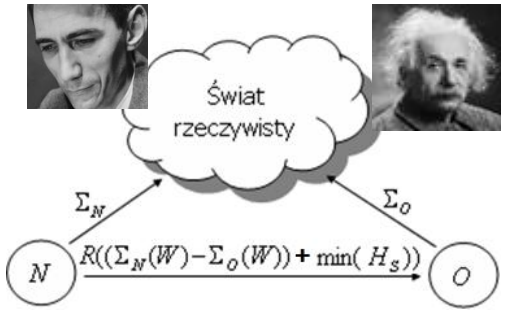
Podstawy Teorii Informacji

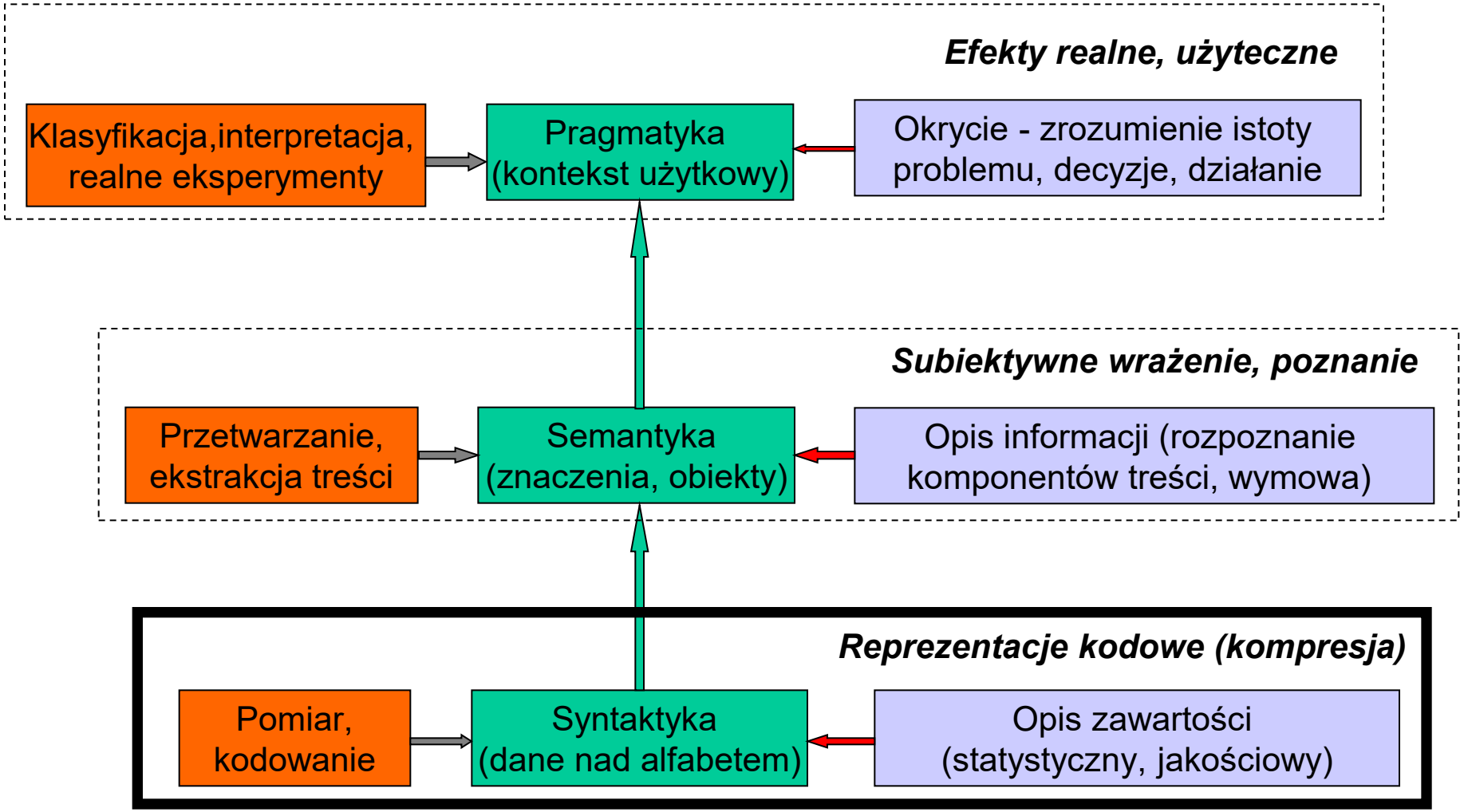


Reprezentacje informacji (problem skutecznej kompresji)

Artur Przelaskowski
MiNI, p. 506, artur.przelaskowski@pw.edu.pl, tel.w. 7821

Materiały: www.ire.pw.edu.pl/~arturp/Dydaktyka/PTI

Poziomy reprezentacji informacji

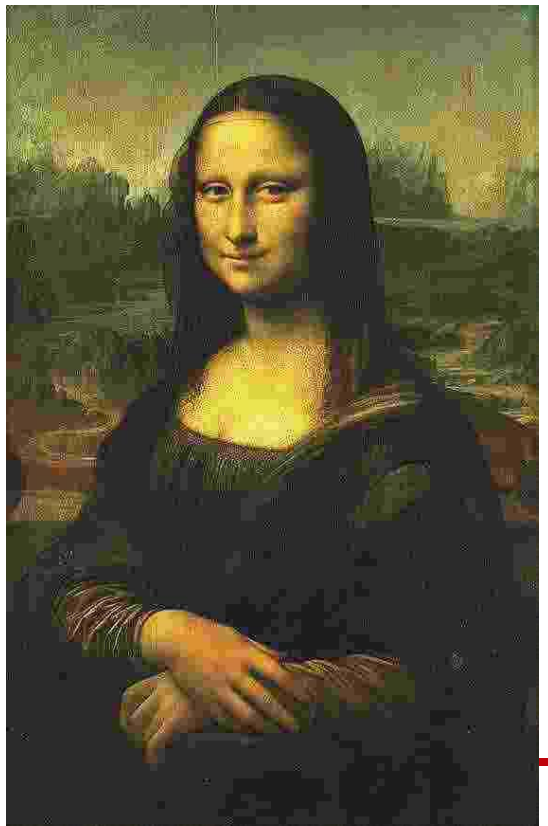


Informacja to niematerialna abstrakcja, którą urealnia (materializuje) reprezentacja **kodowa** (nośnik i reguła) formując **komunikat** (określoną formę przekazu)

Kompresja kształtuje przekaz informacji

Def. 1 (syntaktyczna) Kompresja to odwracalny lub nieodwracalny proces redukcji bitowego rozmiaru reprezentacji danych źródłowych

Def. 2 (semantyczno-pragmatyczna) Kompresja to wyznaczanie możliwie użytecznej w określonym zastosowaniu reprezentacji danych źródłowych (czyli wyznaczanie reprezentacji informacji)



Mona Lisa

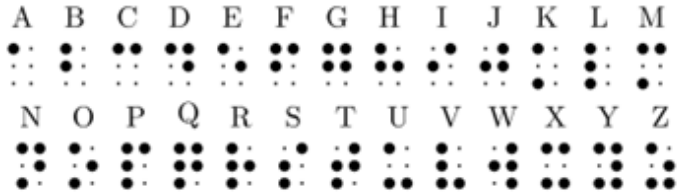
http://en.wikipedia.org/wiki/Image:Mona_Lisa.jpg

Kompresja ułatwia komunikację informacji

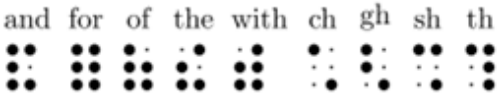
rysunki na ścianach jaskini

- znaki dymne Indian służą kodowane i upakowaniu informacji
- kod Braille'a z 1820 roku, umożliwiający niewidomym czytanie
  - składa się z grup (lub komórek) po 3 × 2 kropki każda, wytłoczonych na grubym papierze
  - kropki w grupie mogą być płaskie albo pogrubione, co daje równoważność 6 bitów, czyli 64 możliwości
- alfabet Morse'a z 1838 roku jest kodem telegraficznym składającym się z krótkich i dłuższych kresek (lub kropek i kresek)
  - krótka kreska trwa jedną jednostkę czasu, a dłuższa-trzy jednostki
  - odstęp między kropkami i kreskami jednego znaku to jedna jednostka, między znakami- trzy jednostki czasu, a pomiędzy słowami- sześć jednostek (pięć przy automatycznej transmisji)

B.Pascal: ***mój list jest dłuższy niż zwykle, gdyż nie miałem czasu, żeby napisać krócej...***



