**None** - pozostawione bez zmian (format **4:4:4),** nie będzie żadnego podpróbkowaniu, najwyższa jakość obrazu; zachowuje rozdzielczość barw identyczną jak luminancji, będą zachowane krawędzie i kontrastowe kolory (nie *zjada* kolorów).
2.**Low** - (format **4:2:2**) Oznacza to że rozdzielczość składowej chrominancji jest zmniejszona o połowę w stosunku do składowej luminancji w kierunku poziomym.
3.**Medium** - (format **4:2:0**) Oznacza zmniejszenie rozdzielczości dla składowej chrominancji o połowę w stosunku do składowej luminancji zarówno w kierunku poziomym (wiersze) jak i pionowym (kolumny).
4.**High** – (format **4:1:1**) Oznacza to że rozdzielczość składowej chrominancji jest zmniejszona do jednej czwartej (ćwiartki) w stosunku do składowej luminancji w kierunku poziomym.

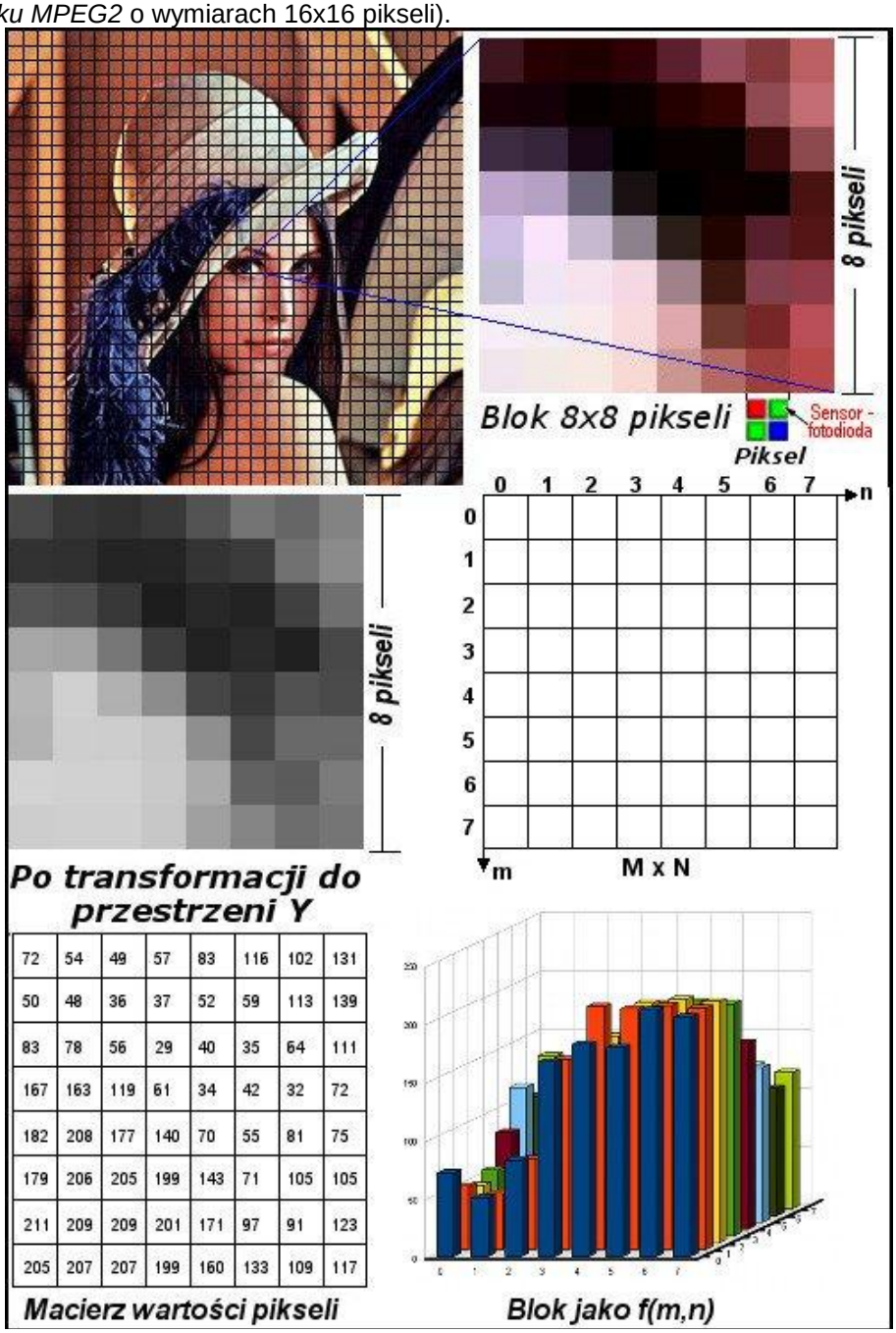
Czyli "podpróbkowanie", określa ile bitów ma być przeznaczonych na sygnał jasności, a ile dla koloru.
Można wybrać format podpróbkowania, który ma wpływ na stratę jakości.
Subsampling widać szczególnie na jedno pikselowych pionowych liniach (tekst). Wtedy kolory przestają odpowiadać oryginałowi (najczęściej blakną). Na zdjęciach tego nie widać, bo jasność sąsiadujących pikseli jest bardzo zbliżona.
Kompresja JPEG kompletnie nie radzi sobie z dużą ilością ostrych krawędzi. JPEG za to doskonale radzi sobie ze zdjęciami, gdzie jest dużo płynnych przejść kolorów.

Kodowanie koloru – daje 50% oszczędności miejsca; otrzymywane tą drogą wyniki są dobre pod względem stopnia kompresji, przy niezauważalnej wizualnie utracie jakości obrazu.
Kodowanie kolorów nie jest stosowane przy obrazach z skalą szarości. W tym przypadku wartość piksela jest bezpośrednio poddawana transformacji DCT.

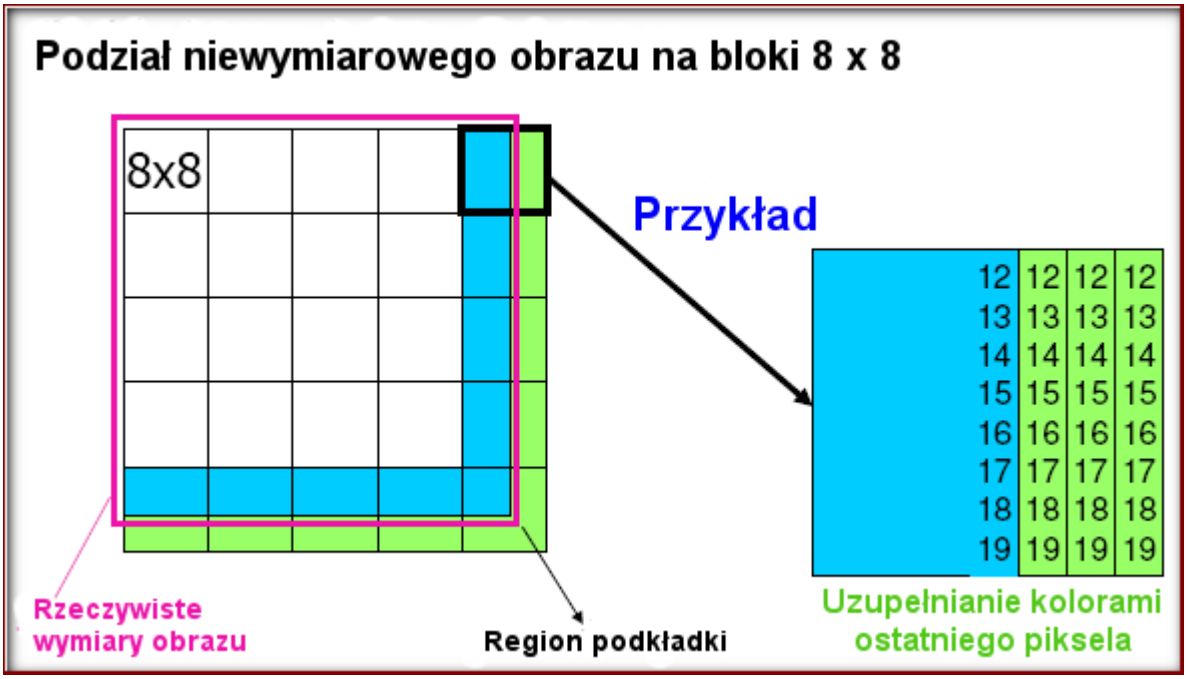
Powyższe dotyczy zamiany z RGB na YUV12, do zapisu informacji o obrazie jest w nim wykorzystywane 12 bitów. Innym formatem YUV jest YUV9. W jego przypadku składowa chrominancji opisuje grupę nie 2x2 piksele, lecz 4x4 piksele. Czyli dla 16 sąsiednich pikseli zapisywana jest taka sama informacja o kolorze. W sumie daje to średnio 9 bitów na piksel. Kodowanie koloru – daje oszczędność miejsca, zmiana bezstratna.

Dla reprezentacji 2 x 2 czyli 4 pikseli i formatu podpróbkowania 4:4:4 potrzebujemy sumarycznie:
(4 piksele) * (8-bit/piksel **Y** + 8-bit/piksel **Cr** + 8-bit/piksel **Cb**) = **96 bit**.
Dla reprezentacji 2 x 2 czyli 4 pikseli i formatu podpróbkowania **4:2:0** potrzebujemy sumarycznie:
(4 piksele) * (8 -bit/piksel **Y** + 8-bit **Cr** + 8-bit **Cb**) = **48 bit** nastąpiło zmniejszenie danych rastrowych o 50%.

Podział obrazu na bloki. (Dekompozycja obrazu na bloki)
Kiedy już mamy "podpróbkowane" składowe **YCbCr** (lub inaczej **YUV** dla analogowego systemu kodowania), w kolejnym kroku stosuje się podział każdej składowej (warstwy) obrazu na nie zachodzącego na siebie bloki 8x8 pikseli (*w przypadku MPEG2* o wymiarach 16x16 pikseli).



Z powodu podziału obrazu na bloki 8x8, pojawia się problem, gdy wymiary obrazu nie są całkowitą wielokrotnością liczby 8. Wtedy obraz rozszerzamy do rozmiarów będących wielokrotnością 8 poprzez uzupełnienie bloków leżących na prawej i dolnej krawędzi obrazu (powtórzenie ostatniej kolumny/wiersza odpowiednią liczbę razy) do całkowitej wielokrotności 8.



Można to wykonać poprzez:

- uzupełnienie zerami – sposób najprostszy ale jednocześnie najgorszy,
- uzupełniamy kolorem ostatniego piksela,
- uzupełniamy takimi pikselami, aby odwrotna transformacja kosinusowa (IDCT) zwróciła wartości pikseli z obrazu jak najbliższe wartości oryginalnych.

Każdy może zapytać w tym momencie, dlaczego cały obraz dzielony jest na mniejsze bloki?
Wpływa na to kilka czynników. Po pierwsze, transformacja małych bloków daje lepsze wyniki niż pełnego obrazu, w niedużym bloku (8×8), większość energii skoncentrowana jest w składowych o małych współrzędnych. Po drugie umożliwia to wykonywanie obliczeń na kilku blokach równolegle, ponadto całemu procesowi towarzyszy powstawania tzw. pierścieni Gibbsa. Są to widoczne pierścienie pojawiające się w obszarach wokół ostrych krawędzi. Efekt ten jest niewidoczny przy małych blokach, które ograniczają propagację artefaktów. Oczywiście rozwiązanie blokowe nie jest pozbawione wad. Główną z nich jest dobrze znany efekt różnicy krawędzi bloków dla wyższych współczynników kompresji.
http://de.wikipedia.org/wiki/Gibbssches_Ph%C3%A4nomen
W niektórych przypadkach, algorytm wywołuje „aureole” wokół ostrych poziomych i pionowych granic w obrazie (efekt – fenomen Gibbsa). Wyjaśnia się go stratami w zakresie wysokich częstotliwości i zależy od współczynnika kompresji. Należy próbować albo stopniem kompresji, albo jakością obrazu.

Jak pracuje algorytm

Algorytm pracuje sekwencyjnie, skanowanie wszystkich komponentów obrazu z lewej do prawej i z góry do dołu. Ponieważ przetwarzanie każdego bloku 8x8 odbywa się bez odniesienia do innych, dalej będziemy koncentrować się tylko na pojedynczym bloku. W szczególności skupimy się tylko na bloku z składowej Lumy Y. Przy okazji można zauważyć że jaśniejszym pikselom składowych Y Cr Cb odpowiadają większe wartości, oraz że Luma Y wykazuje większe zmiany niż chrominancje.

Krok : Transformacja przy użyciu DCT – dyskretnej transformaty kosinusowej

Kiedy już mamy "podpróbkowanie” składowe Y,Cb,Cr, w kolejnym kroku stosuje się podział każdej składowej (kanału) obrazu na nie zachodzącego na siebie bloki 8x8.

Tutaj właśnie rozpoczyna się ważny etap kompresji JPEG, dzięki któremu otrzymywane są dobre efekty kompresji JPEG.
Podkreślić należy, że bez:
- odkrycia przez Fouriera szeregów nie było by dzisiejszej komunikacji i informatyki. Rozwoju technologii pozwalającej na zamianę sygnałów z osi czasu na oś częstotliwości, który umożliwił powstanie zaawansowanych układów przesyłania i magazynowania danych: kompresji, kodowania, korekty.
- dyskretnej transformaty kosinusowej (wprowadzili ją w roku 1974 Nasir Ahmed, T.Natarajan i Kamisetty R. Rao), nie było by kompresji JPEG.

Możliwość analizowania widma niektórych sygnałów pozwala nam zrozumieć ich fizykę i zachowanie.
Są to zaawansowane metody matematyczne, ale dalej korzystając z wiedzy innych, spróbuję opisać je w miarę prosto i szczegółowo aby wiedzieć, co one dają.

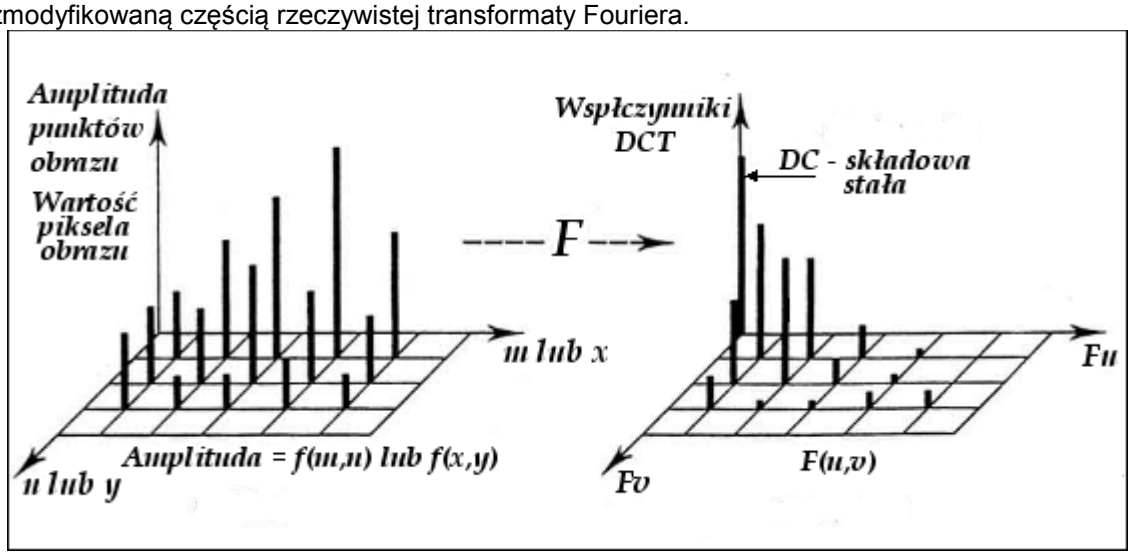
> **Analiza spektralna obrazów pozwala, nie wnosząc widocznych zniekształceń, odrzucić najmniej znaczącą część informacji.**

Transformata Fouriera przenosi obraz do dziedziny częstotliwości. Intuicyjną definicję częstotliwości obrazu oprzeć można na kontrastach zawartych w rozpatrywanym obrazie. Wysoki kontrast to wysoka częstotliwość,

niski kontrast to niska częstotliwość. Krawędzie zawartych w obrazie powierzchni o jednolitej barwie, znajdujących się na tle o barwie różnej od barwy tych powierzchni to obszary o wysokiej częstotliwości. Można wyobrazić sobie, że DCT powoduje usunięcie drobnych, mało znaczących składowych części obrazu określających obszary o wysokiej częstotliwości co spowoduje „rozmycie" obrazu.
Jeżeli więc obraz zawiera ostre krawędzie, które zapisywane są w zakresie wysokich częstotliwości, DCT spowoduje ich "*rozmazanie*".

Celem transformacji DCT jest to, by zamiast przetwarzania oryginalnych składników obrazu, można było stwierdzić które jego elementy to istotne fragmenty obrazu niosące istotną informację i wpływające na jego jakość, a które to „szum", jakiego nie zauważa ludzkie oko, i który nie jest bardzo istotny.
DCT pozwala na dyskretne przejścia pomiędzy barwami. Po przekształceniu obrazu za pomocą DCT możemy pracować z pewnymi „częstotliwościami" obecnymi w oryginalnym obrazie. Te przestrzenne częstotliwości są ściśle związane z ilością detali obecnych w obrazie. Wysokie częstotliwości przestrzenne związane są z dużą ilością detali, natomiast niskie z małą ilością.

Transformacja z uwagi na swoją złożoność, jest operacją czasochłonną i wykonywanie jej na przykład w czasie rzeczywistym dla złożonych sygnałów (obraz) wymaga dużej mocy obliczeniowej. Jest to związane z funkcją, która wymaga wykonania dużej ilości mnożeń i dodawań. Pojedyncza dwuwymiarowa operacja obliczenia DCT lub IDCT na bloku o rozmiarze 8x8 pikseli wymaga kilkuset cykli instrukcji w typowym procesorze DSP.
Transformata DCT jest transformatą bezstratną która przekształca zbiór danych który został spróbkowany z ustaloną częstotliwością próbkowania w jego komponenty częstotliwościowe. Liczba próbek musi być skończona i być wielokrotnością 2 w celu skrócenia czasu obliczeń.
Dyskretna kosinusowa transformata (DCT) w zastosowaniu do typowych obrazów o dużej korelacji wartości pikseli daje dużą efektywność. Bazę tego przekształcenia stanowi zbiór ortonormalnych funkcji kosinusoidalnych, będący zmodyfikowaną częścią rzeczywistej transformaty Fouriera.