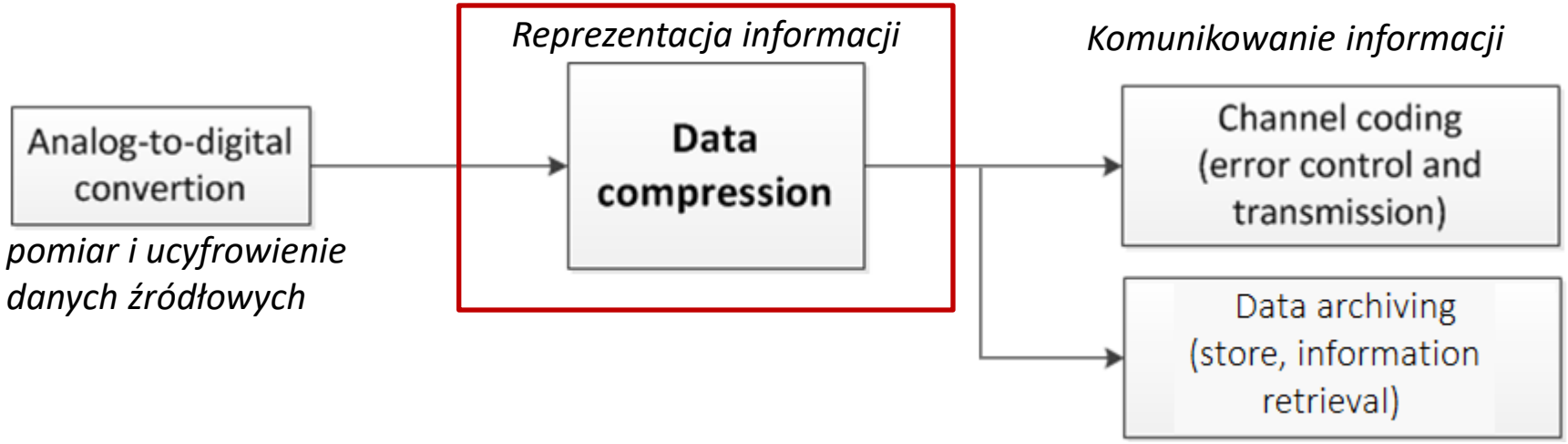
The 26 Braille Letters



Some Words and Strings in Braille

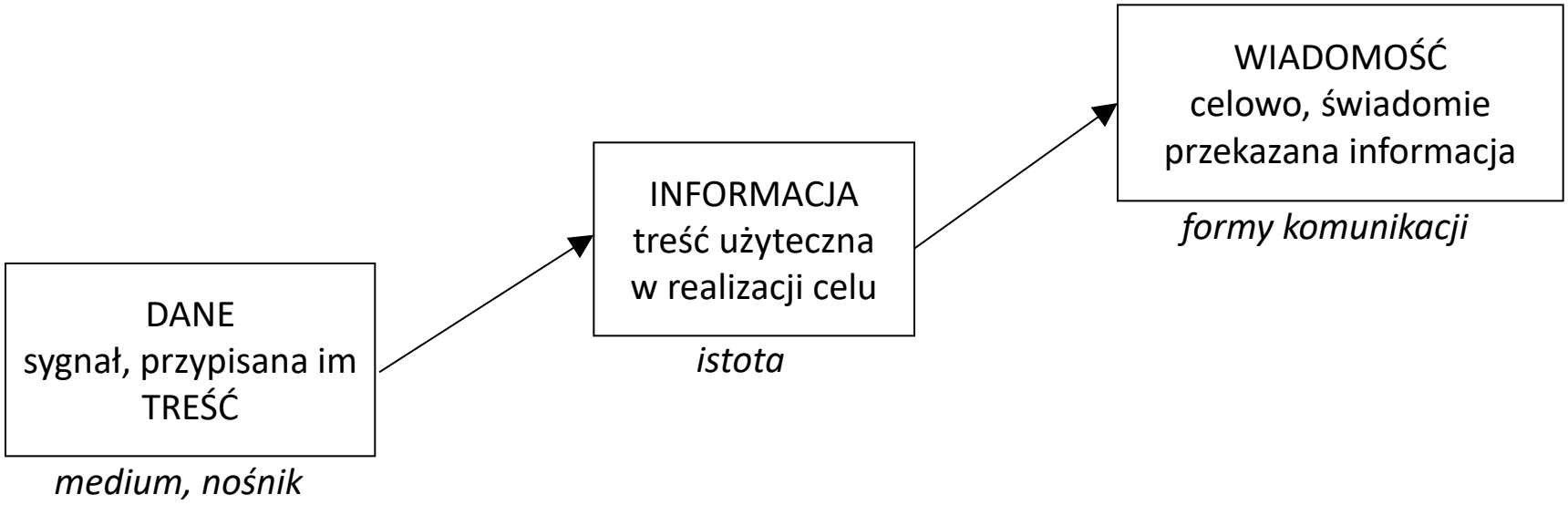
A	.-	N	-.	1	.----	Period	.-.-.-
B	-...	O	---	2	..---	Comma	--..--
C	-.-.	P	.--.	3	...--	Colon	---...
Ch	----	Q	--.-	4	....-	Question mark	..--..
D	-..	R	.-.	5	.....	Apostrophe	.----.
E	.	S	...	6	-....	Hyphen	-....-
F	..-.	T	-	7	--...	Dash	-..-.
G	--.	U	..-	8	---..	Parentheses	-.--.-
H	....	V	...-	9	----.	Quotation marks	.-..-.
I	..	W	.--	0	-----		
J	.---	X	-..-				
K	-.-	Y	-.--				
L	.-..	Z	--..				
M	--						

Kontekst kompresji danych jako metody reprezentacji źródeł informacji



- Kompresja danych służy przekazowi informacji (pomiar, gromadzenie, przesyłanie)
- **Jest uogólnieniem pojęcia kodowanie** poprzez dostosowanie do potrzeb odbiorcy
- Optymalizacja reprezentacji informacji poprzez
  - redukcję różnego typu nadmiarowości reprezentacji źródłowej
  - wprowadzenie dodatkowych form nadmiarowości celem zabezpieczenia przed błędami transmisji, wyjaśnienia znaczeń/uwarunkowań zarejestrowanych obserwacji (np. ułatwienie przeglądania czy indeksowania przesyłanych danych)
- Pragmatyka: reprezentacja, uporządkowanie i ekstrakcja informacji, przyspieszenie transmisji, zwiększenie wartości użytkowej, wyjaśnianie treści

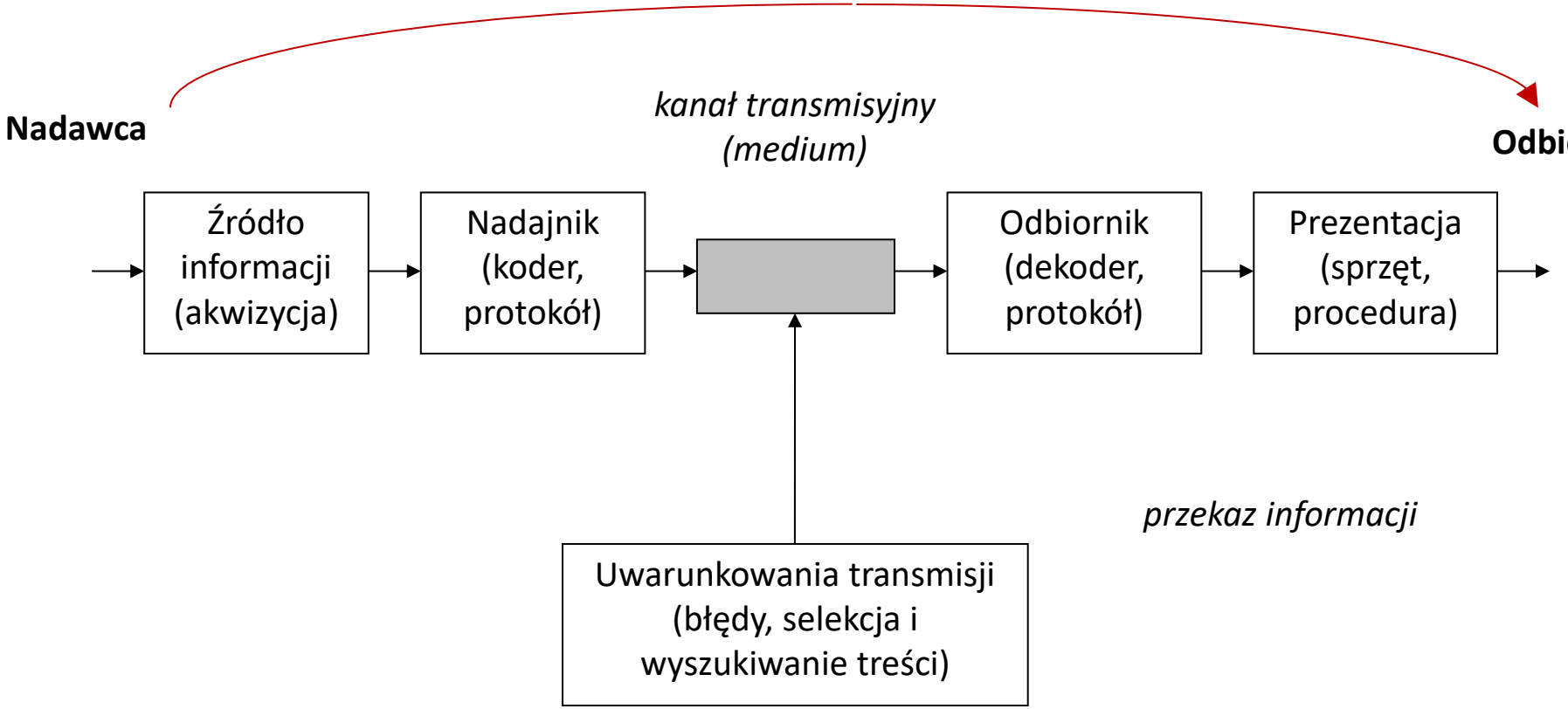
Uwarunkowania informacji …



- dane nie zawsze wyrażają informację (nadmiarowość, użyteczność)
- ważne jest uzgodnienie treści (semantyki)
- wygodna forma (składnia-syntaktyka, nośnik, technologia) wobec ograniczeń treści
- przeciwieństwa: pragmatyczny, obiektywizowany przekaz versus ludzka myśl/odczucie (subiektywizm, głębia, indywidualizm, zmienność)

Transmisja: źródło – kanał - odbiorca …

- Występuje w kontekście **przekazu** (podstawowy schemat nadawca - kanał transmisji - odbiorca)



- **Koszt** pozyskania informacji (zwykle << korzyści)
- **Przekaz** powinien być wielostrumieniowy (wielowymiarowy, multimedialny), możliwie realny (inaczej staje się interpretacją ...), choć informacja może być jedynie śladem ...

Jak skompresować Lenę?