1. Dyskretna transformata kosinusowa (DCT)

http://www.mathematik.de/spudema/spudema_beitraege/beitraege/rooch/nkap06.html !!!!!!!!
Terminologia - określenia:
W tym akapicie spotkamy się z określeniami:
Transformata (przekształcenie).
Stosujemy ją, ponieważ w wielu sytuacjach przetwarzanie danych w ich pierwotnej postaci jest niedogodne, albo z powodu natury stosowanych operacji lub otrzymywanych wyników. W kompresji danych przekształceniu podlega grupa danych w inną grupę danych (inną postać danych), np. po przekształceniu wyniki prezentowane w dziedzinie czasu (pokazujące zmiany amplitudy fali w czasie), zostaną zmienione tak, że fala będzie reprezentowana w dziedzinie częstotliwości, czyli zależności amplitudy i fazy fali od częstotliwości, a nie od czasu. Transformatą funkcji ***f*** będzie funkcja ***F*** jednoznacznie określona i jednoznacznie powiązana z funkcją ***f***. W kontekście kompresji danych najistotniejsze jest, aby przekształcone dane były nieskorelowane bo wtedy większość energii sygnału, będzie skoncentrowana w niewielkiej części transformaty, która zostanie wykorzystana. Nie będę tutaj przytaczał wszystkich uwarunkowań i wymagań, które musi spełniać transformata aby miała sens, była jednoznaczna, ortogonalna itp. - zainteresowani szczegółami muszą szukać w literaturze np. w internecie.
Macierz http://pl.wikipedia.org/wiki/Macierz http://portalwiedzy.onet.pl/6822,,,,macierz,haslo.html

- układ zapisanych w postaci prostokątnej tablicy danych nazywanych elementami bądź współczynnikami będących elementami ustalonego zbioru liczbowego.
- macierzą o wymiarze M x N nazywamy strukturę złożoną z M wierszy i N kolumn
- jest to uporządkowana prostokątna tablica liczb, dla której zdefiniowane są działania algebraiczne dodawania (odejmowania) i mnożenia.
- sposób strukturyzacji danych w postaci macierzy pozwala na łatwą organizację i zmianę danych poprzez przestawianie, łączenie, wydzielanie itp.

Wykorzystuje się je np. do przechowywania współczynników i w ogólności danych zależnych od wielu parametrów.

Dyskretna transformata kosinusowa (*DCT – Discrete Cosine Transform)* należy do zbioru metod prowadzących tzw. kodowanie transformatowe dyskretnego (próbkowanego) jednowymiarowym czy więcej rozmiarowym sygnałem. Do tego samego zbioru należą również optymalna KLT (*Karhunen-Loéve Transform*) czy znana FFT (*Fast Fourier Transform* – szybka dyskretna transformata Fouriera).

W większości przy transformacjach prowadzi się transfer (czy mapowanie) przetwarzanego sygnału z domeny czasu (przestrzennej) do domeny częstotliwości, ponieważ istnieje domniemanie, że przykładowo obrazy rzeczywistych przedmiotów nie zawierają dużo energii w wyższych częstotliwościach i jest wtedy dobrze zgromadzić jak największa ilość odpowiednich danych do małej ilości współczynników. Kodowanie jako wynik miało by prowadzić do całkowitego obniżenia ilości bitów niosących informację wizualną (psychowizualna redundancja).
Istnieje wiele typów dyskretnych transformat kosinusowych, które są w literaturze oznakowane rzymskimi liczbami jako DCTI aż do DCTIV. Poniżej przedstawimy tylko DCTII. Jest ona najszerzej stosowana i tworzy podstawę dla większości praktycznych zadań, które DCT w jakiś sposób wykorzystują. Dwuwymiarowa transformata kosinusowa typu II jest stosowana w wielu formatach np. Standard JPEG, MPEG1, MPEG2 itd..

Transformacja DCT – pozwala na dyskretne przejścia pomiędzy barwami; zmiana bezstratna; Oryginalne wartości komponentów mogą być odtworzone odwrotną transformacją DCT. W praktyce błędy zaokrągleń mogą spowodować, że odtworzone wartości różnią się nieco, jednak różnice te są nieistotne w porównaniu z efektami kwantyzacji. Konsekwencja niezależnego przetwarzania fragmentów obrazu jest tak zwany „efekt blokowy", charakterystyczne i zbyt widoczne po nadmiernej kompresji, blokowe w kształcie artefakty.

Głównym problemem kodowania transformatowego opartego na blokach DCT są gwałtowne skoki wartości na granicach zrekonstruowanych bloków (tzw. efekt blokowy). Jednym ze sposobów minimalizacji tego zjawiska jest używanie większych bloków (w przypadku mocno skompresowanych obrazków używane przez JPEG bloki 8x8 są stanowczo za małe, 16x16 lub nawet 32x32 byłyby bardziej odpowiednie).

Definicja DCT i odwrotnej DCT
Co to jest -

Jednowymiarowa dyskretna transformata kosinusowa (1D DCT)

Jednowymiarowa dyskretna transformata kosinusowa typu II (oznaczana skrótowo jako 1D DCT) służy do przetwarzania jednorozmiarowego dyskretnego (fj. próbkowanego) sygnału otrzymanego np. z matrycy CCD. Nie muszą to być oczywiście liczby całkowite. Czyli
Jedno wymiarowa DCT konwertuje tablicę liczb, które przedstawiają amplitudy sygnału w różnych punktach w czasie, w inną tablicę liczb, z których każda reprezentuje amplitudę pewnych częstotliwości komponentów z oryginalnej tablicy. Wynikający z tablicy numer zawiera taką samą liczbę wartości jak oryginał tablicy.
Każdą wartość tego sygnału oznaczamy jako **f*(m)***, gdzie ***m*** jest pozycją (indeksem) sygnału.
Po zastosowaniu jednowymiarowej dyskretnej transformaty kosinusowej uzyskamy zbiór (prostą tablicę wyników - macierz) współczynników transformaty DCT, które będziemy oznaczać ***F(u)***.
Jednowymiarowa DCT nie jest definiowana dla M < 2.
Jednowymiarowa (1D DCT), 8 - mio punktowa dyskretna transformata kosinusowa (ang. DCT - Discrete Cosine Transform) określona jest następująco:

$$F(u)=C(u)\frac{\sqrt{2}}{\sqrt{M}}\sum_{m=0}^{M-1} f(m)\cos\frac{\pi(2m+1)u}{2M} \quad \textbf{(1)}$$

gdzie:
F(u) = amplituda oscylacji (drgań) kosinusa dla częstotliwości o indeksie **u**,
f(m) = wartości przestrzenne pikseli w tablicy (bloku) - próbka sygnału – „Dane".
M = ilość wierszy w wejściowej tabeli danych – macierzy, (oznacza typ/rząd DCT, w prakt. Zastos. **8 lub 16**).
***F*(u)** = współczynniki transformaty DCT - w domenie częstotliwości
m* =** indeks próbki z wiersza (lub kolumny - ***n) w domenie przestrzennej od **0 do M – 1**
u* =** indeks wiersza (lub kolumny - ***v) w domenie częstotliwości od **0** do **M – 1**
Współczynnik skalowania (stała) ***C(u)*** jest równy:

$$C(u) = \begin{cases} 1/\sqrt{2} & \text{gdy } u = 0, \\ 1 & \text{w pozostałych przypadkach} \end{cases}$$

W szczególnym przypadku gdy **u = 0** mamy:

$$F(u)=c(u)\frac{\sqrt{2}}{\sqrt{M}}\sum_{m=0}^{M-1}f(m)=\frac{1}{\sqrt{M}}\sum_{m=0}^{M-1}f(m)$$

Dla pozostałych przypadków i M = 8 mamy $\sqrt{2}/\sqrt{M} = 1/2$

Trzeba pamiętać, że zawsze musi istnieć również odwrotna dyskretna transformata kosinusowa **(IDCT),** zarówno jednowymiarowa jak i dwuwymiarowa.

$$f(m)=\frac{1}{\sqrt{M}}\sum_{m=0}^{M-1}F(o)+c(u)\frac{\sqrt{2}}{\sqrt{M}}\sum_{m=0}^{M-1}F(u)\cos\frac{\pi(2m+1)u}{2M}$$

dla m = 1, 2, ...,M - 1

Z uwagi na zakres opracowania pomine ich omawianie.

Przy pomocy jednowymiarowej dyskretnej transformaty kosinusowej możemy opracować zawsze maksymalnie **M** próbek, w przypadku większej ich ilości powtarzamy wzór kolejno dla dalszych M próbek sygnału wejściowego.

Transformatę 1-wymiarową DCT, łatwo można składać, dzięki czemu działa również na blokach 2 lub więcej wymiarowych. Może być przeprowadzona kolejno jako, transformata jednowymiarowa najpierw wierszami, następnie kolumnami lub najpierw kolumnami, następnie wierszami. W obu przypadkach wynik będzie identyczny.

Jest to bardzo ważna własność, z której wynika, że funkcje bazowe mogą być - obliczane rozłącznie i następnie wyniki mnożone. Zmniejsza to ilość operacji matematycznych (mnożeń i dodawań) tym samym zwiększając współczynnik sprawności obliczeń.

Ponieważ dla kompresji JPEG z dwuwymiarową DCT (2D-DCT) jest stosowany M = N = 8, będziemy odtąd zwłaszcza do rozważenia 1D-DCT również stosować M = 8.

Aby zrozumieć i utrwalić pojęcia, zignorujemy we wzorze komponenty ***f(m)*** i współczynniki przed sumą.

Skorzystamy więc z $\sum_{m=0}^{M-1}\cos\left[\frac{\pi(2m+1)u}{2M}\right]$ dla **M** = 8 i zmienianych wartości (***u*** = 0,1,2,…, 7) otrzymamy wyniki, które tworzą **linie falowe (waveform) - fale** przedstawione poniżej w tabeli i na rysunkach.

Wyrażenie $\cos\frac{\pi(2m+1)u}{2M}$ nazywamy ***jądrem transformacji*** lub ***funkcją bazową kosinusa***.

Dogodne jest wyrażenia transformaty w postaci macierzy:

$$F_{(u)}=\begin{bmatrix} f_{0,0} & \circ\circ\circ & f_{0,7} \\ \circ & \circ\circ\circ & \circ \\ \circ & \circ\circ\circ & \circ \\ \circ & \circ\circ\circ & \circ \\ f_{7,0} & \circ\circ\circ & f_{7,7} \end{bmatrix}\xrightarrow{m}=\begin{bmatrix} 1 & 1 & \circ\circ\circ & 1 \\ \cos\frac{\pi}{16} & \cos\frac{3\pi}{16} & \circ\circ\circ & \cos\frac{15\pi}{16} \\ \circ & \circ & \circ\circ\circ & \circ \\ \circ & \circ & \circ\circ\circ & \circ \\ \circ & \circ & \circ\circ\circ & \circ \\ \cos\frac{7\pi}{16} & \cos\frac{21\pi}{16} & \circ\circ\circ & \cos\frac{105\pi}{16} \end{bmatrix}$$

Zapis w postaci macierzy dla M=8, przy założeniu f(m) = 1 i pominięciu współczynników przed sumą

$$F(u)=c(u)\frac{\sqrt{2}}{\sqrt{M}}\sum_{m=0}^{M-1}f(m)\cos\frac{\pi(2m+1)u}{2M}$$

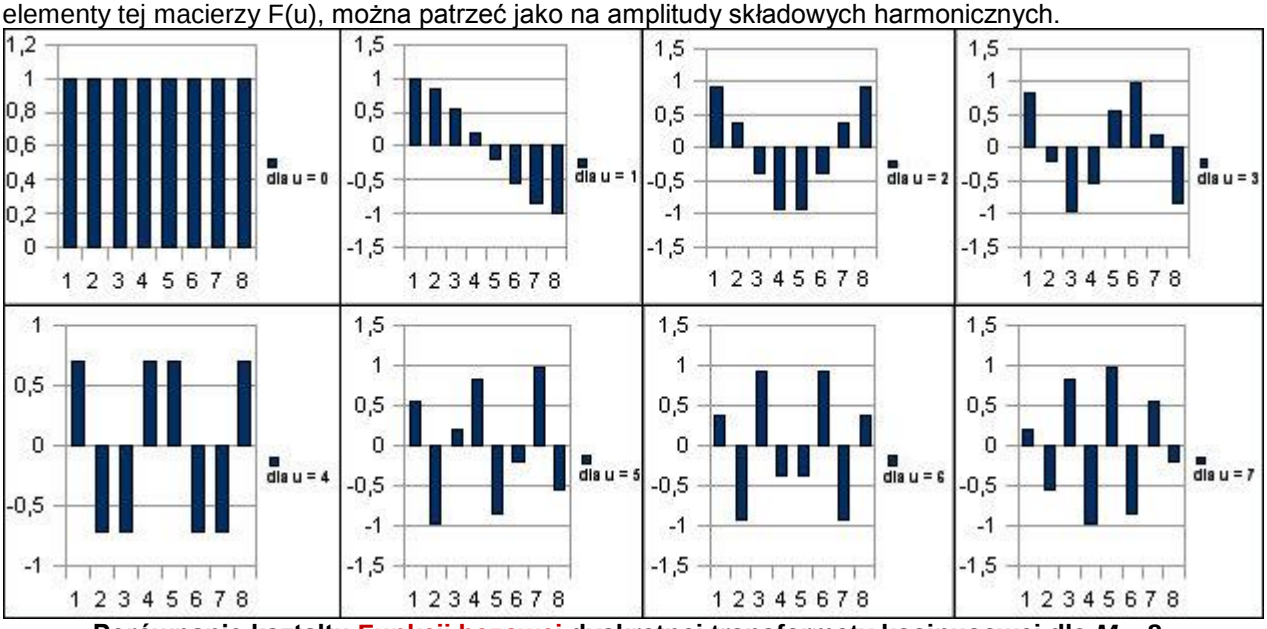
W poniższej tabeli zestawiono wartości zmiennych $c_i = \cos(i*\pi/16)$ występujących w funkcji bazowej kosinusa (1D-DCT) dla M = 8:

Tabela wartości zmiennych - c_i				
	Kąt		Wartość	
	radiany	stopnie	precyzyjnie	cyfrowo
c_0	0	0	1	1
c_1	π/16	11,25	$\frac{1}{2}\sqrt{2+\sqrt{2+\sqrt{2}}}$	0.9807853
c_2	2π/16	22,50	$\frac{1}{2}\sqrt{2+\sqrt{2}}$	0.9238794
c_3	3π/16	33,75	$\frac{1}{2}\sqrt{2+\sqrt{2-\sqrt{2}}}$	0.8314695
c_4	4π/16	45	$\frac{1}{2}\sqrt{2}$	0.7071065
c_5	5π/16	56,25	$\frac{1}{2}\sqrt{2-\sqrt{2-\sqrt{2}}}$	0.5555699
c_6	6π/16	67,50	$\frac{1}{2}\sqrt{2-\sqrt{2}}$	0.3826829
c_7	7π/16	78,75	$\frac{1}{2}\sqrt{2-\sqrt{2+\sqrt{2}}}$	0.1950897

tablica wartości wag dla wierszy w macierzy 8x8 będzie następująca:

m →	0	1	2	3	4	5	6	7
u = 0	1	1	1	1	1	1	1	1
u = 1	0,98	0,83	0,56	0,2	-0,2	-0,56	-0,83	-0,98
u = 2	0,92	0,38	-0,38	-0,92	-0,92	-0,38	0,38	0,92
u = 3	0,83	-0,2	-0,98	-0,56	0,56	0,98	0,2	-0,83
u = 4	0,71	-0,71	-0,71	0,71	0,71	-0,71	-0,71	0,71
u = 5	0,56	-0,98	0,2	0,83	-0,83	-0,2	0,98	-0,56
u = 6	0,38	-0,92	0,92	-0,38	-0,38	0,92	-0,92	0,38
u = 7	0,2	-0,56	0,83	-0,98	0,98	-0,83	0,56	-0,2

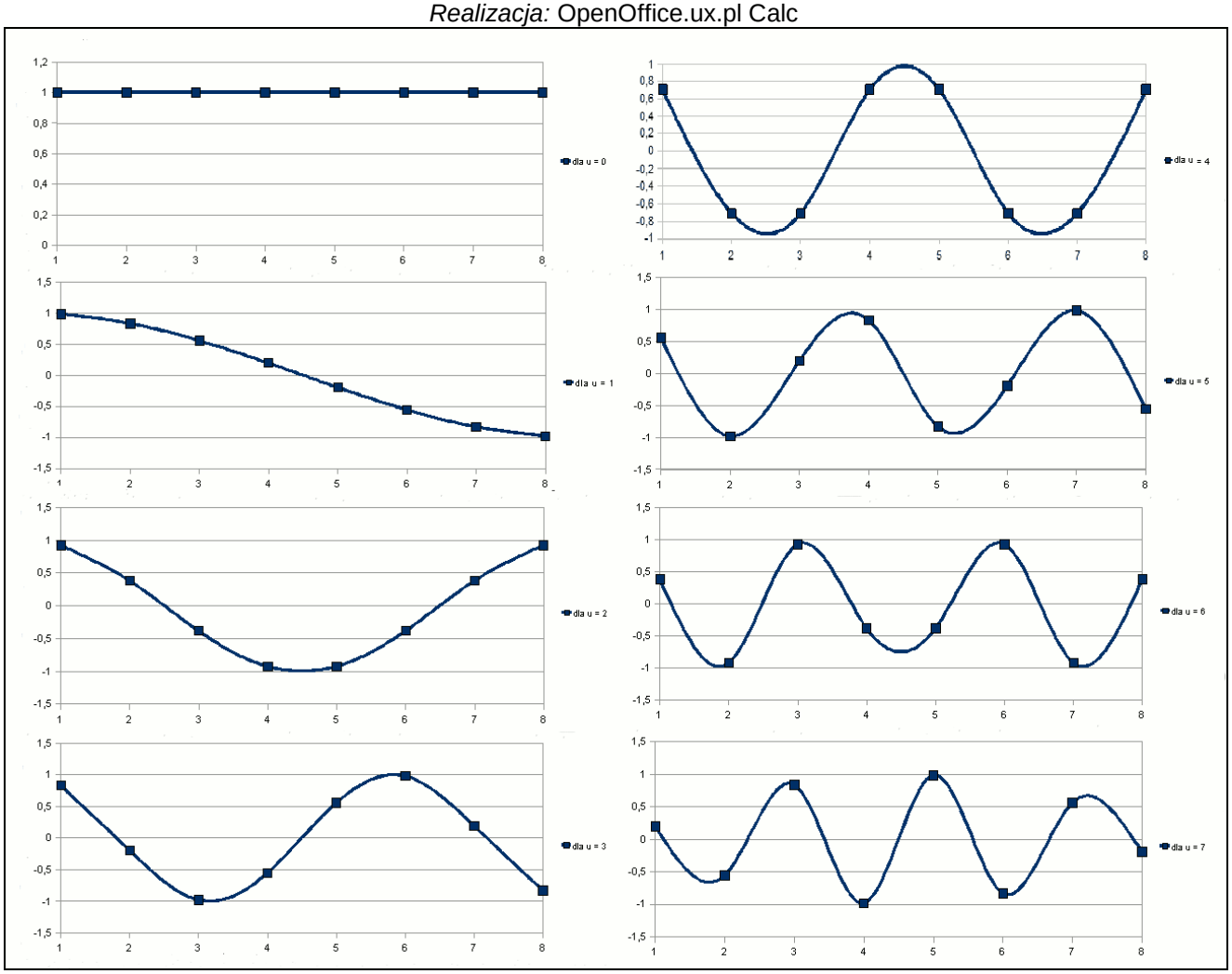
Pierwszy element w tablicy wyników jest zwykłą *średnią wartością* wszystkich próbek wejściowych w tablicy i jest w literaturze określany jako współczynnik ***DC*** -Direct current.
Wszystkie pozostałe elementy przekształcenia w każdej tablicy wyników wskazują amplitudy częstotliwości konkretnych komponentów tablicy wejściowej, i są nazywane współczynnikami ***AC*** - Alternating current.
Częstotliwościowa wartość próbki określana dla każdej częstotliwości jest obliczana w drodze średniej ważonej dla całego zestawu. Te współczynniki wagowe są jak cosinusy fali, których częstotliwości są proporcjonalne do wyników tablicy indeksów.
Na elementy tej macierzy F(u), można patrzeć jako na amplitudy składowych harmonicznych.

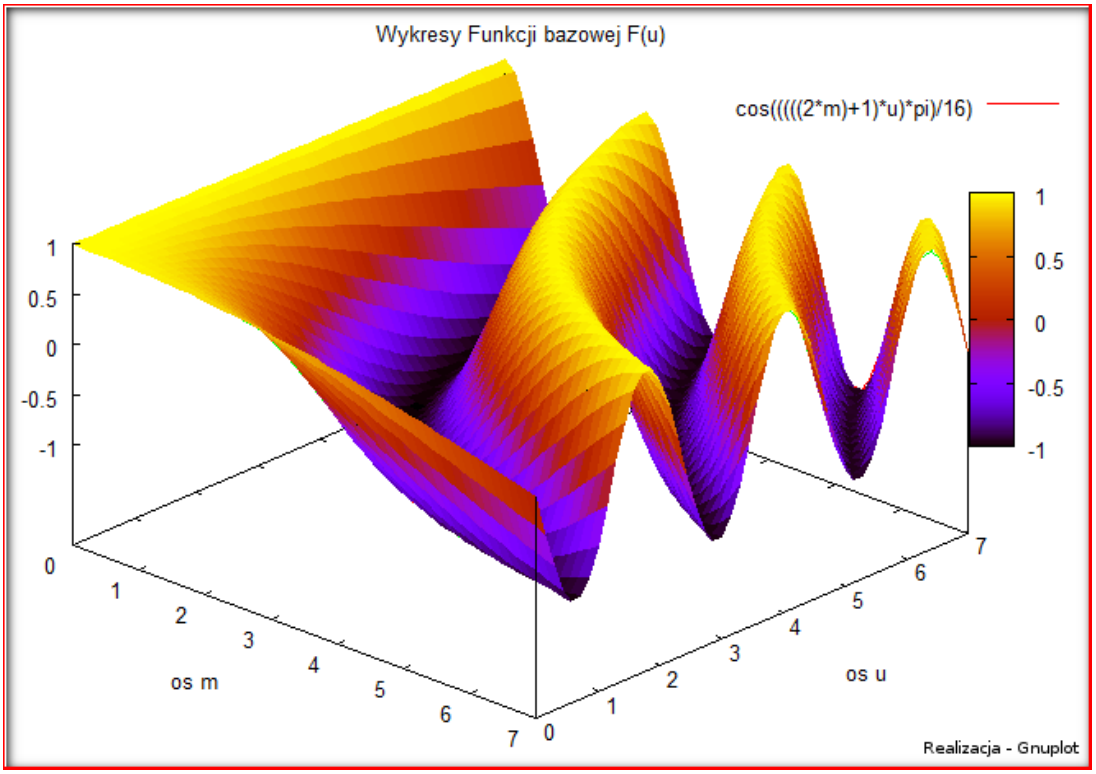


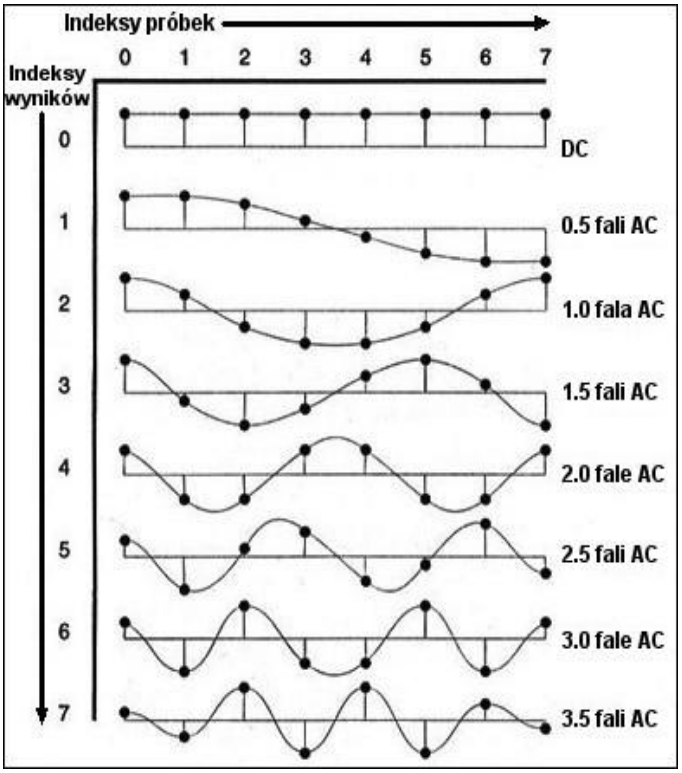
Porównanie kształtu Funkcji bazowej dyskretnej transformaty kosinusowej dla *M* = 8.

Poniżej wartości z wierszy funkcji bazowej są zaznaczone jako punkty. Punkty układają się jak podstawowa funkcja ciągła cosinusa. Amplituda skalowania i maksymalna częstotliwość będą się różnić w zależności od wartości M.

Realizacja: OpenOffice.ux.pl Calc







Widać dokładnie że, pierwszy **kształt fali** (***u* = 0**) odtwarza wartość składowej stałej (**DC**), podczas gdy, wszystkie inne **kształty fal** (***u* = 1,2,…, 7**) są o kolejno rosnących częstotliwościach.
Kształty tych fal są nazywane **Bazową funkcją cosinusa**.
Dla **u = 1** (lub **v = 1**). mamy:

$$\cos\left(\frac{(2m+1)u\pi}{16}\right)=\cos\left(\frac{1}{16}2\pi m+\frac{\pi}{16}\right) \qquad \cos\left(\frac{2(m+0{,}5)u\pi}{16}\right)=\cos\left(\frac{1}{8}\pi m+\frac{\pi}{16}\right)$$

lub

reprezentuje to falę kosinusoidalną, która dla ***m* = 0 do 7** wynosi pół okresu. Fala jest przesunięta o **π/16** lub pół szerokości piksela. Jest to miejsce maksimum w lewej granicy bloku pikseli, pół piksela w lewo pierwszej próbki.
Najwyższą wartość mamy dla **u = 7** (lub **v = 7**)

$$\cos\left(\frac{(2m+1)u\pi}{16}\right)=\cos\left(\frac{7}{16}2\pi m+\frac{7\pi}{16}\right) \qquad \cos\left(\frac{2(m+0{,}5)u\pi}{16}\right)=\cos\left(\frac{7}{8}\pi m+\frac{7\pi}{16}\right)$$

lub

Reprezentuje to falę kosinusoidalną wynoszącą 3,5 pełnego okresu, na drugiej stronie bloku.

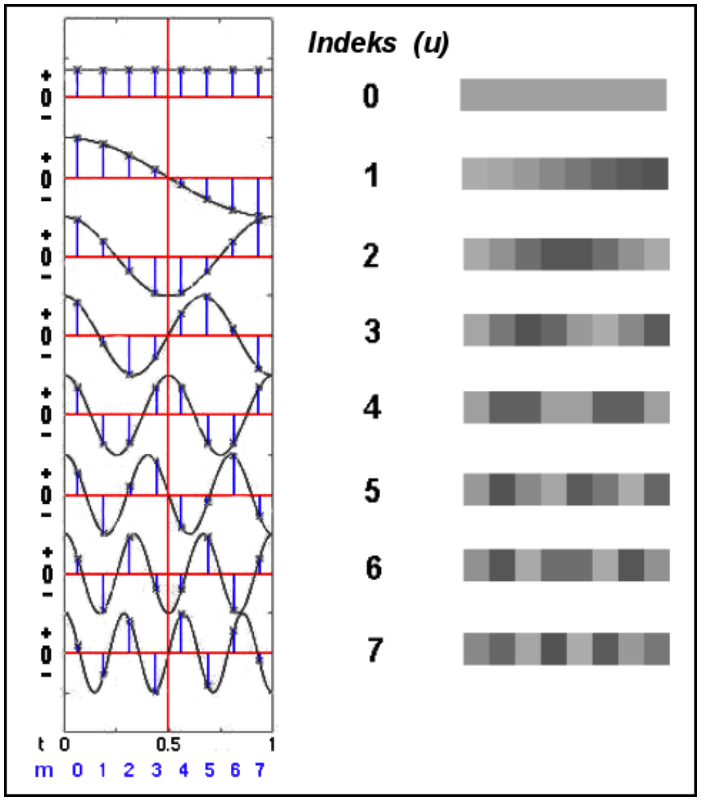
Biorąc pod uwagę powyższe rysunki, można również spojrzeć na to w inny sposób:
Punktem wyjścia są ciągłe zmiany cosinusa z różną częstotliwością:

$$\cos tu\pi \;;\; u=0,1,\ldots,7 \;;\; 0<t<1$$

u jest dyskretnym indeksem, dla zmiennej ***t*** w przedziale od 0 do 1 wzrasta częstotliwość
Zatem dla u = 0 mamy funkcję składowej stałej; dla u = 1 falę półokresu cosinusa; dla u = 2 pełny okres fali cosinusa itp.
Zmienna **t** jest zawarta w całości w zakresie od 0 do 1 dla punktów skanowania ***m***

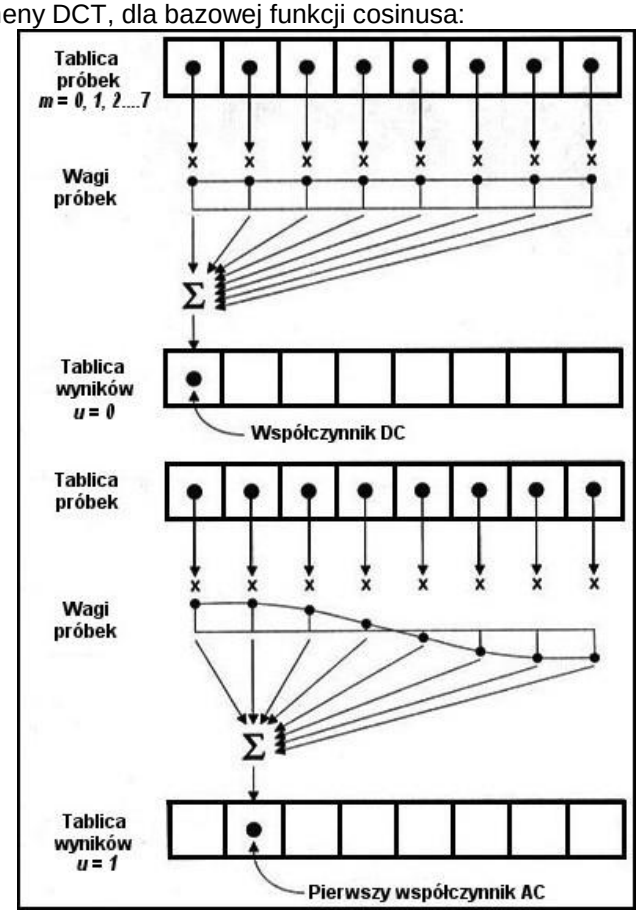
$$t=\frac{2m+1}{16} \;;\; m=0,1,\ldots,7$$

Ciąg wartości t = f(m) = {0; 0,1875; 0,3125; 0,4375; 0,5625; 0,6875; 0,8125; 0,9375;}

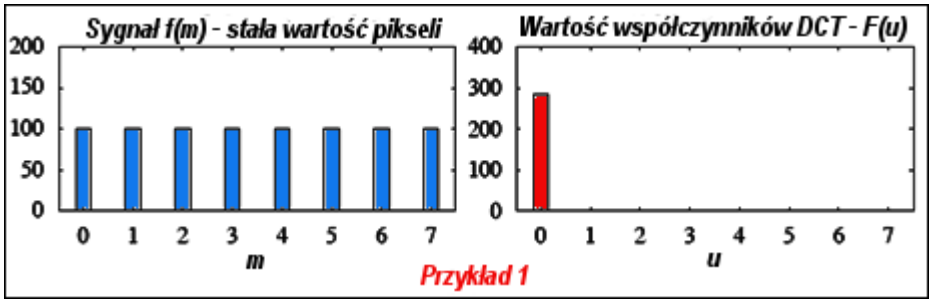


A więc, w formacie JPEG, pobieramy próbki w odstępach co 1 / 8 między punktami bloku JPEG i odległości 1 / 16 od krawędzi. Te określenia ułatwią zrozumienie następnej definicji - 2DCT.

Poniżej pokazano w formie graficznej procedurę matematycznego przekształcania (transformacji) z domeny przestrzennej piksela do domeny DCT, dla bazowej funkcji cosinusa:



Przykład:
wyznaczenie współczynnika DC dla danych w postaci ciągu próbek funkcji $\mathbf{f_1(m)}$:



Zakładamy, że sygnał **f_1(m)** w tablicy wartości pikseli ma stałą wartość wynoszącą **100**

$$F_{(0)}=\frac{1}{\sqrt{M}}\sum_{m=0}^{M-1} f(m)$$

- Pamiętamy, że:
- ***F_1*(u=0)** = [1/√8] × (1×100 + 1×100 +1×100 +1×100 + 1×100 + 1×100 +1×100 + 1×100) ≈ 283

Dla ***u*** **= 1**, z wykresu funkcji bazowej zauważamy, że:

cos (π/16) = - cos(15π/16),
cos(3π/16) = - cos(13π/16),
cos(5π/16) = - cos(11π/16),
cos(7π/16) = - cos(9π/16)

Mamy więc:

- ***F_1*(u=1)** = (1/2) × [cos(π/16) × 100 + cos(3π/16) × 100 + cos(5π/16) × 100 + cos(7π/16) × 100 + cos(9π/16) × 100 + cos(11π/16) × 100 + cos(13π/16) × 100 + cos(15π/16) × 100] = 0
- To samo otrzymamy dla: ***F_1*(u=2), *F_1*(u=3), …, *F_1*(u=7)** dla każdego = 0

Jeśli założymy, że (intensywność każdego piksela) **f(m)** jest równa w całym wierszu, czyli ***f(m)= 1,*** oraz gdy uwzględnimy współczynniki we wzorze przed sumą, wtedy tablica wartości wag transformaty bazowej dla wierszy w macierzy 8x8 będzie wyglądać następująco:

Indeks próbki/Wynik	0	1	2	3	4	5	6	7
0	+0.3536	+0.3536	+0.3536	+0.3536	+0.3536	+0.3536	+0.3536	+0.3536
1	+0.4904	+0.4157	+0.2778	+0.0975	-0.0975	-0.2778	-0.4157	-0.4904
2	+0.4619	-0.1913	-0.1913	-0.4619	-0.4619	-0.1913	+0.1913	+0.4619
3	+0.4157	-0.0975	-0.4904	-0.2778	+0.2778	+0.4904	+0.0975	-0.4157
4	+0.3536	-0.3536	-0.3536	+0.3536	+0.3536	-0.3536	-0.3536	+0.3536
5	+0.2778	-0.4904	+0.0975	+0.4157	-0.4157	-0.0975	+0.4904	-0.2778
6	+0.1913	-0.4619	+0.4619	-0.1913	-0.1913	+0.4619	-0.4619	+0.1913
7	+0.0975	-0.2778	+0.4157	-0.4904	+0.4904	-0.4157	+0.2778	-0.0975

Macierz transformaty bazowej jednowymiarowej dla M=8 można przedstawić również następująco:

$$T_{(8)} = \frac{1}{\sqrt{8}} \cdot \begin{pmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ a_1 & b_1 & c_1 & d_1 & -d_1 & -c_1 & -b_1 & -a_1 \\ a_2 & b_2 & -b_2 & -a_2 & -a_2 & -b_2 & b_2 & a_2 \\ b_1 & -d_1 & -a_1 & -c_1 & c_1 & a_1 & d_1 & -b_1 \\ 1 & -1 & -1 & 1 & 1 & -1 & -1 & 1 \\ c_1 & -a_1 & d_1 & b_1 & -b_1 & -d_1 & a_1 & -c_1 \\ b_2 & -a_2 & a_2 & -b_2 & -b_2 & a_2 & -a_2 & b_2 \\ d_1 & -c_1 & b_1 & -a_1 & a_1 & -b_1 & c_1 & -d_1 \end{pmatrix}$$

gdzie:

$$a_1 = \sqrt{2} \cdot \cos\frac{\pi}{16} \approx 1.387, \quad b_1 = \sqrt{2} \cdot \cos\frac{3\pi}{16} \approx 1.176,$$

$$c_1 = \sqrt{2} \cdot \cos\frac{5\pi}{16} \approx 0.786, \quad d_1 = \sqrt{2} \cdot \cos\frac{7\pi}{16} \approx 0.276,$$

$$a_2 = \sqrt{2} \cdot \cos\frac{2\pi}{16} \approx 1.307, \quad b_2 = \sqrt{2} \cdot \cos\frac{6\pi}{16} \approx 0.541.$$

A tak wygląda macierz obliczonych wartości wag transformaty bazowej jednowymiarowej:

$$T = \begin{bmatrix} 0.3536 & 0.3536 & 0.3536 & 0.3536 & 0.3536 & 0.3536 & 0.3536 & 0.3536 \\ 0.4904 & 0.4157 & 0.2778 & 0.0975 & -0.0975 & -0.2778 & -0.4157 & -0.4904 \\ 0.4619 & 0.1913 & -0.1913 & -0.4619 & -0.4619 & -0.1913 & 0.1913 & 0.4619 \\ 0.4157 & -0.0975 & -0.4904 & -0.2778 & 0.2778 & 0.4904 & 0.0975 & -0.4157 \\ 0.3536 & -0.3536 & -0.3536 & 0.3536 & 0.3536 & -0.3536 & -0.3536 & 0.3536 \\ 0.2778 & -0.4904 & 0.0975 & 0.4157 & -0.4157 & -0.0975 & 0.4904 & -0.2778 \\ 0.1913 & -0.4619 & 0.4619 & -0.1913 & -0.1913 & 0.4619 & -0.4619 & 0.1913 \\ 0.0975 & -0.2778 & 0.4157 & -0.4904 & 0.4904 & -0.4157 & 0.2778 & -0.0975 \end{bmatrix}$$