Lenna, 512x512, 8bpp

Lenna, 128x128

Lenna, 512x512, 3bpp

Lenna, 512x512, 0.5 bpp (JPEG)

Podstawowe pojęcia: efektywność

- Obliczeniowe miary efektywności
  - Stopień kompresji

    $CR=|L_{source}|/|L_{coded}|$ (:1), e.g. CR=5:1 lub CR=5
  - Procent kompresji

    CP=(1-1/CR)*100%
  - Średnia bitowa

    $BR= L_{zbiór}$ [w bitach]/$L_{zbiór}$ [w symbolach alfabetu]

    np. 8 bitów/piksel dla źródłowego zbioru danych obrazowych czy też 2,55 bitów/symbol dla skompresowanego zbioru danych tekstowych nad określonym alfabetem

- Inne kryteria
  - Czas kompresji (T)
  - Jakość efektów kompresji (Q)
  - Iloczyn CR i Q lub tym podobna relacja stopnia kompresji do jakościowych efektów kompresji
  - Uporządkowanie (hierarchia treści, jakości lub ogólnie informacji)
  - Skalowalność (od ogółu do szczegółu)
  - Kontrola długości reprezentacji kodowej (w koderze)
  - Odporność na zakłócenia
  - Wymagania implementacji (zrównoleglenie, oszczędność pamięci itd.)
  - Adaptacyjność, elastyczność, uniwersalność kodu, integracja w określonym środowisku etc.

Podstawowe pojęcia: odwracalność procesu kompresji

- Odwracalność numeryczna

- Odwracalność percepcyjna (psychowizualna)
- Odwracalność lokalna
- Odwracalność semantyczna
- Odwracalność syntaktyczna

- Selekcja informacji – kompresja stratna/nieodwracalna ze względu na
  - konkretne korzyści użytkowe: znacząca redukcja rozmiaru zbioru archiwizowanych danych, ze względu na czas transmisji, wymagania sprzętowe itp.
  - kontrolowaną selekcję informacji, np. ograniczenie obszaru obrazu do rozpoznanych wstępnie obiektów
  - itp.

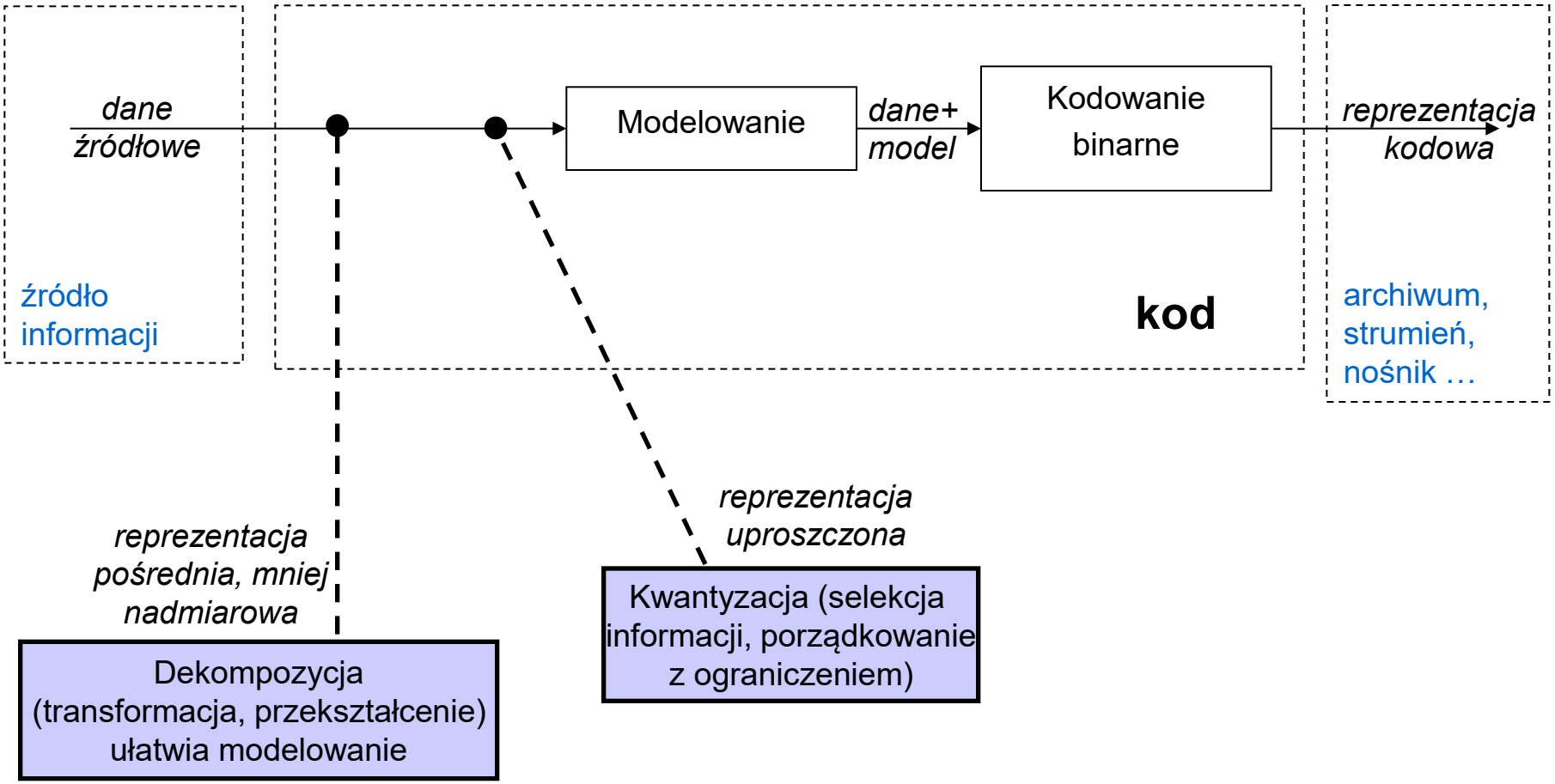
Podstawowe pojęcia w kompresji danych

- Kod – **reguła** (koncepcja, algorytm) tworzenia reprezentacji kodowej (efektywnego ciągu bitowego) danych źródłowych, która jest dekodowalna według przyjętej zasady, procedury itp. zgodnie z przyjętymi kryteriami efektywności

- Kodek – **realizacja kodu**, algorytmu kodowania (koder) i dekodowania (dekoder) - oprogramowanie, sprzęt)

- Kodowanie – **proces** tworzenia kodowej reprezentacji danych źródłowych (za pomocą określonego kodera według ustalonego kodu)
- Dekodowanie – **odtwarzanie** reprezentacji danych źródłowych (oryginalnej lub przybliżonej)

Kodowanie (paradygmat podstawowy)



Modelowanie (1 etap kodowania źródłowego)

- **pomysł**, inteligencja, mózg (opis informacji, inteligencja)
- opracowanie **modelu** danych źródłowych: probabilistycznego, deterministycznego, funkcjonalnego (przekształcenia), aproksymacji, dziedzinowego, przybliżonego, hybrydowego
  - ustalenie **specyfiki** modelu: adaptacyjny, statyczny, z uczeniem, przełączany, blokowy, słownikowy, obiektowy itp.
- dodatkowo: wykorzystanie **wstępnego przekształcenia danych** (zgodnie z przyjętym modelem danych), uzyskanie pośredniej reprezentacji w nowej dziedziniej (uporządkowanej, rzadkiej etc.) **podatnej na efektywne modelowanie i kodowanie**

Kody binarne (2 etap kodowania źródłowego)

- silnik, napęd
  - konsekwencja **modelu**
  - kryterium minimalizacji wyjściowego ciągu bitów
  - reguła ustalania słów kodowych (w oparciu o model)
  - podstawowa koncepcja: różnicowanie długości i formy słów kodowych zależnie od przyjętych kryteriów
- wykorzystanie konkatenacji **słów kodowych**
  - słowa przypisane symbolom, blokom symboli, stanom modelu
  - kodowanie przyrostowe
- warunek konieczny: jednoznaczna dekodowalność
  - różnicowanie długości słów (lub ich części)
  - unikanie nadmiarowości

Zakodujmy!

We: $\mathbf{s}_{we}$ = (5, 5, 5, 2, 2, 11, 11, 11, 11, 11, 8)

A_S = {0,1,2,…,15}

Wy: $B_4(\mathbf{s}_{we})$= 0101 0101 0101 0010 0010 1011 1011 1011 1011 1011 1000

- długość: 44 bity

M1: $R(\mathbf{s}_{we})$ = ((3, 5), (2, 2), (5, 11), (1, 8)) =$((l_i,s_i))_{i=1,2,…}$

Wy1: $B_3(l_i\text{-}1)|B_4(s_i)_{i=1,2,….}$=010|0101 001|0010 100|1011 000|1000

- długość 28 bitów

M2: wagi symboli (kolejno $3_{(5)}$, $2_{(2)}$, $5_{(11)}$, $1_{(8)}$

K2: $K_{VLC}(11)=0$, $K_{VLC}(5)=10$, $K_{VLC}(2)=110$, $K_{VLC}(8)=111$

Wy2: $K_{VLC}(\mathbf{s}_{we})$=10 10 10 110 110 0 0 0 0 0 111 - 20 bitów + nagłówek

D: $R''(\mathbf{s}_{we}) = \{r_i : r_i = s_i - s_{i-1},\ i = 1, \ldots, 11,\ s_0 = 0\}=$

$=\{5, 0, 0, -3, 0, 9, 0, 0, 0, 0, -3\}$

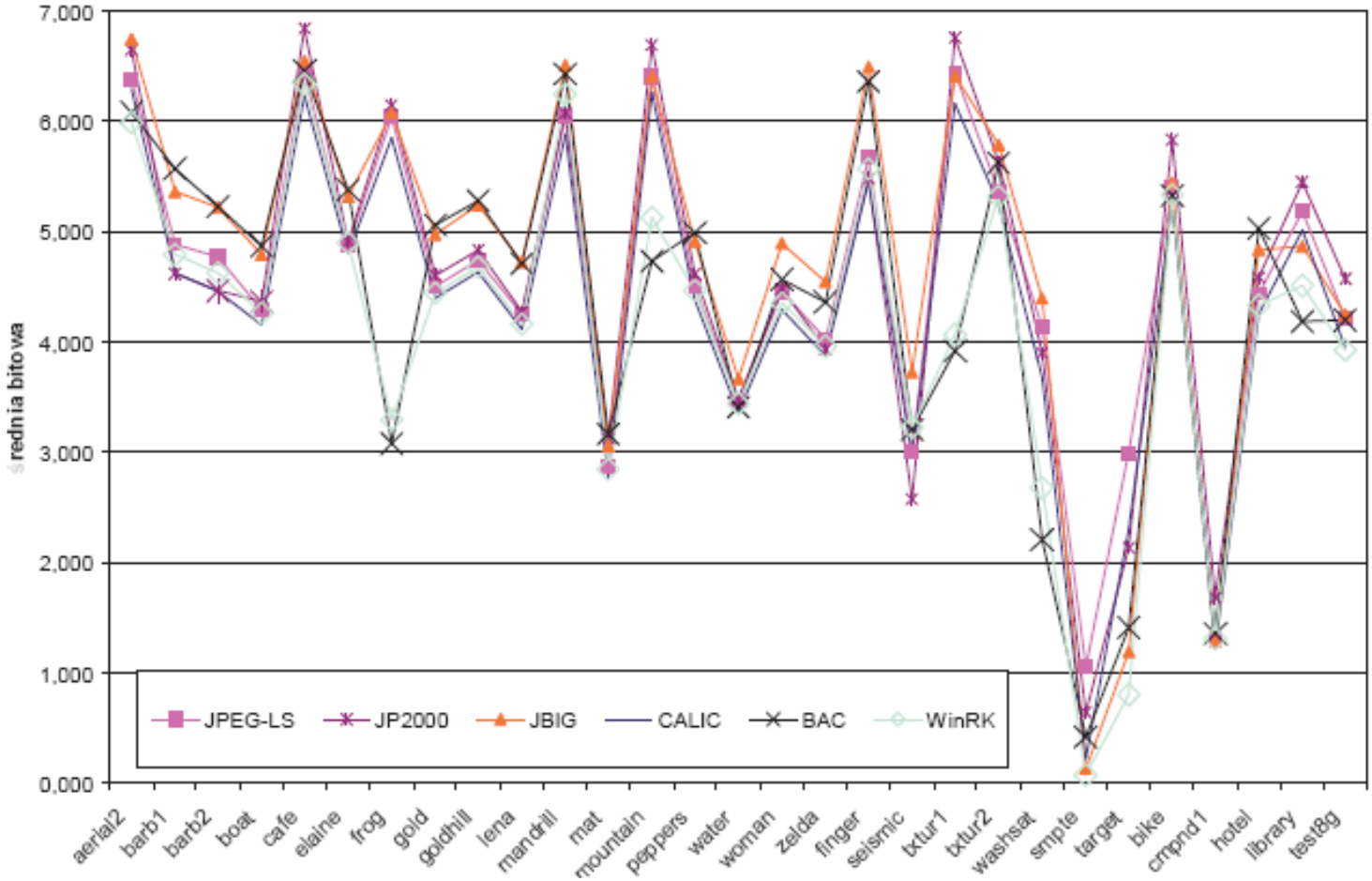
M3: wagi symboli (kolejno $1_{(5)}$, $7_{(0)}$, $2_{(-3)}$, $1_{(9)}$)

K3: $K_{VLC}(5)=110$, $K_{VLC}(0)=0$, $K_{VLC}(-3)=10$, $K_{VLC}(9)=111$

Wy3: $K_{VLC}(\mathbf{s}_{we})$=110 0 0 10 0 111 0 0 0 0 10 - 17 bitów + nagłówek

Eksperymentalna weryfikacja

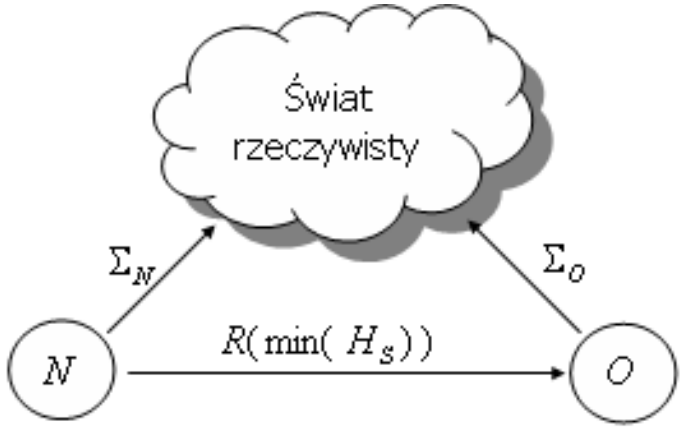
- Archiwizery (uniwersalne narzędzia do archiwizacji danych)



4,55 bpp (JPEG-LS), 4,60 bpp (JPEG2000), 4,75 bpp (JBIG), 4,35 bpp (CALIC), 4,36 (BAC) oraz 4,1 bpp (WinRK v. 2.1.6)

Pragmatyka maszyny: informacja zobiektywizowana (znaczenie bez znaczenia)

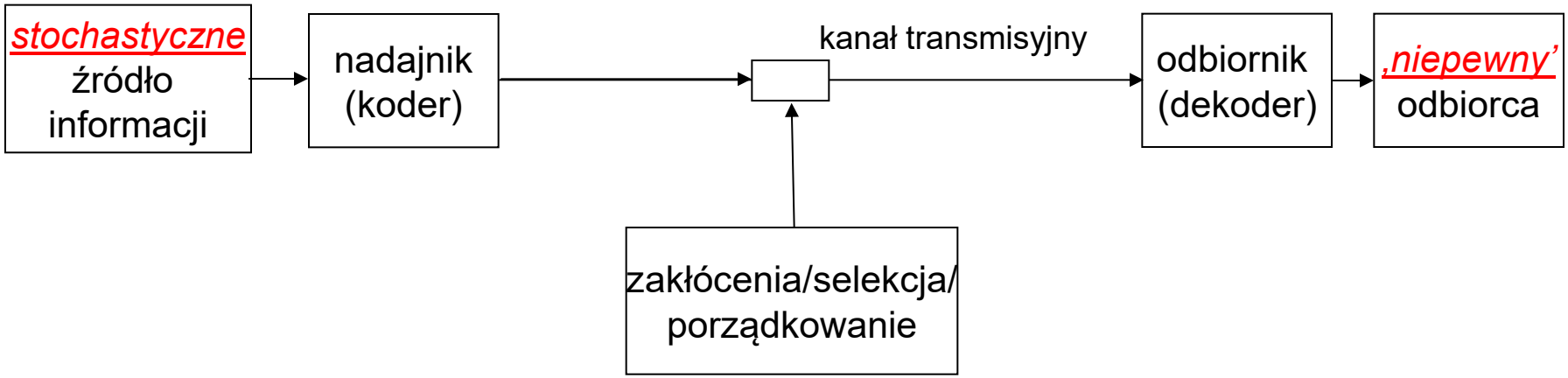
- Abstrahujemy od semantyki (dominuje syntaktyka)
- **Probabilistyczny model** źródła informacji (Shannon)
- **Informacja to poziom niepewności** odbiorcy w kontekście przekazu (wszystko co nieprzewidywalne jest informacją)
- **Miara ilości informacji** (niepewności) nawiązuje do entropii swobodnej (termodynamika – energia swobodna, *Massieu-Plancka*, *Gibbs* jako miary nieuporządkowania w zamkniętym systemie cząstki w równowadze pod względem rozkładu prawdopodobieństwa energii)



Model niepewności

reprezentacja danych *R* źródła S o minimalizowanej entropii przesyłana od nadawcy *N* do odbiorcy *O* przy (z założenia) zgodnych wartościach semantycznych ($\Sigma_N = \Sigma_O$), gdzie Σ jest funkcją semantyczną

Matematyczna interpretacja realnego przekazu



- Źródło losowe
- Odbiorca uproszczony, bez cech subiektywnych, przesłanek pragmatycznych (bardziej niepewny przy silniejszej informacji)
- Kanał stratny lub bezstratny
- Koder wykorzystuje model z rozkładem prawdopodobieństw możliwych symboli alfabetu
- Kryteria kodowania: minimalna średnia bitowa, zbiegająca do granicznej entropii

Teoria Shannona – podstawowe cele

- modelowanie (statystyczne modele źródeł informacji)
  - stworzenie wiarygodnej, strukturalnej charakterystyki źródła informacji tłumaczącego dane źródłowe w sposób zwarty
  - *de facto* jest to zwarty opis informacji w celu jej obróbki/kodowania/analizy
- wyznaczanie ilości informacji dostarczanej przez źródła
  - charakterystyka źródeł informacji celem ich analizy/kodowania (alfabet, prawdopodobieństwa)
  - miara informacji (entropia)
- opracowanie reprezentacji konkretnych źródeł informacji (generowanych realizacji)
  - kody binarne

Losowe (stochastyczne) modele źródeł informacji

- Źródło dostarcza (emituje) symbole a_i , i=1,…,n nad określonym alfabetem z określonym prawdopodobieństwem p_i=Pr(a_i)= $P(a_i)$
- minimalny rozmiar alfabetu to 2, inaczej nie ma informacji
- Informacja własna symbolu a_i: $I(a_i) = \lg(1/p_i)$
- średnia informacja symboli źródła (dostarczana przez źródło):

$$-\sum_{i=1}^{n} P(a_i) \log_2 P(a_i)$$

- Informacja jest określoną realizacją zmiennej losowej, procesu, łańcucha

- Modele dyskretne
  - stacjonarne: prawdopodobieństwo jest niezmienne względem przesunięć w czasie
  - ergodyczne: każda generowana sekwencja ma te same co do natury właściwości