A tak macierz transportowana obliczonych wartości wag transformaty bazowej jednowymiarowej:

$$T' = \begin{bmatrix} 0.3536 & 0.4904 & 0.4619 & 0.4157 & 0.3536 & 0.2778 & 0.1913 & 0.0975 \\ 0.3536 & 0.4157 & 0.1913 & -0.0975 & -0.3536 & -0.4904 & -0.4619 & -0.2778 \\ 0.3536 & 0.2778 & -0.1913 & -0.4904 & -0.3536 & 0.0975 & 0.4619 & 0.4157 \\ 0.3536 & 0.0975 & -0.4619 & -0.2778 & 0.3536 & 0.4157 & -0.1913 & -0.4904 \\ 0.3536 & -0.0975 & -0.4619 & 0.2778 & 0.3536 & -0.4157 & -0.1913 & 0.4904 \\ 0.3536 & -0.2778 & -0.1913 & 0.4904 & -0.3536 & -0.0975 & 0.4619 & -0.4157 \\ 0.3536 & -0.4157 & 0.1913 & 0.0975 & -0.3536 & 0.4904 & -0.4619 & 0.2778 \\ 0.3536 & -0.4904 & 0.4619 & -0.4157 & 0.3536 & -0.2778 & 0.1913 & -0.0975 \end{bmatrix}$$

Następnym krokiem jest zapoznanie się z pojęciem transformaty dwuwymiarowej DCT, która jest określona ogólnym wzorem:

$$F(u,v) = \frac{2}{\sqrt{MN}} C(u)C(v) \sum_{m=0}^{M-1} \sum_{n=0}^{N-1} f(m,n) \cos\left(\frac{(2m+1)u\pi}{2M}\right) \cos\left(\frac{(2n+1)v\pi}{2N}\right)$$

gdzie:

$$C(u,v) = \begin{cases} 1/\sqrt{2} & u,v = 0 \\ 1 & \text{w pozostałych przypadkach} \end{cases}$$

M = ilość wierszy w wejściowym zestawie danych
N = ilość kolumn w wejściowym zestawie danych
m = indeksy próbek z wiersza w domenie przestrzennej gdzie: 0 ≤ m ≤ M - 1
n = indeksy próbek z kolumny w domenie przestrzennej gdzie: 0 ≤ n ≤ M - 1
f(m,n) = wartości przestrzenne pikseli w bloku - "Dane"
u = indeks wiersza w domenie częstotliwości
v = indeks kolumny w domenie częstotliwości
F(u,v) = współczynniki transformaty w domenie częstotliwości

Ponieważ jednak w kompresji JPEG z dwuwymiarową DCT (2D-DCT) stosujemy M = N = 8, odtąd do rozważań będziemy stosować zależność:

$$F(u,v) = \frac{C(v)}{2}\sum_{n=0}^{7}\left[\frac{C(u)}{2}\sum_{m=0}^{7} f(m,n)\cos\left(\frac{(2m+1)u\pi}{16}\right)\right]\cos\left(\frac{(2n+1)v\pi}{16}\right)$$

gdzie: **dla M = N = 8**

$$C(u,v) = \begin{cases} 1/\sqrt{2} & u,v = 0 \\ 1 & \text{w pozostałych przypadkach} \end{cases}$$

m = indeksy próbek z wiersza w domenie przestrzennej : 0,1,2...7
n = indeksy próbek z kolumny w domenie przestrzennej : 0,1,2...7
f(m,n) = wartości przestrzenne pikseli w bloku - "Dane"
u = indeks wiersza w domenie częstotliwości : 0,1,2...7
v = indeks kolumny w domenie częstotliwości : 0,1,2...7
F(u,v) = współczynniki transformaty w domenie częstotliwości

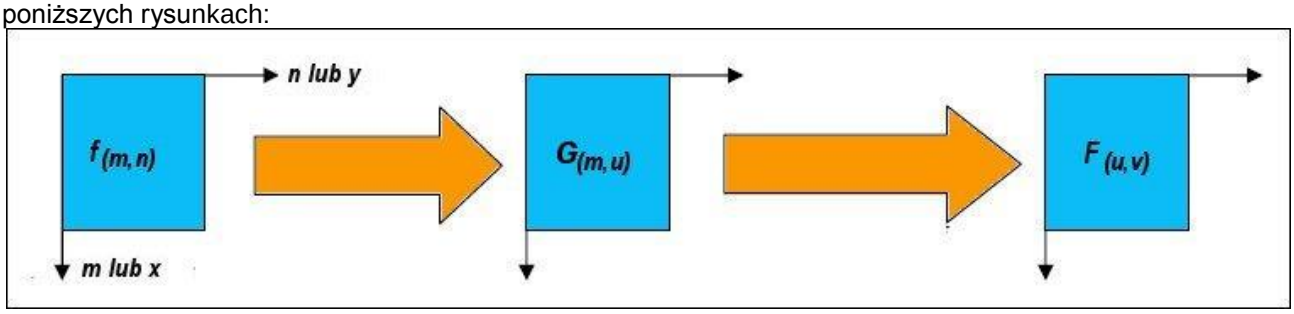
W poniższym przykładzie, pokazany jest jeden z sposobów wizualizacji bazowej dwuwymiarowej dyskretnej transformacji kosinusowej. W odróżnieniu od poprzedniego przykładu nie będę pokazywać bezpośrednio przebiegu dyskretnego sygnału, ale każdą wartość funkcji bazowej określimy stopniem szarości na obrazie rastrowym. Bazową funkcję 2D DCT możemy określić z wzoru:

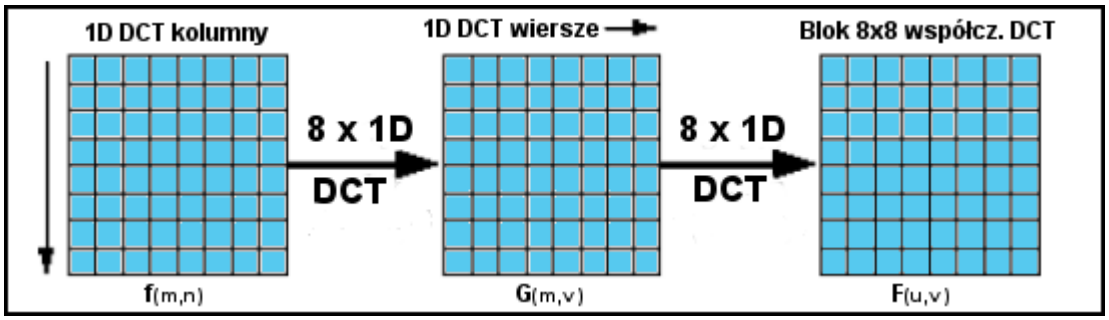
$$\cos\left[\frac{\pi(2m+1)u}{2N}\right]\cos\left[\frac{\pi(2n+1)v}{2N}\right]$$

Nie ma tu matematyki liczb zespolonych. Każda wartość w widmie jest amplitudą funkcji bazowej.
Dla sygnałów ***rzeczywistych*** **transformata** DCT ma również wartości rzeczywiste.
Znalezienie **dyskretnej dwuwymiarowej transformacie kosinusowej** (2D DCT) sprowadza się do złożenia dwóch transformat jednowymiarowych:

- najpierw stosujemy 1DCT wierszami ***m*** a potem, kolumnami ***n*** *macierzy* otrzymanej w wyniku pierwszej transformacji **lub**
- stosujemy 1DCT (pionowo) wzdłuż każdej kolumny ***n*** a potem, (poziomo) wzdłuż każdego wiersza ***m*** *macierzy* otrzymanej w wyniku pierwszej transformacji.

Wyznaczenie transformaty dwuwymiarowej za pomocą jednowymiarowych transformacji, pokazano na poniższych rysunkach:





Jest to bardzo ważna własność, przedstawia, że funkcje bazowe mogą być - obliczane rozłącznie i następnie wyniki mnożone. To zmniejsza ilość operacji matematycznych (mnożeń i dodawań) tym samym zwiększając współczynnik sprawności obliczeń.
Implementacja: według definicji - od 11 do 13 mnożeń i 29 dodawań, szybka DCT - 5 mnożeń i 29 dodawań.

World record is 11 multiplies and 29 adds. (C. Loeffler, A. Ligtenberg and G. Moschytz, "Practical Fast 1-D DCT Algorithms with 11 Multiplications", Proc. Int'l. Conf. on Acoustics, Speech, and Signal Processing 1989 (ICASSP `89), pp. 988-991)

Transformata 2D:

$$F_{(u,v)} = T*X*T'$$

lub

$$F_{NxN} = A_{NxN} * X_{NxN} * A^{T}_{NxN}$$

$$\begin{bmatrix} F_{11} & \cdots & \cdots & F_{1N} \\ \vdots & \ddots & & \vdots \\ \vdots & & \ddots & \vdots \\ F_{N1} & \cdots & \cdots & F_{NN} \end{bmatrix} = \begin{bmatrix} a_{11} & \cdots & \cdots & a_{1N} \\ \vdots & \ddots & & \vdots \\ \vdots & & \ddots & \vdots \\ a_{N1} & \cdots & \cdots & a_{NN} \end{bmatrix} \begin{bmatrix} x_{11} & \cdots & \cdots & x_{1N} \\ \vdots & \ddots & & \vdots \\ \vdots & & \ddots & \vdots \\ x_{N1} & \cdots & \cdots & x_{NN} \end{bmatrix} \begin{bmatrix} a_{11} & \cdots & \cdots & a_{N1} \\ \vdots & \ddots & & \vdots \\ \vdots & & \ddots & \vdots \\ a_{1N} & \cdots & \cdots & a_{NN} \end{bmatrix}$$

dane

transformata bazowa 1D wierszy **transformata bazowa 1D kolumn**

Czyli:

$$F_{(u,v)} = \begin{bmatrix} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} & \circ\circ\circ & \frac{1}{\sqrt{2}} \\ \frac{1}{2}\cos\frac{\pi}{16} & \frac{1}{2}\cos\frac{3\pi}{16} & \circ\circ\circ & \frac{1}{2}\cos\frac{15\pi}{16} \\ \circ & \circ & \circ\circ\circ & \circ \\ \circ & \circ & \circ\circ\circ & \circ \\ \circ & \circ & \circ\circ\circ & \circ \\ \frac{1}{2}\cos\frac{7\pi}{16} & \frac{1}{2}\cos\frac{21\pi}{16} & \circ\circ\circ & \frac{1}{2}\cos\frac{105\pi}{16} \end{bmatrix} \begin{bmatrix} f_{0,0} & \circ\circ\circ & f_{0,7} \\ \circ & \circ\circ\circ & \circ \\ \circ & \circ\circ\circ & \circ \\ \circ & \circ\circ\circ & \circ \\ f_{7,0} & \circ\circ\circ & f_{7,7} \end{bmatrix} \begin{bmatrix} \frac{1}{\sqrt{2}} & \frac{1}{2}\cos\frac{\pi}{16} & \circ\circ\circ & \frac{1}{2}\cos\frac{7\pi}{16} \\ \frac{1}{\sqrt{2}} & \frac{1}{2}\cos\frac{3\pi}{16} & \circ\circ\circ & \frac{1}{2}\cos\frac{21\pi}{16} \\ \circ & \circ & \circ\circ\circ & \circ \\ \circ & \circ & \circ\circ\circ & \circ \\ \circ & \circ & \circ\circ\circ & \circ \\ \frac{1}{\sqrt{2}} & \frac{1}{2}\cos\frac{15\pi}{16} & \circ\circ\circ & \frac{1}{2}\cos\frac{105\pi}{16} \end{bmatrix}$$

Przykład:
mnożenie macierzy kwadratowych

$$X = \begin{bmatrix} 100 & 100 & 98 & 99 \\ 100 & 100 & 94 & 94 \\ 98 & 97 & 96 & 100 \\ 100 & 99 & 97 & 94 \end{bmatrix} \quad A = \begin{bmatrix} 0.5 & 0.5 & 0.5 & 0.5 \\ 0.5 & -0.5 & 0.5 & -0.5 \\ 0.5 & 0.5 & -0.5 & -0.5 \\ 0.5 & -0.5 & -0.5 & 0.5 \end{bmatrix} \quad A^T = \begin{bmatrix} 0.5 & 0.5 & 0.5 & 0.5 \\ 0.5 & -0.5 & 0.5 & -0.5 \\ 0.5 & 0.5 & -0.5 & -0.5 \\ 0.5 & -0.5 & -0.5 & 0.5 \end{bmatrix}$$

$$A = \begin{bmatrix} a & b & c \\ d & e & f \\ g & h & i \end{bmatrix}$$

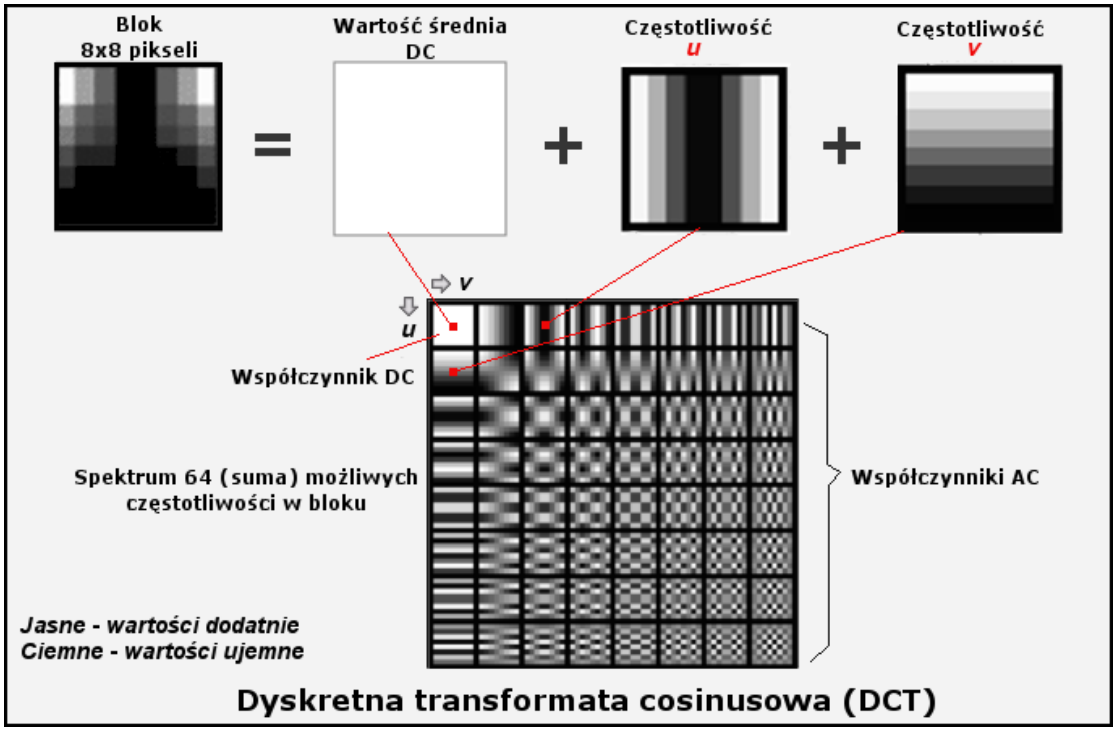
X - macierz danych
A - macierz transformaty dla wierszy
A^T- macierz transformaty transponowanej dla kolumn

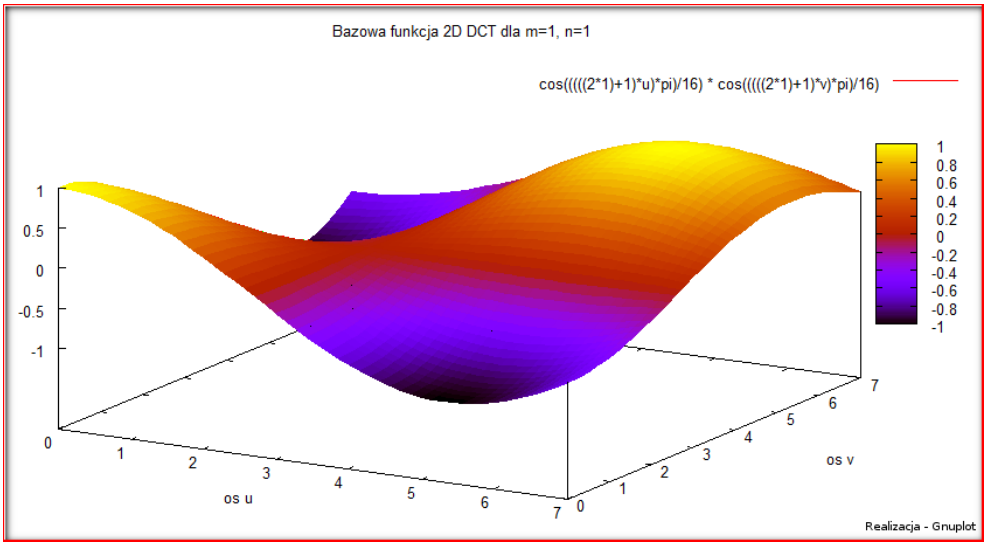
$$F_{NxN} = A_{NxN} * X_{NxN} * A^{T}_{NxN}$$

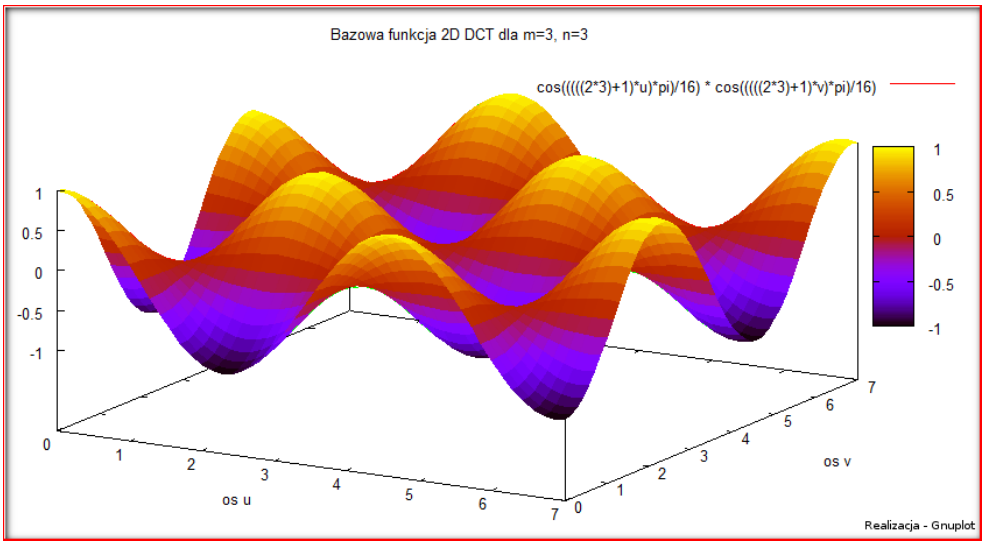
$$A * X = \begin{bmatrix} 0.5 & 0.5 & 0.5 & 0.5 \\ 0.5 & -0.5 & 0.5 & -0.5 \\ 0.5 & 0.5 & -0.5 & -0.5 \\ 0.5 & -0.5 & -0.5 & 0.5 \end{bmatrix} \begin{bmatrix} 100 & 100 & 98 & 99 \\ 100 & 100 & 94 & 94 \\ 98 & 97 & 96 & 100 \\ 100 & 99 & 97 & 94 \end{bmatrix} = \begin{bmatrix} 199 & 198 & 192.5 & 193.5 \\ -1 & -1 & 1.5 & 5.5 \\ 1 & 2 & -0.5 & -0.5 \\ 1 & 1 & 2.5 & -0.5 \end{bmatrix}$$