Model bez pamięci (DMS)

- alfabet źródła $A_S = \{a_1, a_2, \ldots, a_n\}$
- zbiór prawdopodobieństw $P_S = \{p_1, p_2, \ldots, p_n\}$

Rozszerzenie stopnia N źródła S

$$A_S^N = \underbrace{A_S \times A_S \times \cdots \times A_S}_{N}$$

łączenie symboli w bloki N symboli

Iloczyn kartezjański

$$X \times Y := \{(x, y) : x \in X \ \wedge \ y \in Y\}$$

Model z pamięcią (CSM)

Wykorzystywane są prawdopodobieństwa warunkowe $P(\cdot|\mathbf{c}^t)$

- alfabet źródła $A_S = \{a_1, a_2, \ldots, a_n\}$
- zasada określania kontekstu C
- zbiór kontekstów C dla źródła S postaci
  $A_C{}^S = \{\mathbf{b}_1, \mathbf{b}_2, \ldots, \mathbf{b}_k\}$
- zbiór prawdopodobieństw warunkowych
  $P(a_i|\mathbf{b}_j) = N(a_i, \mathbf{b}_j)/N(\mathbf{b}_j)$

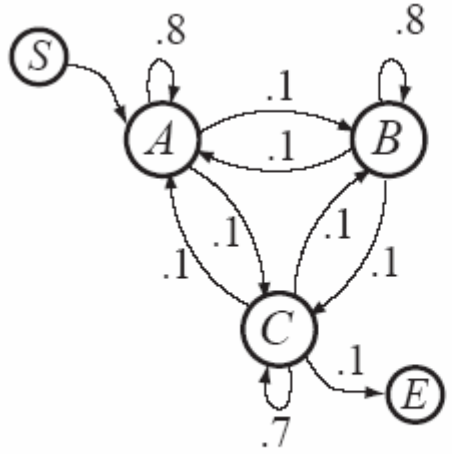
Model Markowa (rzędu m)

$P(a_i|\mathbf{c}_j) = P(a_i|a_{j1}, a_{j2}, \ldots, a_{jm})$

czyli że np. $P(s_{t+1}|\mathbf{c}^t) = P(s_{t+1}|s_t, s_{t-1}, \ldots, s_{t-m})$

Model Markowa

- następny stan zależy od obecnego, model ma zdeterminowane wyjście, nie zależy od *t*
- przykład modelu rzędu 1 z alfabetem {A,B,C}

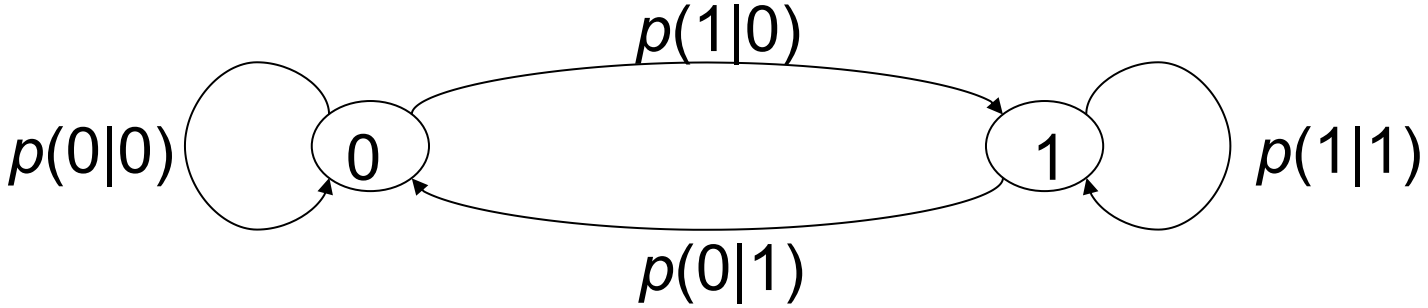


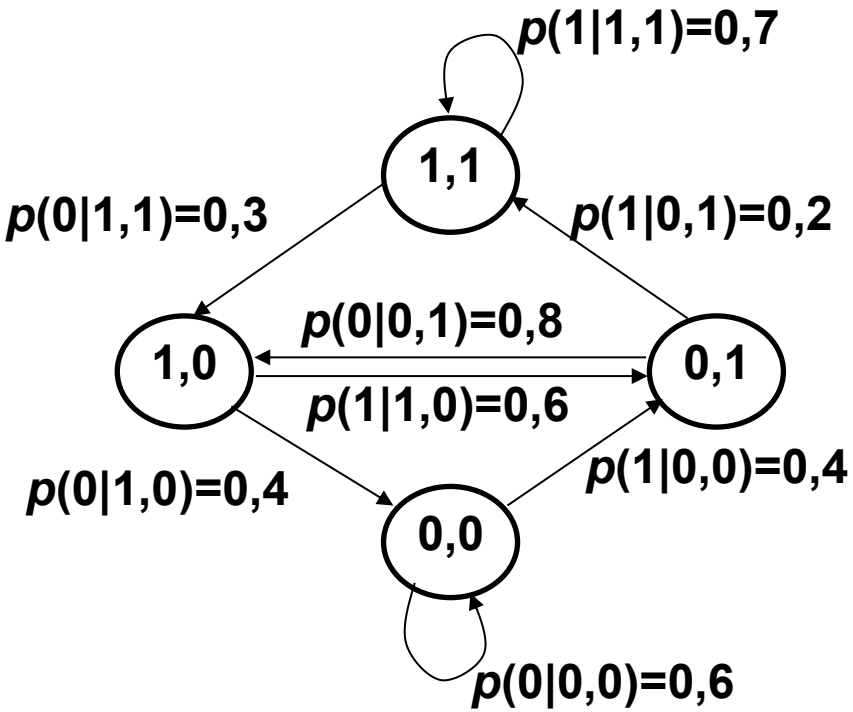
$P_{j,i}$	a_i S	A	B	C	E
$b_j=a_{j1}$ S	0	1	0	0	0
A	0	.8	.1	.1	0
B	0	.1	.8	.1	0
C	0	.1	.1	.7	.1
E	0	0	0	0	1

S A A A A A A A A B B B B B B B B B C C C C B B B B B B C E

- skończona liczba stanów zależy od rzędu modelu i rozmiaru alfabetu A_S

Przykłady modelu Markowa





Miara ilości informacji (entropia)

- informacja własna symbolu a_i: $I(a_i) = \lg(1/p_i)$ $\quad P(a_i) = p_i$
- entropia jako średnia informacja źródła symboli (średnio na symbol)

$$-\sum_{i=1}^{n} P(a_i) \log_2 P(a_i) \;=\; H(S_{\mathrm{DMS}})$$

- entropia warunkowa źródła Markowa

$$H(S|a_{j_1}, a_{j_2}, \ldots, a_{j_m}) = -\sum_{i=1}^{n} P(a_i|a_{j_1}, \ldots, a_{j_m}) \log_2 P(a_i|a_{j_1}, \ldots, a_{j_m})$$

(w pewnym stanie)

$$H(S|C^{(m)}) = \sum_{A_{C^{(m)}}^{S}} P(a_{j_1}, \ldots, a_{j_m}) H(S|a_{j_1}, \ldots, a_{j_m})$$

(uśredniamy po wszystkich stanach)

Entropia (łączna)

$$H(S) = \lim_{m \to \infty} \frac{1}{m} I_m$$

rozmiar problem dąży do nieskończoności

gdzie

$$I_m = -\sum_{j_1=1}^{n} \sum_{j_2=1}^{n} \cdots \sum_{j_m=1}^{n} P(a_{j_1}, a_{j_2}, \ldots, a_{j_m}) \lg P(a_{j_1}, a_{j_2}, \ldots, a_{j_m}) =$$

$$= -\sum_{j_1,\ldots,j_m=1}^{n} \Pr(s_1 = a_{j_1}, \ldots, s_m = a_{j_m}) \lg \Pr(s_1 = a_{j_1}, \ldots, s_m = a_{j_m})$$

Relacje entropii hipotetycznej i modelowanej

$$H(S) \leq H(S|C^{(m)}) \leq H(S_{\mathrm{DMS}})$$

Zasada: modele Markowa **możliwie** wysokiego rzędu, dostosowane do realnych zależności danych, są szczególnie użyteczne

Kilka przykładów skutecznych kodów – metod bezstratnych i stratnych

METODY

Kodowanie źródłowe (ze źródeł analogowych)

- W **PCM** (*Pulse Code Modulation* – metoda reprezentacji sygnału analogowego w systemach cyfrowych) zwykle stosowany kod dwójkowy o określonej dynamice (alfabecie), mający określoną interpretację liczbową

  Kod dwójkowy (dane źródłowe): $B_k(a_i)= \xi_i =(i)_{2,k}$ z **alfabetem słów kodowych** $A_{Bk} =\{\xi_0, \xi_1,\ldots, \xi_{n-1}\}$

  gdzie $k = \lceil \log_2 n \rceil$ - liczba bitów słów kodowych, *n* to liczba możliwych postaci danych źródłowych $A_S=\{a_0,a_1,\ldots,a_{n-1}\}$, $a_i \in A_S$ (alfabet źródła informacji)

  Np. w kwantyzacji mamy: $A_Y=\{-2,-1,1,2\}$, wtedy $n=4$, $k=2$ oraz $A_{B2}=\{00,01,10,11\}$

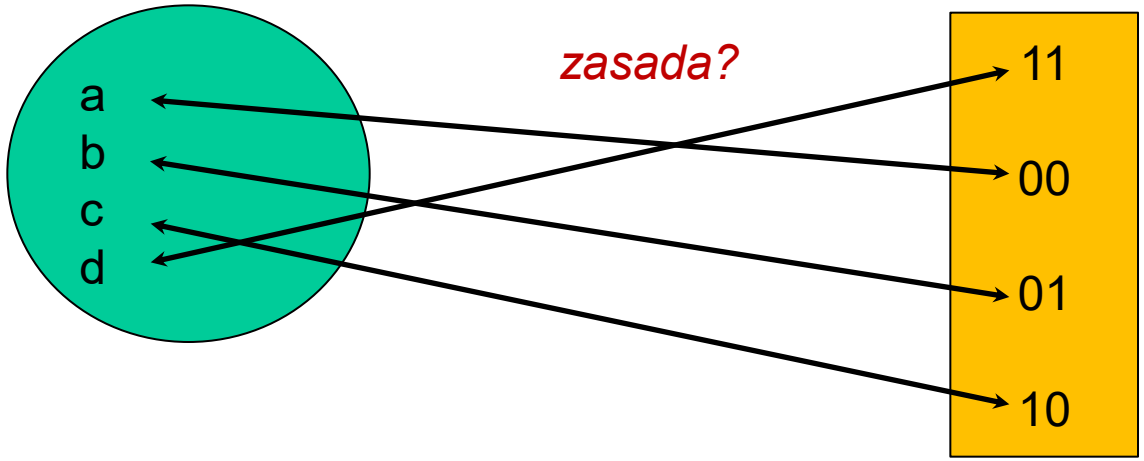
  Inny przykład: $A_S=\{$'Jola','Marta','Stasia','Madzia','Tola'$\}$, gdzie $n=5$, $k=3$

  - wtedy alfabet binarnych słów kodowych przybiera postać $A_{B3}=\{000,001,010,011,100\}$

- Słowa kodowe reprezentują przedziały kwantyzacji o ustalonej charakterystyce
- Nadmiarowość reprezentacji akwizycji (źródłowej) wymusza stosowanie efektywniejszych kodów (Huffmana, Golomba, itd.)

Kody jednoznacznie dekodowalne

- Kod jako funkcja wzajemnie jednoznaczna (bijekcja)



- Każdemu symbolowi alfabetu źródła informacji (cyfrowej) wzajemnie przypisane jest dokładnie jedno wyjątkowe słowo kodowe alfabetu danego kodu
- Oba zbiory są różnowartościowe i równoliczne
- Kod jest regułą wzajemnego jednoznacznego przypisania elementów obu zbiorów

Jednoznaczna dekodowalność kodu – zasady konstrukcji/weryfikacji

1) Kody symboli: liczba różnych słów kodowych odpowiada liczbie symboli

$K_1(\{a_1,a_2,a_3,a_4\})=A_{K2}=\{00,01,10,11\}$ $K'_1(a_1,\ldots,a_4) = 11$

2) Analiza długości słów: nierówność **Krafta-MacMillana**

- warunek konieczny jednoznacznej dekodowalności kodu:

$$\sum_{i=1}^{n} 2^{-L_i} \leq 1$$

$A_{K3} = \{0,10,11,?\}$ $A_{K'3} = \{0,10,11,111\}$

- istnieje kod przedrostkowy dla dowolnych całkowitych L_i >0 spełniających tą nierówność (L_i oznacza długość w bitach *i*-tego słowa kodowego)

3) Kombinacja słów nie jest kombinacją innych słów (warunek konieczny, ale trudny do weryfikacji)

$A_{K2}= \{1,10,00\}$ $A_{K'2}= \{1,10,01\}$ ponieważ 10|1 oraz 1|01

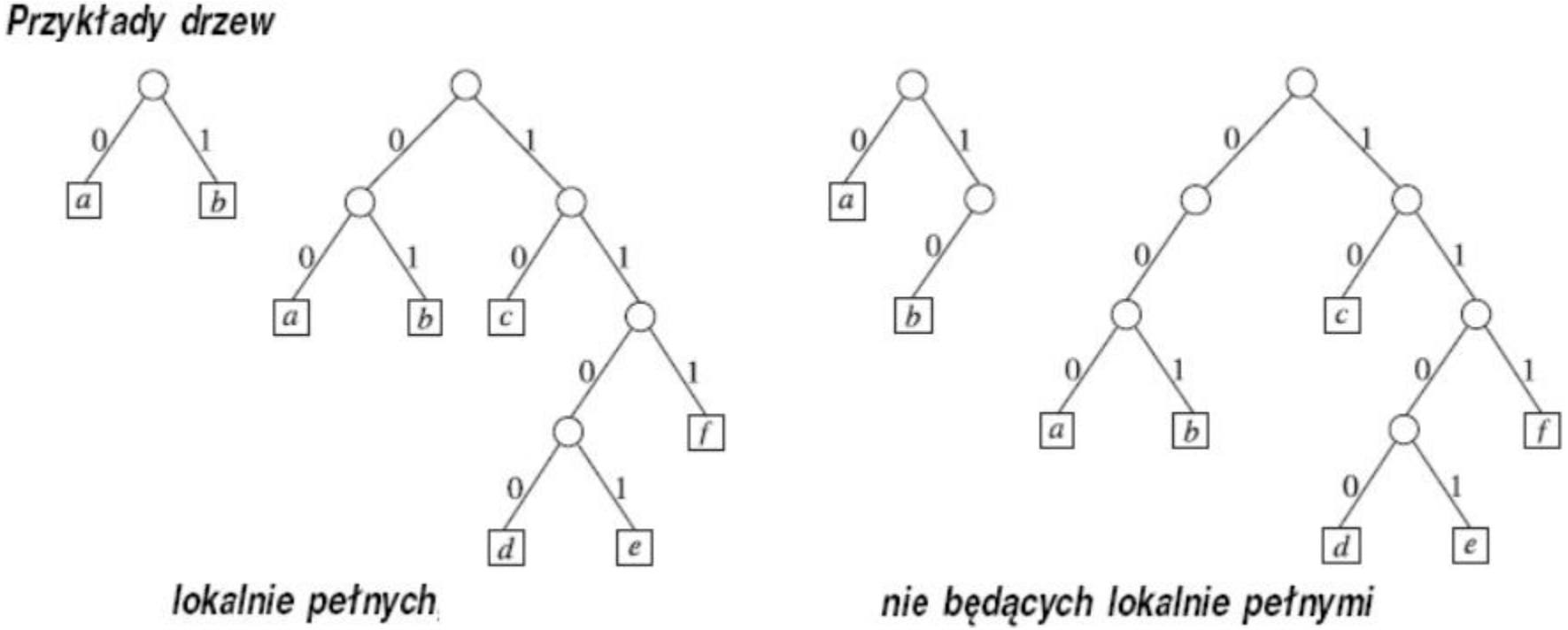
Najlepiej: kody przedrostkowe

(żadne słowo kodowe nie jest przedrostkiem innego słowa kodowego)

4) kody przedrostkowe (bezprzedrostkowe, *prefix codes, prefix condition codes*, *prefix-free codes, comma-free code, instantaneous codes*)

$$A_{K4} = \{1,01,001,0001,00001,\ldots\ldots\}$$

5) kody drzew binarnych

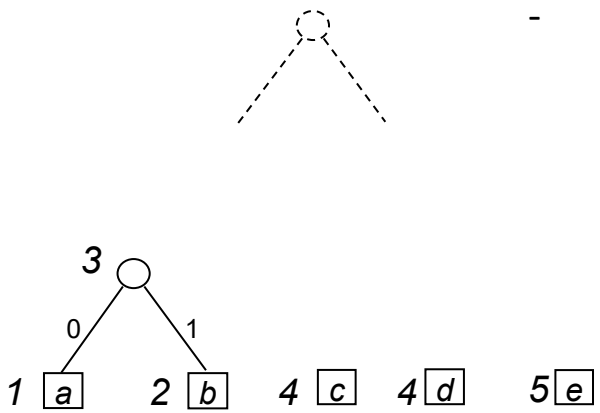


Optymalny, entropijny **kod symboli** (Huffmana)

- Dane wejściowe są określone za pomocą alfabetu symboli źródła $A_S = \{a_1, a_2, \ldots, a_n\}$
- Wejściowy model rozkładu prawdopodobieństw symboli $P_S=\{p_1,\ldots,p_n\}$
- Bijekcja symboli alfabetu w zróżnicowane słowa kodowe im przypisane (jednoznacznie dekodowalne)

symbol ⟶ słowo kodowe

- Zasada ogólna: projektowanie drzew binarnych: symbole są umieszczane w liściach drzewa - idąc **od korzenia** przez krawędzie opisane bitami (etykiety) sczytujemy słowa kodowe
- Optymalna metoda konstrukcji drzew (**kod Huffmana**): uporządkowane, sukcesywne kształtowanie drzewa binarnego od liści do korzenia (**bottom-up**)



- pojedyncze symbole alfabetu są przypisywane kolejnym węzłom-liściom, które są sukcesywnie łączone w rosnące poddrzewa tworząc finalne drzewo binarne
- zasadnicza reguła konstrukcji: dwa symbole o aktualnie najniższych wagach są łączone w pierwszej kolejności
- efekt: **najmniej prawdopodobne symbole mają przypisaną największą liczbę bitów** (najdłuższe słowa kodowe), bo znajdują się na najniższym (najgłębszym) poziomie drzewa

Algorytm Huffmana

(drzewo kodowe konstruowane od dołu – z poziomu liści)

1. Określ wagi symboli na podstawie częstości wystapień lub wiedzy a priori; symbole wraz z wagami przypisz liściom – wolnym węzłom/wierzchołkom
2. **Sortuj** listę wierzchołków wolnych w nierosnącym porządku wag