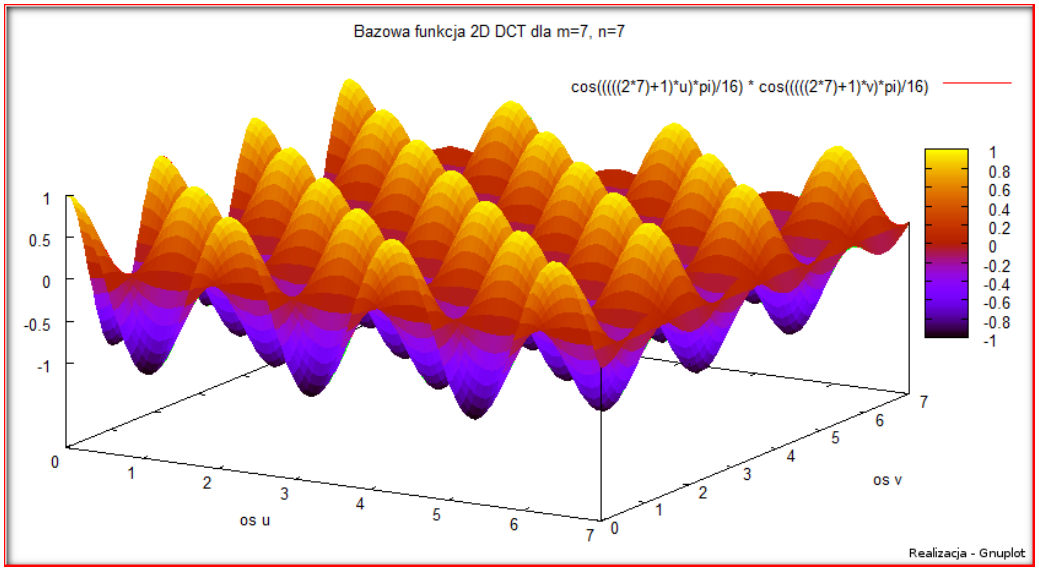
$$* A^T = \begin{bmatrix} 199 & 198 & 192.5 & 193.5 \\ -1 & -1 & 1.5 & 5.5 \\ 1 & 2 & -0.5 & -0.5 \\ 1 & 1 & 2.5 & -0.5 \end{bmatrix} \begin{bmatrix} 0.5 & 0.5 & 0.5 & 0.5 \\ 0.5 & -0.5 & 0.5 & -0.5 \\ 0.5 & 0.5 & -0.5 & -0.5 \\ 0.5 & -0.5 & -0.5 & 0.5 \end{bmatrix} = \begin{bmatrix} 391.5 & 0 & 5.5 & 1 \\ 2.5 & -2 & -4.5 & 2 \\ 1 & -0.5 & 2 & -0.5 \\ 2 & 1.5 & 0 & -1.5 \end{bmatrix}$$

$$A^T = \begin{bmatrix} a & d & g \\ b & e & h \\ c & f & i \end{bmatrix} \quad F_{(u,v)} = \begin{bmatrix} 391.5 & 0 & 5.5 & 1 \\ 2.5 & -2 & -4.5 & 2 \\ 1 & -0.5 & 2 & -0.5 \\ 2 & 1.5 & 0 & -1.5 \end{bmatrix}$$


Blok
8x8 pikseli
Wartość średnia
DC
Częstotliwość
u
Częstotliwość
v
v
u
Współczynnik DC
Spektrum 64 (suma) możliwych
częstotliwości w bloku
Współczynniki AC
Jasne - wartości dodatnie
Ciemne - wartości ujemne
Dyskretna transformata cosinusowa (DCT)


Bazowa funkcja 2D DCT dla m=1, n=1
cos(((((2*1)+1)*u)*pi)/16) * cos(((((2*1)+1)*v)*pi)/16)
os u
os v
Realizacja - Gnuplot


Bazowa funkcja 2D DCT dla m=3, n=3
cos(((((2*3)+1)*u)*pi)/16) * cos(((((2*3)+1)*v)*pi)/16)
os u
os v
Realizacja - Gnuplot



Na powyższym rysunku przedstawiono graficznie odwzorowanie 2D-DCT.
Jak można zobaczyć na rysunkach, funkcja bazowa pokazuje progresywny wzrost częstotliwości zarówno w kierunkach pionowych i poziomych. Funkcja bazowa (0,1) jest połową okresu fali kosinusoidalnej w jednym kierunku i wartością stałą w drugim. Funkcja bazowa dla (1,0) jest podobna lecz obrócona o 90^0.

Z wzoru dla transformaty dwuwymiarowej mamy:

$$DC = F(0,0) = 1/4*1/\sqrt{2}*1/\sqrt{2} = 1/8 * \sum_{n=0}^{7}\sum_{m=0}^{7} f_{(m,n)}$$

Ponieważ **f(m,n)** ≤ 2^{P-1}, największa z wartości współczynnika

$$DC = F(0,0) = 1/8 * \sum_{n=0}^{7}\sum_{m=0}^{7} f_{(m,n)} \leq 1/8\,(64*2^{P-1}) = (2^3 * 2^7) = 2^{P+2};$$

Uwzględniając znak są potrzebne P + 3 bity, by reprezentować dowolną wartość współczynnika DCT. Dlatego po zakończeniu transformacji kosinusowej otrzymujemy macierz DCT (8x8), która wymaga większej pamięci niż pierwotna macierz próbek; w macierzy 2D-DCT każdy element wymaga użycia 11 bitów, podczas gdy każda próbka wymagała użycia tylko 8 bitów.
Dlatego trzeba zastosować jakieś działanie w celu obniżenia wymagań dotyczących pamięci. Tym działaniem jest kwantyzacja elementów transformaty. Kwantyzacja polega na dzieleniu każdego ze współczynników transformaty DCT przez odpowiadający mu element tablicy kwantyzacji.

Obliczenie dwuwymiarowej transformaty DCT na podstawie definicji **F(u,v) = T*X*T'** sprowadza się do dwukrotnego mnożenia macierzy NxN.

- ✓ Jeden element przy mnożeniu dwóch macierzy wymaga N mnożeń i N-1 dodawań.
- ✓ Całkowita liczba mnożeń wynosi $2N^3$, a dodawań $2N^2(N-1)$.
- ✓ Dla N=8 mamy 1024 mnożeń oraz 896 dodawań
- ✓ Opracowane są szybkie algorytmy dla jednowymiarowej DCT, które z kolei stosuje się 2N razy (do N wierszy i N kolumn) by obliczyć dwuwymiarowe DCT.
- ✓ Technika grupowania działań trygonometrycznych pozwala osiągnąć 353 mnożeń i 448 dodawań.
- ✓ Technika Avai (opracowana specjalnie dla N=8) daje 206 mnożeń i 484 dodawań.
- ✓ Dwumianowy algorytm wymaga 54 mnożeń i 464 dodawań i 6 arytmetycznych przesunięć.

Do zapisu liczb rzeczywistych w DCT, stosuje się metody "zmiennoprzecinkową", "stałoprzecinkową" (domyślna) i "szybką stałoprzecinkową". **systemyliczbowe.pdf**
wg.(http://pl.wikipedia.org/wiki/Liczba): Liczby rzeczywiste mogą być zapisywane jako:

- ➢ liczby stałoprzecinkowe, kiedy liczba mnożona jest przez pewną ustaloną z góry stałą, po czym zaokrąglana do najbliższej liczby całkowitej i jako taka zapisywana;
- ➢ liczby zmiennoprzecinkowe, gdy stała dobierana jest w zależności od kodowanej liczby, co czyni tę metodę uniwersalną. Liczba zmiennoprzecinkowa jest komputerową reprezentacją liczb rzeczywistych zapisanych w postaci wykładniczej (zwanej też *notacją naukową*). Ze względu na wygodę operowania na takich liczbach przyjmuje się ograniczony zakres na mantysę i cechę. Powoduje to, że reprezentacja jest tylko przybliżona a jedna liczba zmiennoprzecinkowa może reprezentować różne liczby rzeczywiste z pewnego odcinka.

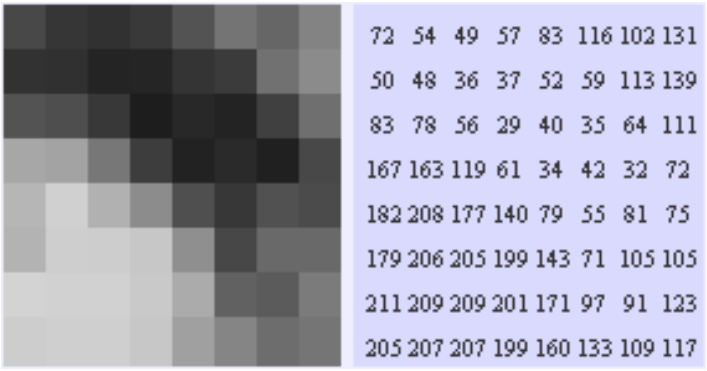
Powszechnie stosuje się zmiennoprzecinkowy zapis liczby rzeczywistej w standardzie IEEE 754.
Znak jest zapisywany jako jeden bit, równy 0 dla + 1 i 1 dla – 1.
Całość zajmuje 4, 8 albo 16 bajtów (w zależności od wymaganej precyzji). Ich kolejność umieszczenia w pamięci jest zależna od procesora.

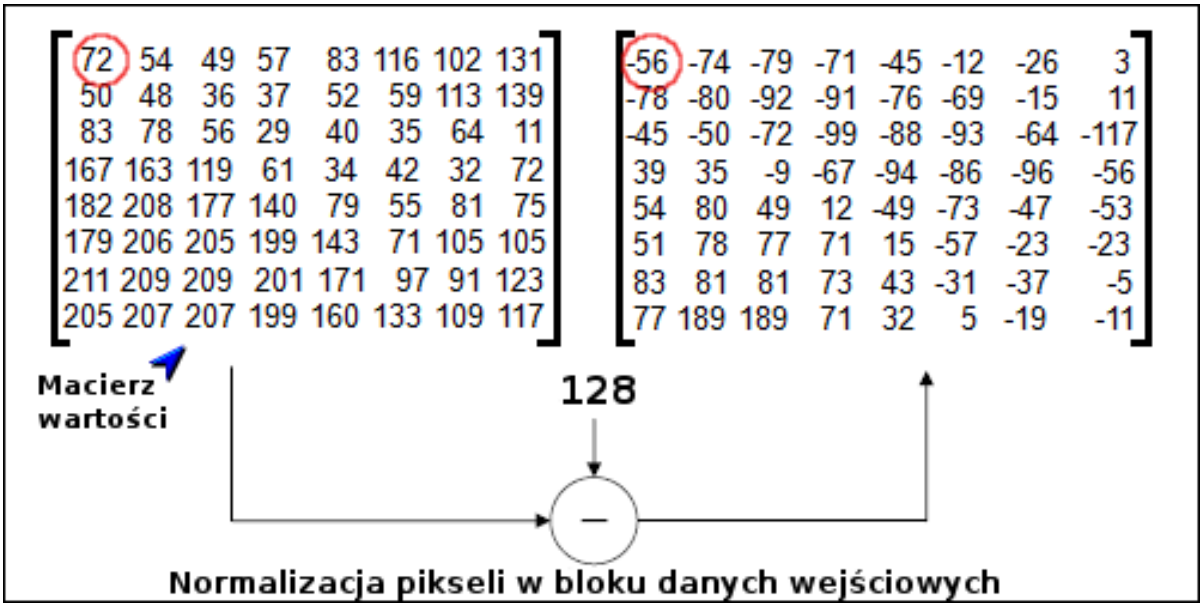
- zmiennoprzecinkowa (**float**): metoda zmiennoprzecinkowa jest nieznacznie bardziej dokładna niż metoda stałoprzecinkowa, lecz jest wiele wolniejsza, chyba że nasz komputer posiada bardzo szybki koprocesor zmiennoprzecinkowy. Metoda zmiennoprzecinkowa może dawać różne efekty na różnych komputerach, podczas gdy metoda stałoprzecinkowa daje wszędzie takie same efekty.
- stałoprzecinkowa (**integer**): ta metoda jest szybsza od zmiennoprzecinkowej ale nie jest dokładna (domyślna – płynna).
- szybka stałoprzecinkowa (**fast integer**): ta metoda jest mniej dokładna od dwóch powyższych (ale szybka).

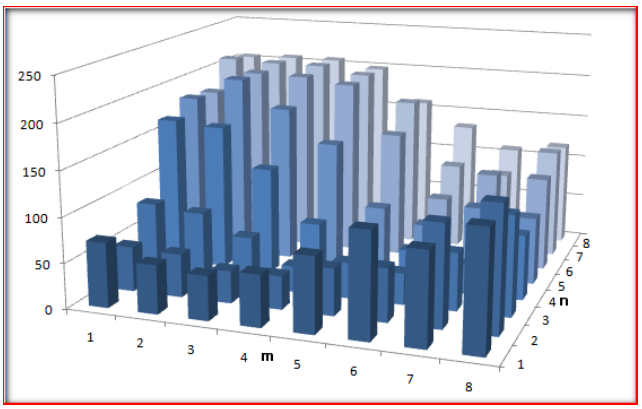
Dla zainteresowanych, więcej szczegółów np.:

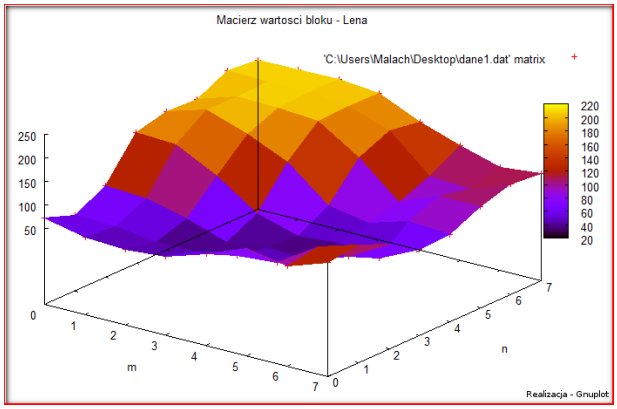
1. http://deuter.am.put.poznan.pl/zwm/eskrypty_pliki/podstawyinformatyki/systemyliczbowe.pdf
2. http://monika.univ.gda.pl/~matrf/studenci2/ak_2004_2005/lab12/koproc.htm
3. http://pl.wikipedia.org/wiki/Liczba_zmiennoprzecinkowa
4. http://pl.wikipedia.org/wiki/Kod_stałopozycyjny

Wiemy już, co to jest transformata jedno i dwuwymiarowa. Przejdziemy więc dalej do przykładu zastosowania w bloku wzorcowego obrazu j/w Lena:

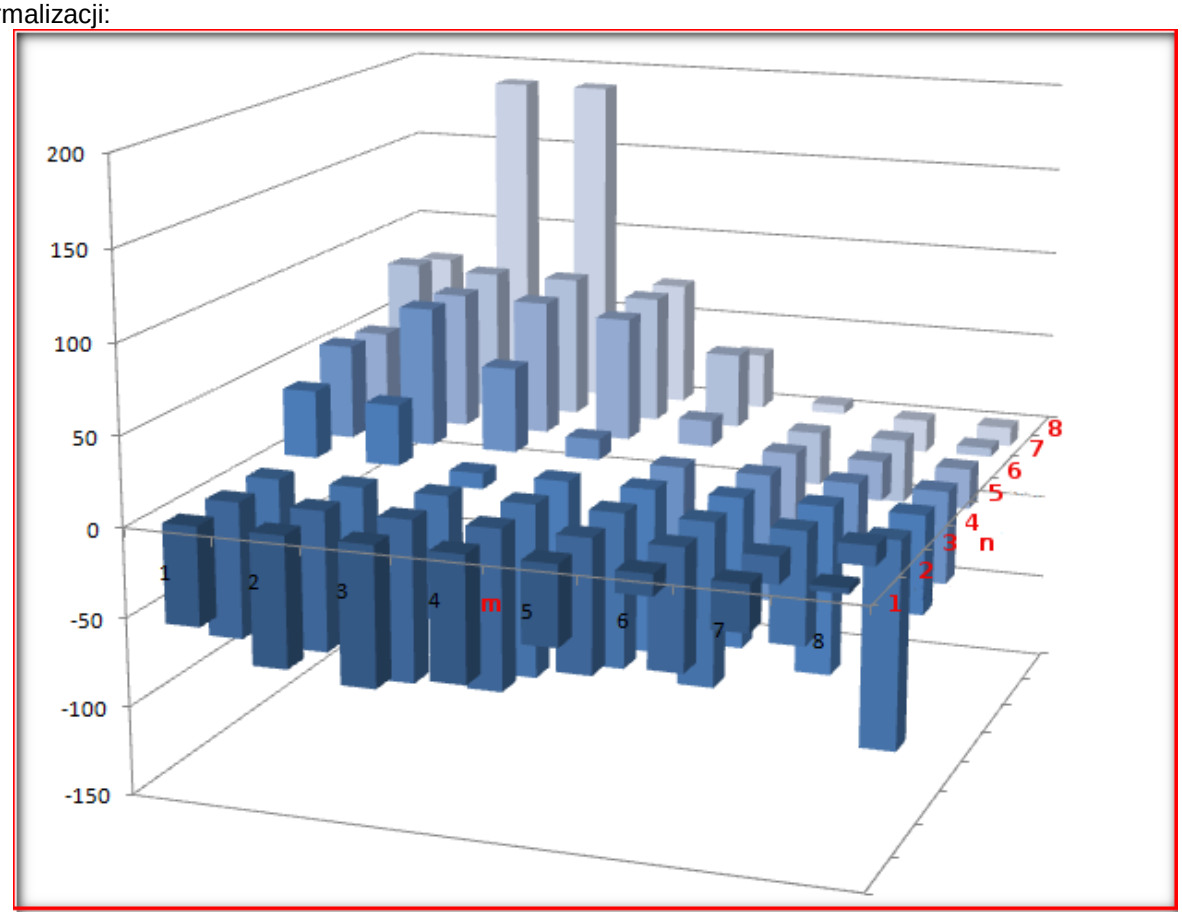


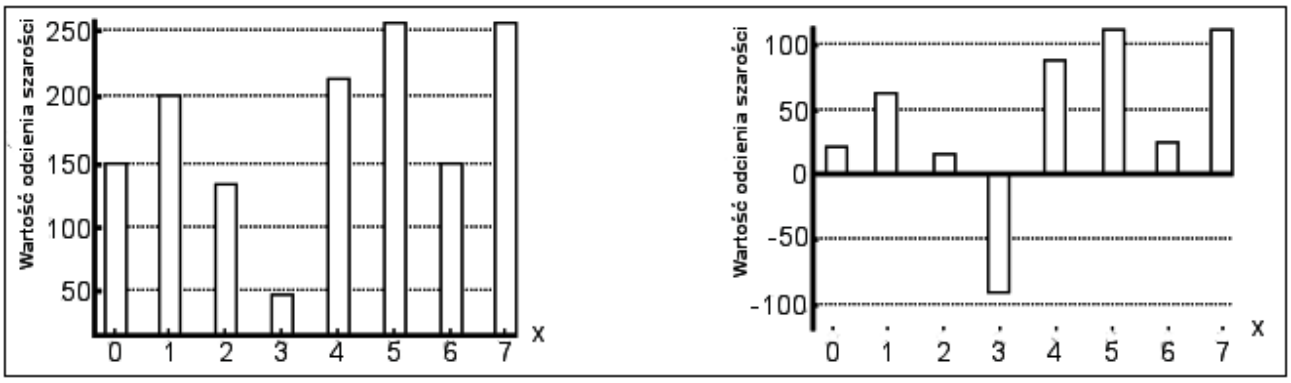


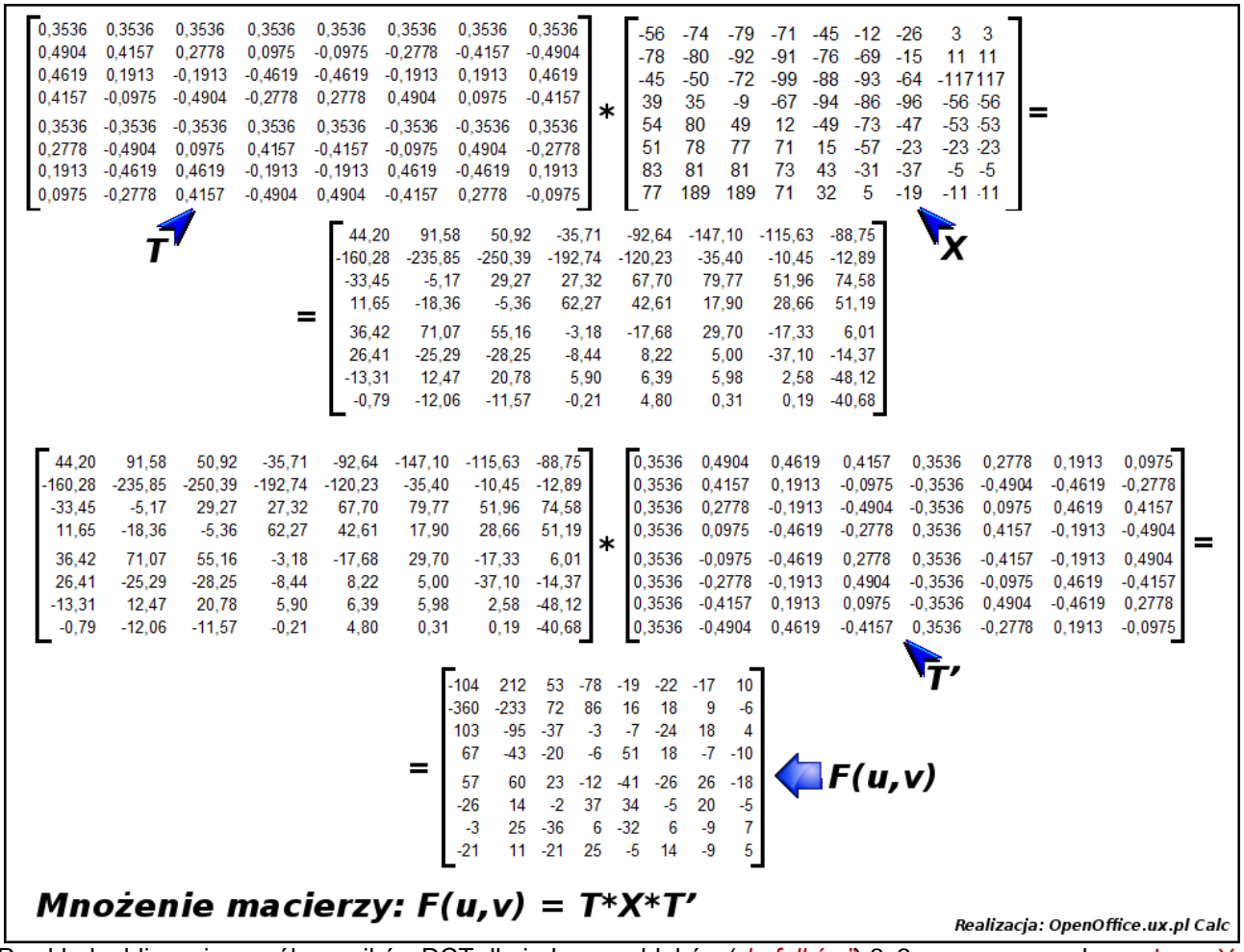




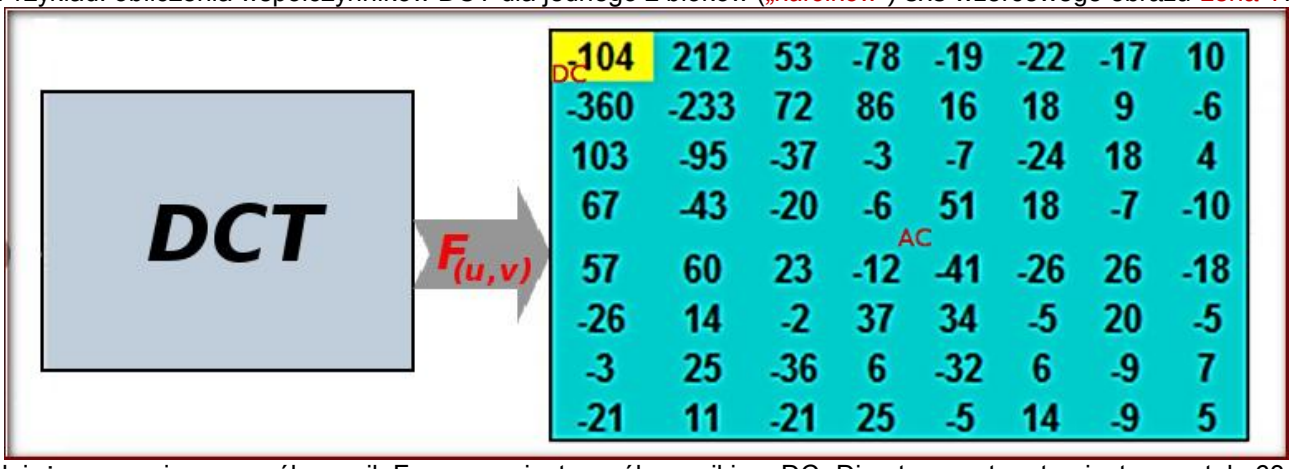
Po normalizacji:







Przykład: obliczenia współczynników DCT dla jednego z bloków (*„kafelków”*) 8x8 wzorcowego obrazu Lena Y.



Jak już wspomniano, współczynnik $F_{0,0}$ zwany jest współczynnikiem DC -Direct current, natomiast pozostałe 63 współczynniki nazywane są współczynnikami ***AC*** - Alternating current.
Ponieważ dane wejściowe były 8 – bitowe, więc nie wystąpiła na razie żadna kompresja danych, do tego jest potrzebny kolejny krok

Kwantyzacja.
Po transformacji otrzymaliśmy macierz wartości rzeczywistych, którą poddamy kwantyzacji. Kwantyzacja, czyli zastąpienie danych zmiennoprzecinkowych przez liczby całkowite. Część danych w ten sposób tracimy – dlatego kompresja JPEG jest *kompresją stratną*. http://en.wikipedia.org/wiki/Quantization_(image_processing)
http://en.wikipedia.org/wiki/Quantization_matrix#Quantization_matrices
Otrzymana w wcześniejszym kroku macierz 64 współczynników transformaty F(u,v) w bloku Lena:

-103,65	211,90	52,51	-77,86	-18,63	-21,71	-17,29	9,80
-360,05	-232,77	72,13	86,28	16,25	18,48	8,50	-5,57
103,25	-94,69	-36,80	-3,35	-6,96	-23,71	18,45	4,15
67,39	-43,48	-19,84	-5,91	51,23	17,99	-7,01	-10,11
56,64	60,15	23,28	-12,49	-41,39	-26,39	26,48	-18,12
-26,10	14,05	-1,82	36,73	34,46	-4,63	20,43	-4,96
-2,59	25,24	-36,29	6,38	-32,16	6,06	-8,69	7,04
-21,22	10,68	-21,39	24,99	-4,87	13,85	-8,53	4,82

poddane zostaną teraz kwantyzacji liniowej przy użyciu macierzy używanej do kwantyzacji współczynników Lumy (luminancji) Q(u,v) o postaci:

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

Tabela kwantyzacji dla współczynników luminancji - Q_{50} *[wg. ITU-T Recommendation T.81*]

Tabela kwantyzacji jest parametrem algorytmu. Komitet JPEG przygotował tabele standardowe dobrane na podstawie eksperymentów z rzeczywistymi obrazami oraz rzeczywistymi urządzeniami, inne dla luminancji inne dla chrominancji. W standardzie JPEG [*ITU-T Recommendation T.81*] zamieszczone są 4 tablice współczynników kwantyzacji, jednakże żadna z nich nie jest obowiązkowa, czyli każda implementacja musi zdefiniować taką tabelę, biorąc pod uwagę parametry urządzenia i specyfikę przetwarzanego obrazu!.
Jak widać największe wartości w macierzy kwantyzacji mają współczynniki poniżej przekątnej (prawy dolny róg) i odpowiadają one współczynnikom macierzy F(u,v) o wyższych częstotliwościach.
Kwantyzacja wykorzystuje fakt, że ludzkie oko jest bardziej wrażliwe na niskie niż wysokie częstotliwości (mała wrażliwość na amplitudę szybkich zmian jasności) i poprzez kwantyzację, która polega na dzieleniu każdego ze współczynników transformaty F(u,v) przez odpowiadający mu element tabeli kwantyzacji Q(u,v) i zaokrągleniu wyniku do najbliższej liczby całkowitej.
Typowo większość współczynników AC transformaty Fu,v ma małe wartości, tym mniejsze im wyższej częstotliwości odpowiadają (małe znaczenie dla percepcji obrazu), po kwantyzacji ulegają zaokrągleniu do zera, co jest równoznaczne z *wymazaniem* wyższych częstotliwości.
Tablica kwantyzacji liniowej ma wymiar równy wymiarowi bloku 8x8 i zawiera dla każdego współczynnika transformacji wartość progu i przedział kwantyzacji. Tę samą tablicę stosujemy do wszystkich bloków.
Kwantyzacja wyraża się wzorem:

$$F^{Q}(u,v) = \mathbf{round}\left[\frac{F(u,v)}{Q(u,v)}\right]$$

Gdzie:
F(*u,v*) - współczynniki transformacji,
Q(*u,v*) — tablica kwantyzacji,
round [x] - funkcja zaokrąglająca **[x]** do najbliższej liczby całkowitej większej bądź równej liczbie rzeczywistej **[x].**

Dzieląc wszystkie współczynniki Fu,v przez stałą wartość Q(u,v) kontrolujemy współczynnik kompresji, po zaokrągleniu wyniku do najbliższej liczby całkowitej.

$F_{(u,v)}$

-104 (DC)	212	53	-78	-19	-22	-17	10
-360	-233	72	86	16	18	9	-6
103	-95	-37	-3	-7	-24	18	4
67	-43	-20	-6	51	18	-7	-10
57	60	23	-12 (AC)	-41	-26	26	-18
-26	14	-2	37	34	-5	20	-5
-3	25	-36	6	-32	6	-9	7
-21	11	-21	25	-5	14	-9	5

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

$$F^Q(u,v) = \textbf{round}\left[\frac{F(u,v)}{Q(u,v)}\right]$$

$F^Q(u,v) =$

-7	19	5	-5	-1	0	0	0
-30	-19	5	5	1	0	0	0
7	-7	-2	0	0	0	0	0
5	-3	-1	0	1	0	0	0
3	3	1	0	-1	0	0	0
-1	0	0	1	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0

Tak więc nasza macierz transformaty F(u,v) bloku *Lena* będzie po kwantyzacji dla luminancji - Q_{50}

-7	19	5	-5	-1	0	0	0
-30	-19	5	5	1	0	0	0
7	-7	-2	0	0	0	0	0
5	-3	-1	0	1	0	0	0
3	3	1	0	-1	0	0	0
-1	0	0	1	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0

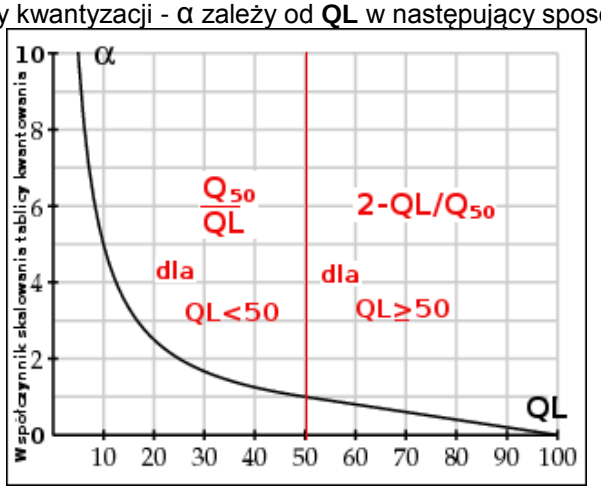
Poniżej zamieszczam **tabelę wg**. ***ITU-T Recommendation T.81*** **do kwantyzacji współczynników chrominancji**.
Współczynniki w tablicach podanych przez komitet JPEG zostały wyznaczone na podstawie eksperymentów wykonanych z różnymi obrazami i urządzeniami i psychowizualnej oceny testowych obrazów.

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

Tabela kwantyzacji dla współczynników chrominancji *[wg. ITU-T Recommendation T.81*].
Jak widać współczynniki w tabeli dla kwantyzacji chrominancji są większe i co za tym idzie zdecydowana większość składowych C_r C_b po kwantyzacji będzie równa zero i zostanie usunięta. Natomiast po kwantyzacji luminancji składowych Y zostanie więcej. Jak już wspomniano, oko ludzkie jest bardziej wyczulone na zmianę jasności (luminancji) niż na zmianę barwy (chrominancji). Dzięki temu taka kwantyzacja będzie bardziej skuteczna. Gdy kodujemy obrazy czarno-białe, używamy jedynie składowej luminancji.
Główną zaletą takiej metody kwantyzacji jest szybkość działania - wykonywane jest tylko dzielenie. Ponadto jest ona obarczona niewielkim błędem oraz pozwala ograniczyć występowanie efektu blokowego (ta sama tablica kwantyzacji do wszystkich bloków).

W przypadku kompresji JPEG, kiedy tworzymy plik, algorytm prosi o parametr kontroli jakości obrazu czyli w jakim stopniu obraz ma być skompresowany np.:

Możemy ustalić tak zwany poziom jakości (Quality lewel), oznacza się go jako QL. Jest to współczynnik o wartościach całkowitych od 1 do 100, określający jakość zdekompresowanego obrazu oraz o stopniu kompresji wyjściowego obrazu. Wyższa wartość QL odpowiada wyższej jakości obrazu i większym rozmiarem pliku. QL równy 1 oznacza najwyższy współczynnik kompresji obrazu ale najniższą jakość zdekompresowanego obrazu. Podana powyżej „standardowa" macierz kwantyzacji oznaczana Q_{50} odpowiada współczynnikowi QL = 50, stanowi pewne wyważenie pomiędzy jakością a współczynnikiem kompresji.
Współczynnik skalowania tablicy kwantyzacji - α zależy od **QL** w następujący sposób:



Z wykresu wynika, że gdy chcemy skompresować obraz przy użyciu innej wartości współczynnika QL, to:

- dla QL ≥ 50 mnożymy współczynniki macierzy Q_{50} przez współczynnik skalowania 2 - QL/Q_{50} (wartość od 1 do 0,02 – QL=99)
- dla QL < 50 mnożymy współczynniki macierzy Q_{50} przez współczynnik Q_{50}/QL (wartość od 1 do 10)

(wykres zakończono dla wartości współczynnika skalowania α równego 10, gdy **QL = 5** !)
Po wymnożeniu, współczynniki nowej macierzy kwantyzacji są zaokrąglane i obcinane, aby ich wartości były liczbami całkowitymi z przedziału od 1 do 255.
Przykładowo współczynnikowi QL = 90 odpowiada macierz:

$$Q_{90} = \begin{bmatrix} 3 & 2 & 2 & 3 & 5 & 8 & 10 & 12 \\ 2 & 2 & 3 & 4 & 5 & 12 & 12 & 11 \\ 3 & 3 & 3 & 5 & 8 & 11 & 14 & 11 \\ 3 & 3 & 4 & 6 & 10 & 17 & 16 & 12 \\ 4 & 4 & 7 & 11 & 14 & 22 & 21 & 15 \\ 5 & 7 & 11 & 13 & 16 & 12 & 23 & 18 \\ 10 & 13 & 16 & 17 & 21 & 24 & 24 & 21 \\ 14 & 18 & 19 & 20 & 22 & 20 & 20 & 20 \end{bmatrix}$$

Gdzie: $Q_{90} = 0{,}2*Q_{50}$ dla pierwszego współczynnika mamy: 0,2*16 = 3,2 ~ 3 itd., gwarantuje nam ona lepszą jakość obrazu zdekompresowanego, ale gorszy współczynnik kompresji.
Współczynnikowi QL = 10 odpowiada macierz:

$$Q_{10} = \begin{bmatrix} 80 & 60 & 50 & 80 & 120 & 200 & 255 & 255 \\ 55 & 60 & 70 & 95 & 130 & 255 & 255 & 255 \\ 70 & 65 & 80 & 120 & 200 & 255 & 255 & 255 \\ 70 & 85 & 110 & 145 & 255 & 255 & 255 & 255 \\ 90 & 110 & 185 & 255 & 255 & 255 & 255 & 255 \\ 120 & 175 & 255 & 255 & 255 & 255 & 255 & 255 \\ 245 & 255 & 255 & 255 & 255 & 255 & 255 & 255 \\ 255 & 255 & 255 & 255 & 255 & 255 & 255 & 255 \end{bmatrix}$$

Kwantyzacja dla różnych wartości QL wyraża się teraz wzorem:

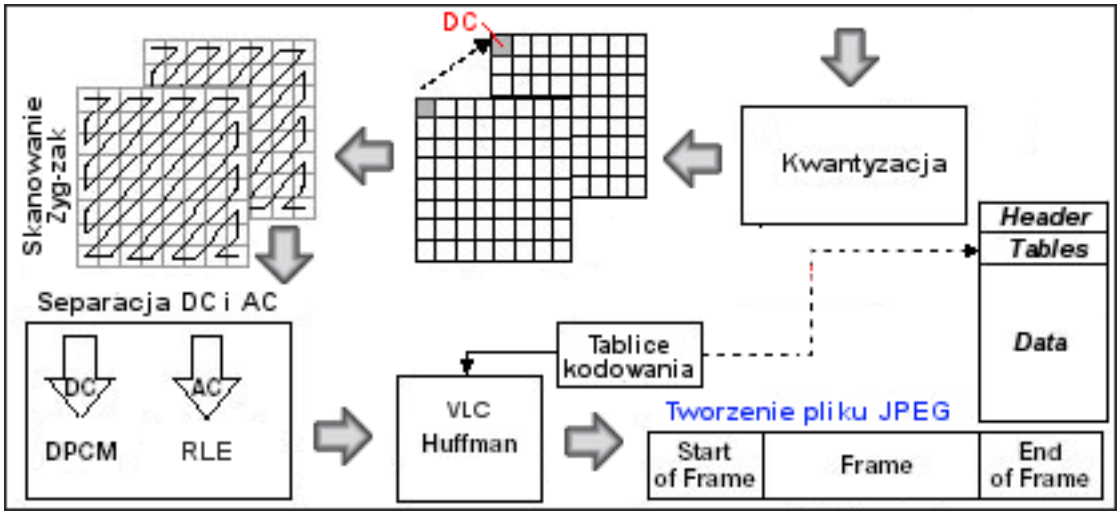
$$F^{Q}(u,v) = \textbf{round}\left[\frac{F(u,v)}{Q(u,v) * \alpha}\right]$$

Jak już wspomniano po zakończeniu transformacji kosinusowej otrzymano macierz DCT, która wymagała większej pamięci niż pierwotna macierz próbek; a mianowicie każdy element wymagał użycia *P + 3 = 11* bitów, podczas gdy każda próbka wymagała użycia tylko 8 bitów. Pojawia się więc pytanie, co zyskaliśmy po zastosowaniu kwantyzacji?. Po kwantyzacji liczba bitów zmniejsza się do P +3 – $\log_2$ (Q(u,v), czyli w przypadku tablicy dla luminancji gdzie najmniejszy współczynnik kwantyzacji = 10, wystarczy liczba
P + 3 – [log₂ 10] = P bitów do zakodowania jednej danej wynikowej kwantyzatora.

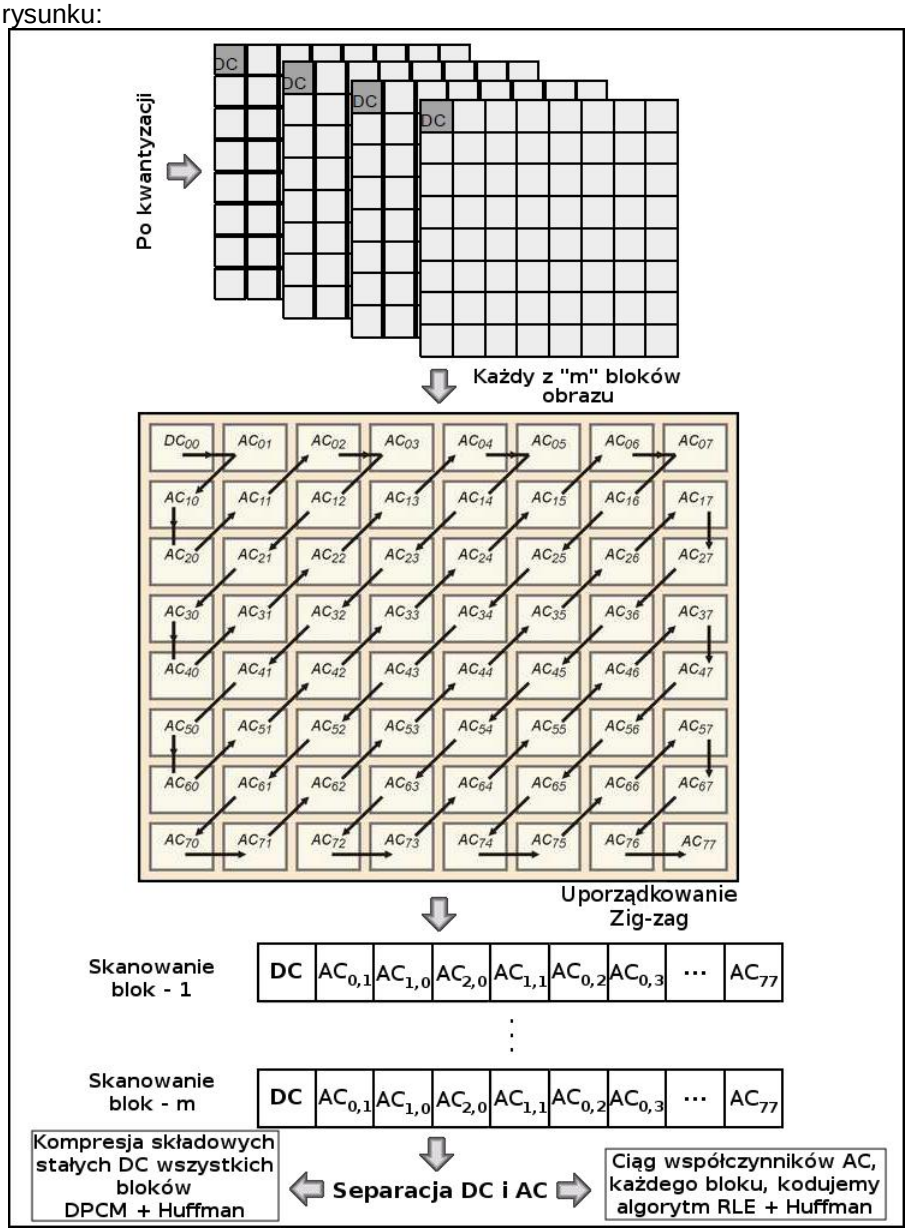
Właściwe w tym miejscu można by zakończyć szczegóły opisu algorytmu systemu bazowego (Baseline) JPEG, jeżeli weźmiemy pod uwagę, że użytkownik nie ma żadnego wpływu w zakresie dalszego przebiegu algorytmu.

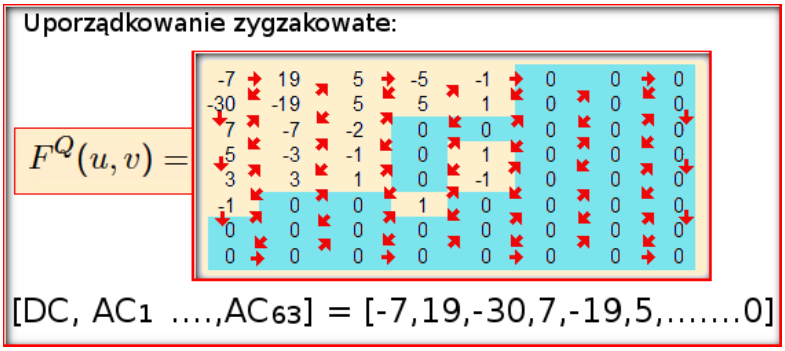
Ale dla zamknięcia tematu trzeba jeszcze omówić istotne końcowe etapy:
Kodowanie długości ciagów (*run-length encoding - RLE*) oraz
Kodowanie algorytmem Huffmana.
Które mają na celu zmniejszenie objętości macierzy. Pozwalają na bezstratne zmniejszenie ilości informacji przy akceptowalnym poziomie strat, a polega to na przypisaniu poszczególnym powtarzającym się ciągom pewnych kodów. Wartościom występującym często, a wiec zajmującym najwięcej miejsca przypisywane są krótkie kody, a występującym rzadko - długie. Kody określone są w specjalnej tabeli służącej później do dekodowania.

Przestawienie zygzakowate macierzy DCT i kodowanie Huffmana.



W tym kroku skwantowane współczynniki transformaty są układane za pomocą tak zwanego uporządkowania **Zyg-zak**. Następuje konwersja macierzy (tablicy) współczynników do postaci wektora. Czyli zmiana kolejności 64 składników DCT, algorytmem zig-zag (opisany przez A.G. Teschera w 1978). Współczynniki transformaty są grupowane jak na rysunku:





Link do animacji: http://nptel.iitm.ac.in/courses/Webcourse-contents/IIT%20Kharagpur/Multimedia%20Processing/animations/fig%209.8.swf

Dzięki zastosowaniu **ZigZak**, skwantowana tablica dwuwymiarowa *F*(*u, v*) przekształcana jest w tablicę jednowymiarową (wektor).

Jak widać na rysunkach, dane zostają kolejno, zygzakiem od lewego-górnego rogu macierzy do dolnego-prawego, przepisywane do 64-elementowego wektora.

Dzięki zastosowaniu takiej reorganizacji tablicy, wartości niezerowe w wektorze występują koło siebie. Tym samym zera tworzą dłuższy ciąg niż przy normalnym obejściu tablicy. Uzyskuje się ciąg (wektor) rozpoczynający się współczynnikiem **DC**, po którym następują 63 współczynniki **AC**.

W takim ciągu często po kilka sąsiednich współczynników AC (podciągów) ma te same wartości np. zero. Możemy wtedy zastosować kodowanie długości serii, polegające na generowaniu słów kodowych dla serii takich samych symboli, za pomocą kodowania **RLE -** (***Run Length Encoding* –** *kodowanie długości serii)*. Uzyskana reprezentacja składa się z par typu (**RRRR** – **liczba powtórzeń, Wartość**) wyznaczonych dla kolejnych fragmentów ciągu utworzonych przez współczynniki o takich samych wartościach.

Kodowanie ***RLE***, ma jednak sporą nadmiarowość – wszystkie wartości pikseli są traktowane w ten sam sposób, mimo iż niektóre z nich występują częściej niż inne i w związku z tym zasługują na większą uwagę.

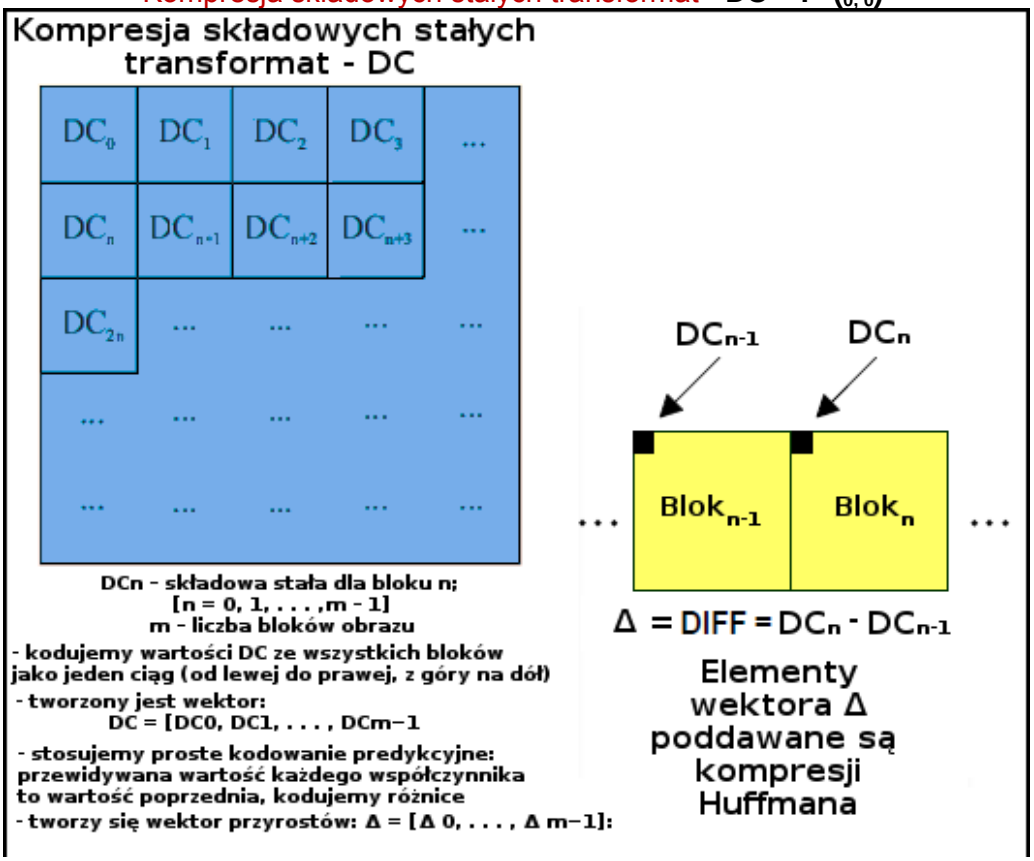
Legło to u podstaw sposobu kodowania znanego jako *kompresja Huffmana* (*Huffman encoding*).

Liczby określające kod RLE są kodowane ze zmienną długością słowa (**VLC ang. *Variable Length Coding***), za pomocą kodów Huffmana.

W **systemie bazowym JPEG** metodą Huffmana do kodowania wszystkich współczynników stosujemy dwa zestawy tabel, jeden do współczynników **DC**, a drugi do kodowania współczynników **AC**.

Bezstratna (entropowa) kompresja współczynników transformaty

Kompresja składowych stałych transformat - **DC = $F^Q_{(0,0)}$**



DPCM (ang. *Differential Pulse Code Modulation*) **współczynników DC, to** = różnica dwóch współczynników DC, dwóch kolejnych bloków, kodowanie binarne sekwencji elementów *DC* wykonuje się techniką predykcji* liniowej rzędu jeden. Dla pierwszego bloku jako predykcję wartości DC przyjmujemy 0.

Przykład: Jeśli współczynniki DC kolejnych bloków wynoszą np.:

$$DC = [DC_0, DC_1, \ldots, DC_{m-1}]$$

DC = **[13,13,10,11,11,10….]**

Wartości wektora przyrostów DPCM wyniosą:

***diff* = Δ = [Δ $_0$, . . . , Δ $_{m-1}$]:**

***Gdzie:* Δ $_0$ = DC_0, Δ n = $DC_n - DC_{n-1}$, n = 1, 2, . . . ,m – 1, m – liczba bloków obrazu**

DC_{n-1} - predyktor

Różnica *diff* jest zazwyczaj mała, co umożliwia efektywne stosowanie do kodowania metody DPCM.
Różnica ***diff*** zostaje zakodowana jako kategoria, za pomocą tabeli kodów Huffmana [dla **bazowego JPEG** tabela zmiennej ***diff*** ma 12 wierszy; (w systemie sekwencyjnym rozszerzonym tabela ta składa się z 16 wierszy)].
Dla powyższego ciągu współczynników, otrzymamy następujące *diff*: **13, 0, -3, + 1, 0, -1, …**

Kod DPCM wyrażany jest przez reprezentację składającą się z par typu: (**SSSS**, **wartość**):

SSSS [*Category, Length*] – wskazuje kategorię ***diff*** (odpowiada jednej z 12 kategorii, w tabeli wartości różnic ***diff***.)
SSSS jest używany jako indeks do odszukania w tablicy Huffmana słowa kodu, odpowiadającego różnicy *diff*, przy czym słowo kodu zależy od bieżąco używanej tabeli Huffmana.
Komitet JPEG podał przykłady tego typu tabel jednej dla **luminancji** i jednej dla **chrominancji**.
Słowo kodu Huffmana składa się z co najmniej dwóch bitów i najwyżej 16 bitów; żadne ze słów kodu nie składa się tylko z jedynek.
Słowa kodu odpowiadają kategoriom wartości ***diff***, a nie poszczególnym abs wartościom *diff***.**

Wartość jest liczbą w kodzie binarnym (dwójkowym), reprezentującą znak i wartość (**najmniej znaczących bitów**) liczby dziesiętnej współczynnika *diff*;

- dla wartości dodatnich **Wartość** reprezentuje binarne wartości *diff*;
- dla wartości ujemnych **Wartość** jest reprezentowana binarnie przez dołączenie najmniej znaczących bitów w zapisie uzupełnień do dwóch liczby **(*diff* – 1)**.

Bity reprezentujące wartość dodatnią *diff* zaczynają się od **1**, a reprezentujące wartości ujemne od **0** i nie ma potrzeby dołączania dodatkowego bitu wskazującego na znak liczby, gdyż jest on częścią reprezentacji wartości *diff*.
Przykłady:
Liczba $\mathbf{13_{10}}$ jest równa liczbie binarnej $\mathbf{1101_2}$,
Liczbie dodatniej **12** odpowiada w kodzie binarnym zapis **1100**, natomiast liczba **-12** w zapisie uzupełnień do dwóch **(*diff* – 1) = (-12-1) = - 13** to **11111110011**, ale cztery najmniej znaczące bity tej liczby, to **0011**.
Trik: jak z powyższego widać, w przypadku liczb ujemnych zapisujemy ich negację z zapisu wartości dodatnich.

Z tablicy **SSSS** wynika, że liczbie **-35** odpowiada kategoria **6**. Kod binarny **35** jest **100011**, jego odwrotnością jest kod **011100**, to znaczy że liczbie **-35** odpowiada **kod DPCM {6, 011100}**. Słowo kodu Huffmana dla indeksu **6** jest **1110**, stąd **-35** zakodujemy 10 bitowym słowem kodowym 1110011100.

Liczba **7** w kodzie binarnym ma zapis **111**, natomiast w zapisie uzupełnień do dwóch, cztery najmniej znaczące bity liczby -7 to **000**.

SSSS (Kategoria)	Wartość *diff* lub współczynnik AC
0	0
1	-1 , 1
2	-3,-2,2,3
3	-7,...,-4,4,...7
4	-15,...,-8,8,...15
5	-31,...-16,16,...31
6	-63,...-32,32,...63
7	-127,...-64,64,...127
8	-255,...-128,128,...255
9	-511,...-256,256,...511
10	-1023,...-512,512,...1023
11	-2047,...-1024,1024,...2047

Wg: ISO/IEC 10918-1 : 1993(E) Table F.1 – Difference magnitude categories for DC coding
ISO/IEC 10918-1 : 1993(E) Table F.2 – Categories assigned to coefficient values AC (str 89)

Uzasadnienie: większość wartości będzie bliskich zeru

Jeśli współczynniki DC kolejnych bloków wynoszą np.:

DC = [13,13,10,11,11,10....]