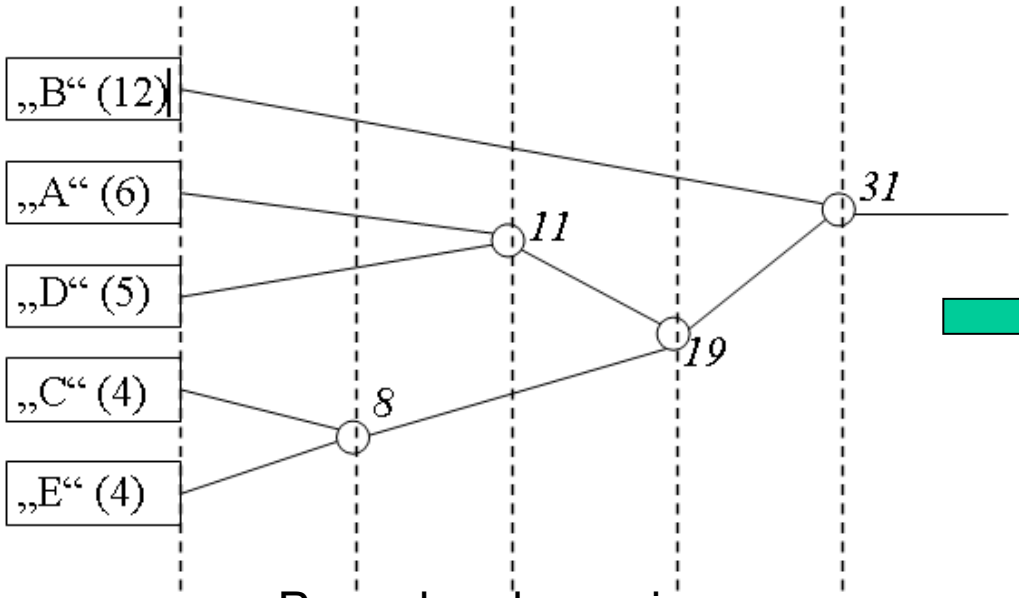
Pętla:

3. **Podłącz** dwa wolne wierzchołki z najmniejszymi wagami w nowe, większe poddrzewo
   - Utwórz nowy węzeł-rodzica z wagą równą sumie wag dzieci
   - Przypisz gałęziom prowadzącym od rodzica do węzłów-dzieci różne etykiety: 0 i 1 (np. lewa gałąź – 0, prawa – 1)
4. **Zaktualizuj listę wolnych węzłów:** usuń z listy wierzchołków wolnych dwa połączone węzły i wpisz na tę listę nowy wierzchołek rodzica (o wadze równej sumie wag dzieci)
5. Powtarzaj kroki 3-4 aż do momentu, gdy na liście wierzchołków wolnych pozostanie tylko korzeń drzewa

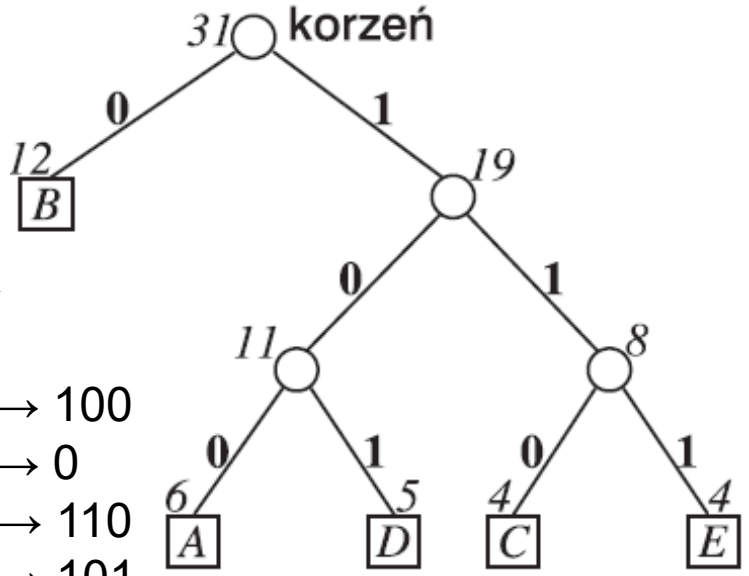
Koniec pętli

6. Ustal/odczytaj ze struktury drzewa słowa kodowe kolejnych liści (symboli) - w kolejności od korzenia do danego liścia

Przykład wykorzystania metody Huffmana



Procedura łączenia



„A” → 100
„B” → 0
„C” → 110
„D” → 101
„E” → 111

Efektywność kodu Huffmana

Symbol a_i	Oszacowane prawdopodobieństwo $P(a_i)$	Długość słowa kodowego $\lvert\varsigma_i\rvert$ [bity]	Długość sekwencji kodowej [bity]
„A”	6/31	3	18
„B”	12/31	1	12
„C”	4/31	3	12
„D”	5/31	3	15
„E”	4/31	3	12
Suma	–	–	69

Ocena względem entropii (liczonej zgodnie z przyjętym modelem prawdopodobieństwa):

Entropia $H(S_{\mathrm{DMS}}) = \mathbf{2{,}176}$ wobec **średniej bitowej** $BR_{\mathrm{Huff}} = \Sigma_i P(a_i) \cdot |\varsigma_i| = \mathbf{2{,}226}$ (+2,3%)

Kody blokowe i strumieniowe

- Rozszerzenie kodów symboli – **kody blokowe**
  - N-symboli tworzy blok, który jest kodowany za pomocą **jednego słowa kodowego**
  - Definicję bloków oraz regułę konstrukcji słów kodowych można dostosować do specyfiki problem – przykładem **kody słownikowe**

- **Kody strumieniowe**
  - Cały zbiór danych wejściowych reprezentowany **jednym, specyficznym słowem kodowym**
  - Koncepcja kodu arytmetycznego: słowo kodowe jest interpretowane jako **unikalne prawdopodobieństwo całej wejściowej sekwencji danych** (bez żadnych ograniczeń długości)
  - Na podstawie tego prawdopodobieństwa można **bezstratnie** zdekodować **dowolnie długą sekwencję wejściową danych**
  - Klucz: **całkowitoliczbowa** implementacja **(o ograniczonej precyzji)** kodu arytmetycznego (KA)

Arytmetyczne kodowanie – jedna liczba reprezentująca dowolny zbiór danych

- Zamiast relacji prawdopodobieństwo (alfabet) → słowo kodowe ustalamy jedynie prawdopodobieństwo sekwencji wejściowej
- Skumulowane prawdopodobieństwo F_S bazujące na **linii prawdopodobieństw** P_S

$A_S=\{a_1,\ldots,a_n\}$ i $P_S=\{p_1,\ldots,p_n\}$ → $F_S=\{p_1,p_1+p_2,\ldots,p_1+\ldots+p_n\}$

czyli $$F_S=\left\{F(a_i)=\sum_{j=1}^{i}p_j; i=1,\ldots,n\right\}$$

a_i	$P_s(a_i)$	$F_s(a_i)$	$\pi_s(a_i)$
„A”	2/10	2/10	[0,2/10)
„E”	1/10	3/10	[2/10,3/10)
„K”	1/10	4/10	[3/10,4/10)
„M”	1/10	5/10	[4/10,5/10)
„R”	1/10	6/10	[5/10,6/10)
„T”	2/10	8/10	[6/10,8/10)
„Y”	2/10	1	[8/10,1)

- Przyporządkowanie podprzedziału ($F(a_0)=0$):

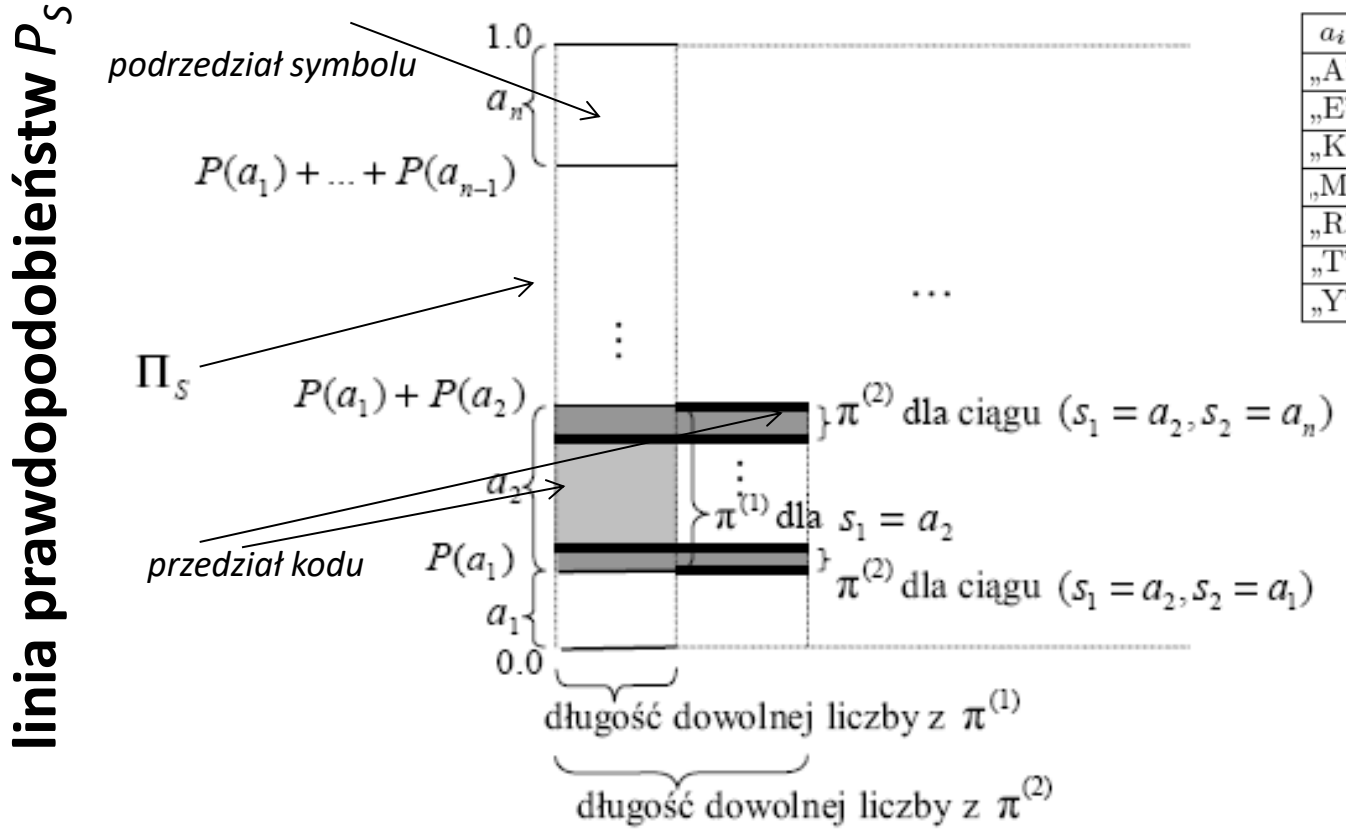
$$\pi_s:\ a_i \in A_S \rightarrow [F(a_{i-1}), F(a_i)) \subset [0,1)$$