Wartości wektora przyrostów DPCM wyniosą:

***diff:* 13, 0, -3, + 1, 0, -1, ...**

Liczba 13_{10} jest równa liczbie binarnej 1101_2

Liczba dziesiętna $3 = 11_2$ odwrotnością jest kod 00

Wartości *diff*	*Kod DPCM*	
	SSSS	*Wartość*
13	4	1101
0	0	
-3	2	00
1	1	1
0	0	
-1	1	0

Wartości DPCM dla Luminancji

Kategoria	Długość słowa	Słowo kodu
0	2	00
1	3	010
2	3	011
3	3	100
4	3	101
5	3	110
6	4	1110
7	5	11110
8	6	111110
9	7	1111110
10	8	11111110
11	9	111111110

ISO/IEC 10918-1 : 1993(E Table K.3

Tabela VLC Huffmana dla luminancji różnic współczynników DC

W przypadku kodowania bloków chrominancje korzystamy z poniższej tabeli:

Kategoria	Długość słowa	Słowo kodu
0	2	00
1	2	01
2	2	10
3	3	110
4	4	1110
5	5	11110
6	6	111110
7	7	1111110
8	8	11111110
9	9	111111110
10	10	1111111110
11	11	11111111110

ISO/IEC 10918-1 : 1993(E)- Table K.4

Tabela VLC Huffmana dla chrominancji różnic współczynników DC

Wartości *diff*	*SSSS*	*Kod Huffmana*	*Wartość*	*Zakodowane bity*
13	4	101	1101	1011101
0	0	00		00
-3	2	011	00	01100
1	1	010	1	0101
0	0	00		00
-1	1	010	0	0100

Kod DPCM zakodowany VLC Huffman`a

W powyższym przykładzie otrzymaliśmy bity kodu VLC dla poszczególnych kategorii i wartości ***diff***. Najdłuższe jest 7 bitowe słowo kodowe 1011101.

Widać, że podział wartości *diff* na kategorie umożliwia osiągnięcie dużej kompresji, ponieważ małym wartościom *diff* odpowiadają krótkie słowa kodu, o wiele krótsze niż 11 bitów używanych poprzednio do kodowania współczynników AC i wartości DC.

W naszym przykładzie mamy *diff-y* **6-ciu** kolejnych bloków DC zakodowanych w następujący ciąg bitów:

101110100011000101000100;

składa się on z 24 bitów, czyli mniej niż 11 * 6 = 66 bitów potrzebnych do zakodowania nieskwantowanych współczynników DC (bez kompresji)

lub 8 * 6 = 48 bitów jakie były by potrzebne na zakodowanie 6 skwantowanych współczynników DC.

Jeśli piksele powtarzają się długimi fragmentami, to można osiągnąć prawie 64-krotną redukcję, gdyż w najlepszym razie zamiast 128 bajtów wstawiamy 2 - *znacznik oraz wartość*.

Kompresja składowych zmiennych transformat – elementów AC

Proces kodowania elementów AC, można podzielić na dwie fazy.

- W fazie pierwszej ciąg współczynników uzyskany po kwantyzacji i uporządkowaniu *zygzak*, jest dzielony na **sekwencje zer zakończonych elementem niezerowym** za pomocą kodowania **RLE –** (*run length encoding* **–** *kodowanie długości serii*).
- 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*.
  W systemie bazowym JPEG stosujemy przykładowe tablice kodowe Huffmana, do kodowania wszystkich współczynników, używając dwóch zestawów tabel, które podano w opisie standardu JPEG **ISO/IEC 10918-1 : 1993(E)** *jeden do kodowania współczynników DC i drugi do kodowania pozostałych współczynników.*

Zgodnie z w/w normą długie ciągi zer w ZZ są kodowane za pomocą metody **RLE.**
Kod prefiksowy służący do kodowania długości serii zer tworzy bajt w postaci:

RS = binarny RRRRSSSS

w którym 4 najmniej znaczące bity **LSB** (Least Significant Bit) **SSSS** (od **Size** rozmiar**)** - są kategorią niezerowych współczynników AC - ilość bitów potrzebna do zakodowania $\mathbf{AC_i}$.
natomiast 4 najbardziej znaczące bity **MSB** (Most Significant Bit) **RRRR** (od **Run-length** liczba powtórzeń) definiują długość ciągu zer pomiędzy współczynnikiem

$\mathbf{AC_i}$ i poprzednim niezerowym $\mathbf{AC_{i-1}}$,

W celu określenia kategorii **SSSS** dla AC - stosujemy tablicę identyczną jak dla DC (dla **bazowego JPEG** - ale **pomijamy wiersz pierwszy i ostatni - mamy 10 kategorii AC**).
Stosuje się dwa specjalne symbole.
Pierwszym specjalnym symbolem jest **ZRL** (*Zero Run Length*), używany wówczas, gdy ciąg zer składa się z więcej niż 15 zer; wtedy bajt **RRRRSSSS** równa się liczbie binarnej 11110000, oznaczającej 16 zer (w tablicy w zapisie szesnastkowym oznaczone jako F/0 = 15 zer + 0 = 16 zer).
Następnym specjalnym symbolem jest **EOB** (End-of-Block), oznaczający same zera za ostatnią zakodowaną wartością (koniec bloku, dalej same zera); symbolu EOB nie włącza się do ciągu słów kodu, gdy ostatni 63 współczynnik ZZ jest różny od zera, wtedy bajt RRRRSSSS jest równy liczbie binarnej 00000000.

Przykład: mamy np. ciąg **63** współczynników AC

6,7,0,0,0,3,-1,0,0,0,0,0,0,-12,0,…,0.

Po zastosowaniu kodowania długości serii, **otrzymamy współczynniki AC w słowach kodu RLE:**

(0,6),(0,7),(3,3),(0,-1),(6,-12),(0,0)

- gdzie np. para **(6,-12)** oznacza ciąg 6 zer zakończony niezerową liczbą **-12**;
- ostatni **(0,0)** =0/0 (**EOB**) wskazuje koniec ciągu dla tego bloku. (współczynniki pozostałych AC wszystkie są zerami)

Ponieważ przy kodowaniu par AC na **pierwszy element pary** mamy do dyspozycji **4 bity,** to jeśli występuje więcej zer niż 15 np. 22, kodujemy parę jako dwie pary: (15,0)=15 zer + 0 = 16 zer i (6, $\mathbf{AC_i}$) = ciąg 6 zer zakończony niezerową liczbą $\mathbf{AC_i}$**.**

Słowo kodu **RRRRSSSS** oznaczające dany ciąg zer nie jest dalej przesyłane. Używamy go podobnie jak przy kodowaniu DC jako indeksu umożliwiającego odszukanie w tabeli VLC Huffmana słowa kodu dla każdej pary RRRR/SSSS. Dla systemu bazowego JPEG mamy 4 tabele dla AC luminancji i 4 tabele dla AC chrominancji, poniżej przykład pierwszych.
Podobnie jak dla DC mamy możliwość jednoznacznego kodowania również współczynników AC, dokonuje się tego w ten sam sposób jak w przypadku kodowania ***diff***.
Mianowicie do słowa kodu VLC Huffmana odpowiadającego bajtowi RRRRSSSS dołącza się dodatkowe bity kodujące **znak i wartość** niezerowego współczynnika AC występującego po ciągu zer.
Dodatkowe bity są liczbą (w kodzie binarnym - dwójkowym), reprezentującą znak i wartość (**najmniej znaczących bitów**) liczby dziesiętnej współczynnika AC;

- dla wartości dodatnich **Wartość** reprezentuje najmniej znaczące bity współczynnika AC;
- dla wartości ujemnych **Wartość** jest reprezentowana przez dołączenie najmniej znaczących bitów w zapisie uzupełnień do dwóch (w skrócie - U2) liczby **(*współczynnik AC* – 1)**.

Przykłady:

1. słowo kodu RLE (**2,-4**), ilość zer **2**; dla (**- 4**) w tabeli kategorii znajdujemy, że to kategoria **3**;
   RRRR/SSSS = **2/3**; w tabeli VLC Huffmana szukamy słowa kodu dla 2/3 mamy: **1111110111**;
   kodujemy drugi element pary, liczbę **- 4**: liczbie dziesiętnej **4** odpowiada liczba binarna **100**, ponieważ mamy współczynnik o wartości ujemnej, zapisujemy **zanegowaną** reprezentację binarną liczby 4, czyli **011**.
2. słowo kodu RLE (**0,7**), ilość zer 0; dla liczby 7 w tabeli kategorii, znajdujemy, że to kategoria **3**
   RRRR/SSSS = **0/3;** w tabeli VLC Huffmana szukamy słowa kodu dla **0/3**, mamy: **100**;
   teraz szukamy wartość binarną niezerowego współczynnika **7** jest to **111**.
3. Słowo kodu RLE (**7,-12**), ilość zer 7; dla (**- 12**) w tabeli kategorii znajdujemy, że to kategoria **4**;

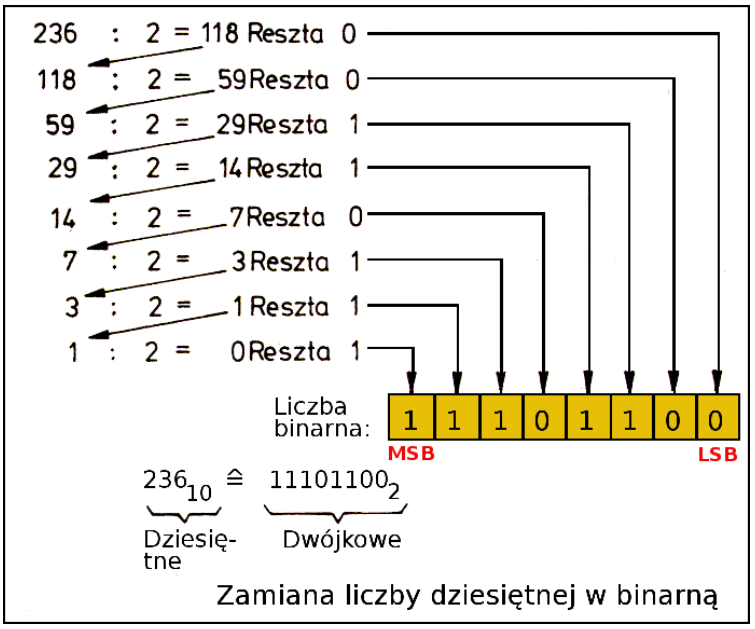
RRRR/SSSS = 7/4; w tabeli VLC Huffmana dla 7/4 mamy słowo kodu: 1111111110101111
kodujemy drugi element pary, liczbę **-12**: liczbie dziesiętnej **12** odpowiada liczba binarna **1100**, ponieważ mamy współczynnik o wartości ujemnej, zapisujemy **zanegowaną** reprezentację binarną liczby **12**, czyli **0011**.

4. słowo kodu RLE (**0,50**) RRRR/SSSS = 0/6 kodujemy jako: z tabeli 1111000; + 50_{10} = 110010_2
5. słowo kodu RLE (**0,20**) RRRR/SSSS = 0/5 kodujemy jako: z tabeli 11010; + 20_{10} = 10100_2
6. Gdy mamy podciąg 0000000000000000 słowo kodu RLE RRRR/SSSS = **15/0** (ZRL) szukamy w tabeli słowa kodu VLC Huffmana – mamy:

Run/Size	Code length	Code word
F/0 (ZRL)	11	11111111001

Dodatkowe bity – brak

Przypomnienie:



Zamiana liczby dziesiętnej w binarną

Mamy przykładowy ciąg współczynników AC w słowach kodu RLE:

(2,-4), (0,7), (3,3), (0,-1), (0,0)

Kod RLE (w formacie binarnym):

(RRRR, SSSS, wartość)

Kod RLE	RRRR	SSSS	Wartość
2,-4	2	3	011
0,7	0	3	111
3,3	3	2	11
0,-1	0	1	0
0,0	0	0	

Kodowanie RLE w słowo kodu VLC Huffmana

Kod RLE	RRRR/SSSS	Słowo kodu Huffmana	Wartość Non-zero
2,-4	2/3	1111110111	011
0,7	0/3	100	111
3,3	3/2	111110111	11
0,-1	0/1	00	0
0,0	0/0		

W rezultacie współczynniki zostają zakodowane (po dołączeniu do słowa kodu VLC wartości Non-zero) w następujący ciąg bitów:

1111110111011 100111 11111011111 000 1010

Tablica K.5 VLC dla współczynników AC luminancji

Run/Size	Code length	Code word	Run/Size	Code length	Code word	Run/Size	Code length	Code word	Run/Size	Code length	Code word
0/0 (EOB)	4	1010	4/1	6	111011	8/1	9	111111000	C/1	10	1111111010
0/1	2	00	4/2	10	1111111000	8/2	15	111111111000000	C/2	16	1111111111011001
0/2	2	01	4/3	16	1111111110010110	8/3	16	1111111110110110	C/3	16	1111111111011010
0/3	3	100	4/4	16	1111111110010111	8/4	16	1111111110110111	C/4	16	1111111111011011
0/4	4	1011	4/5	16	1111111110011000	8/5	16	1111111110111000	C/5	16	1111111111011100
0/5	5	11010	4/6	16	1111111110011001	8/6	16	1111111110111001	C/6	16	1111111111011101
0/6	7	1111000	4/7	16	1111111110011010	8/7	16	1111111110111010	C/7	16	1111111111011110
0/7	8	11111000	4/8	16	1111111110011011	8/8	16	1111111110111011	C/8	16	1111111111011111
0/8	10	1111110110	4/9	16	1111111110011100	8/9	16	1111111110111100	C/9	16	1111111111100000
0/9	16	1111111110000010	4/A	16	1111111110011101	8/A	16	1111111110111101	C/A	16	1111111111100001
0/A	16	1111111110000011	5/1	7	1111010	9/1	9	111111001	D/1	11	11111111000
1/1	4	1100	5/2	11	11111110111	9/2	16	1111111110111110	D/2	16	1111111111100010
1/2	5	11011	5/3	16	1111111110011110	9/3	16	1111111110111111	D/3	16	1111111111100011
1/3	7	1111001	5/4	16	1111111110011111	9/4	16	1111111111000000	D/4	16	1111111111100100
1/4	9	111110110	5/5	16	1111111110100000	9/5	16	1111111111000001	D/5	16	1111111111100101
1/5	11	11111110110	5/6	16	1111111110100001	9/6	16	1111111111000010	D/6	16	1111111111100110
1/6	16	1111111110000100	5/7	16	1111111110100010	9/7	16	1111111111000011	D/7	16	1111111111100111
1/7	16	1111111110000101	5/8	16	1111111110100011	9/8	16	1111111111000100	D/8	16	1111111111101000
1/8	16	1111111110000110	5/9	16	1111111110100100	9/9	16	1111111111000101	D/9	16	1111111111101001
1/9	16	1111111110000111	5/A	16	1111111110100101	9/A	16	1111111111000110	D/A	16	1111111111101010
1/A	16	1111111110001000	6/1	7	1111011	A/1	9	111111010	E/1	16	1111111111101011
2/1	5	11100	6/2	12	111111110110	A/2	16	1111111111000111	E/2	16	1111111111101100
2/2	8	11111001	6/3	16	1111111110100110	A/3	16	1111111111001000	E/3	16	1111111111101101
2/3	10	1111110111	6/4	16	1111111110100111	A/4	16	1111111111001001	E/4	16	1111111111101110
2/4	12	111111110100	6/5	16	1111111110101000	A/5	16	1111111111001010	E/5	16	1111111111101111
2/5	16	1111111110001001	6/6	16	1111111110101001	A/6	16	1111111111001011	E/6	16	1111111111110000
2/6	16	1111111110001010	6/7	16	1111111110101010	A/7	16	1111111111001100	E/7	16	1111111111110001
2/7	16	1111111110001011	6/8	16	1111111110101011	A/8	16	1111111111001101	E/8	16	1111111111110010
2/8	16	1111111110001100	6/9	16	1111111110101100	A/9	16	1111111111001110	E/9	16	1111111111110011
2/9	16	1111111110001101	6/A	16	1111111110101101	A/A	16	1111111111001111	E/A	16	1111111111110100
2/A	16	1111111110001110	7/1	8	11111010	B/1	10	1111111001	F/0 (ZRL)	11	11111111001
3/1	6	111010	7/2	12	111111110111	B/2	16	1111111111010000	F/1	16	1111111111110101
3/2	9	111110111	7/3	16	1111111110101110	B/3	16	1111111111010001	F/2	16	1111111111110110
3/3	12	111111110101	7/4	16	1111111110101111	B/4	16	1111111111010010	F/3	16	1111111111110111
3/4	16	1111111110001111	7/5	16	1111111110110000	B/5	16	1111111111010011	F/4	16	1111111111111000
3/5	16	1111111110010000	7/6	16	1111111110110001	B/6	16	1111111111010100	F/5	16	1111111111111001
3/6	16	1111111110010001	7/7	16	1111111110110010	B/7	16	1111111111010101	F/6	16	1111111111111010
3/7	16	1111111110010010	7/8	16	1111111110110011	B/8	16	1111111111010110	F/7	16	1111111111111011
3/8	16	1111111110010011	7/9	16	1111111110110100	B/9	16	1111111111010111	F/8	16	1111111111111100
3/9	16	1111111110010100	7/A	16	1111111110110101	B/A	16	1111111111011000	F/9	16	1111111111111101
3/A	16	1111111110010101							F/A	16	1111111111111110

ISO/IEC 10918-1 : 1993(E) -Table K.5 – Table for luminance AC coefficients (sheet 1 of 4)

Jakie osiągamy przykładowe efekty poprzez zastosowanie kodowania VLC współczynników AC?

Powyżej mieliśmy przykładowy ciąg współczynników:
$ZZ_{[1...63]}$=[…0,0,-4,0,7,0,0,0,3,0,-1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0] czyli 31 współczynników

Otrzymaliśmy ciąg = 1111110111011100111111110111110001010 składający się z - 37 bitów (po VLC Huffman`a), czyli mniej niż 11 bit * 31 (współczynników) = 341 bitów potrzebnych do zakodowania tej ilości nieskwantowanych współczynników AC (bez kompresji);
lub 8 bit *31 = **248** bitów jakie były by potrzebne na zakodowanie skwantowanych współczynników AC.
Im mniej niezerowych wartości AC i powtarzających się ciągów zer tym większy stopień kompresji:
stopień kompresji CR (*compression ratio*).

$$CR = B_{original} : B_{compressed}$$

procent kompresji

$$CP = \left(1 - \frac{1}{CR}\right) \bullet 100\%$$

Rezultatem całego procesu jest plik wynikowy ze skompresowanym obrazem, co pozwala na łatwą edycję takiego pliku (proces dekompresji takich danych przebiega w odwrotnym kierunku do opisanego powyżej).

*** Prognozowanie** (predykcja) jest naukowym sposobem przewidywania, w jaki sposób będą kształtowały się w przyszłości procesy lub zdarzenia.
Zwykle predykcja opiera się na tworzenie statystycznego rankingu występowania symboli.

Opracowanie:
inż. Zbigniew Małach
Zbyma72age