- Ogólniej przydzielamy podprzedział w danym przedziale kodowym $[d,g) \subset [0,1)$:

$$\pi_{[d,g)}:\ a_i \in A_S \rightarrow [d+F(a_{i-1})\cdot(g-d), d+F(a_i)\cdot(g-d)) \subset [d,g)$$

Reguła: model identyfikuje ciąg informacji, czyli dystrybuanta, podprzedziały i (nieograniczone) bloki

Modyfikacja przedziału kodowego
(zasadnicza koncepcja **kodu arytmetycznego**)



a_i	$P_s(a_i)$	$F_s(a_i)$	$\pi_s(a_i)$
„A”	2/10	2/10	[0,2/10)
„E”	1/10	3/10	[2/10,3/10)
„K”	1/10	4/10	[3/10,4/10)
„M”	1/10	5/10	[4/10,5/10)
„R”	1/10	6/10	[5/10,6/10)
„T”	2/10	8/10	[6/10,8/10)
„Y”	2/10	1	[8/10,1)

kodowanie to modyfikacja <u>przedziału kodu</u> $\pi^{(i)}= \pi(s_1,s_2,\ldots s_i)= \pi(s_i=a_k, \pi^{(i-1)})$

zasada <u>zawężania przedziału</u>: $\pi^{(i)} \subset \pi^{(i-1)}$

Arytmetyczne kodowanie i dekodowanie

*nowy **przedział kodowy***

- Koder:

$$\pi^{(i)} = \left[D^{(i)}, G^{(i)}\right), \; i = 1,2,\ldots$$

***aktualne granice** przedziału kodowego*

*granice podprzedziału **kodowanego symbolu***

$$D^{(i)} = D^{(i-1)} + \left(G^{(i-1)} - D^{(i-1)}\right) F(a_{k-1})$$

$$G^{(i)} = D^{(i-1)} + \left(G^{(i-1)} - D^{(i-1)}\right) F(a_k)$$

liczba kodowa: dowolna $\mathcal{L}^t \in \pi^{(t)}$

- Dekoder (dwa sposoby):
  - przeskalowanie liczby kodowej do linii prawdopodobieństw

$$\mathcal{L}^t = \mathcal{L}^{(1)} \quad s_i = a_k \Leftrightarrow \mathcal{L}^{(i)} \in \left[F(a_{k-1}), F(a_k)\right]$$

$$\mathcal{L}^{(i+1)} = \frac{\mathcal{L}^{(i)} - F(a_{k-1})}{F(a_k) - F(a_{k-1})}$$

modyfikacja liczby kodowej według π *bez śledzenia modyfikacji*

  - śledzenie przedziału kodowego z ograniczoną dokładnością liczby kodowej

$$s_i = a_k \Leftrightarrow \frac{\left[\mathcal{L}^t\right]^{(i)} - D^{(i)}}{G^{(i)} - D^{(i)}} \in \left[F(a_{k-1}), F(a_k)\right]$$

przybliżona liczba kolejno odczytywana i rzutowana na normalizowaną rekurencyjnie $\pi^{(i)}$

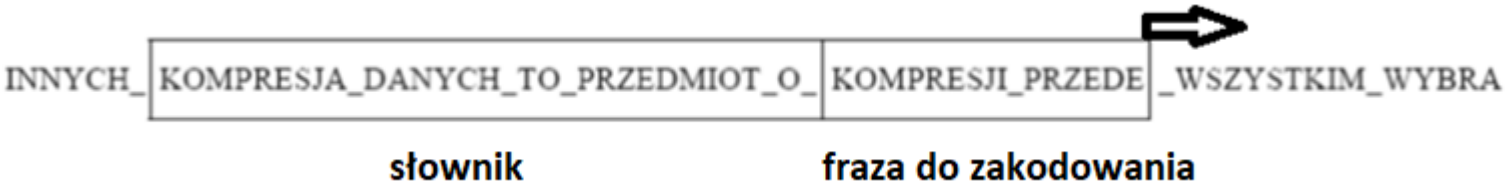
Kodowanie słownikowe – koncepcja indeksów

- Wykorzystanie słownika fraz
- Ciąg symboli o zmiennej długości → indeks (frazy słownika) o (prawie) stałej długości
- Zasada (deterministyczna – zamiast przewidywania, jest wyszukiwanie):

  ciąg symboli == identyczna fraza słownika → indeks frazy
- Efektywność:

  CR = bitowa długość frazy / bitowa długość indeksu (średnio)
  - dłuższe frazy → nieograniczona długość fraz ?
  - krótsze indeksy → mały słownik?
- Klucz: koncepcja słownika
  - statyczna (*a priori*, np. słownik językowy zewnętrzny)
  - póładaptacyjna (słownik na podstawie analizy danych, konieczność kodowania słownika)
  - dynamiczna (budowanie słownika z adaptacją przyczynową)

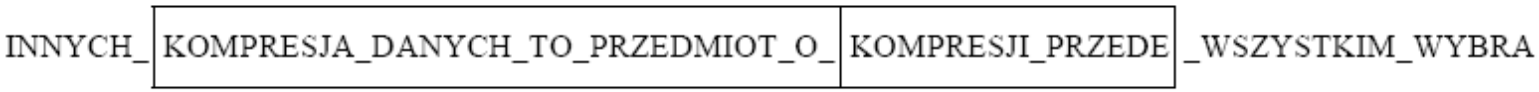
LZ77



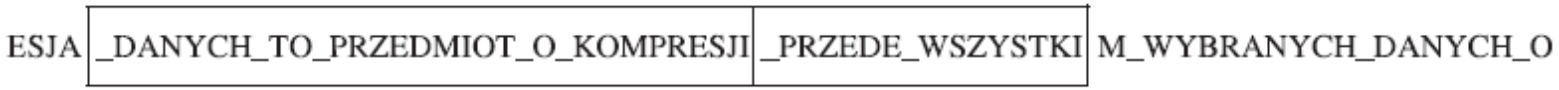
- słownik jako okno przesuwne
  - dynamiczny
  - o ustalonym, ograniczonym rozmiarze (< ciąg zakod. dotąd symboli)
  - struktura nasuwana na strumień ostatnio zakodowanych danych (model przyczynowy)
  - ograniczony rozmiar (bufora) frazy
  - indeks: (wskaźnik położenia frazy w słowniku, długość frazy, pierwszy symbol po kodowanym łańcuchu)
  - po zakodowaniu przesuwamy słownik (o długość frazy +1)

- wady:
  - długi indeks
  - uwzględnienie jedynie ‘najbliższej historii’
  - ograniczona długość kodowanego łańcucha

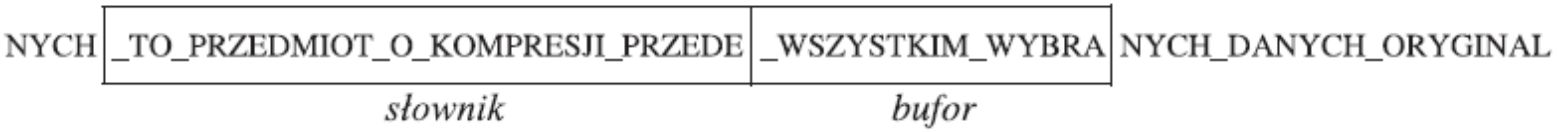
Przykład (LZ77)



Indeks (1,8,”I”)



Indeks (11,6,”E”)



Indeks (1,1,”W”)

Rozmiar indeksu: np. 12bitów wskaźnika (4096 elementów słownika) +

+ 5bitów długości frazy (32 symbole) + 8bitów symbolu = 25 bitów

LZ78

- Nieograniczony słownik zewnętrzny (koncepcja)
  - dynamiczny (adaptacyjny)
  - początkowo pusty z symbolem NULL
  - wpisywane kolejno, coraz dłuższe frazy
  - ‘nieograniczona’! długość frazy
  - możliwy zmienny rozmiar (rosnący indeks)
  - ograniczanie rozmiaru słownika (do części aktywnej)
  - indeks: (wskaźnik położenia frazy w słowniku, pierwszy symbol po kodowanym łańcuchu)
  - po zakodowaniu łańcucha wprowadzamy nową frazę do słownika (łańcuch plus symbol); możliwe są inne frazy jako kombinacja wcześniejszych łańcuchów
- Wady podstawowej realizacji LZ78:
  - początkowo mało efektywny słownik (niewiele pozycji) niewykorzystany rozmiar wskaźnika (krótkie frazy)

Modyfikacja (LZW)

- Wstępne zapełnienie słownika alfabetem

  Efekt: krótsze słowo: (tylko wskaźnik)

- Rozbudowa słownika
  - rosnący indeks
  - szybsza rozbudowa słownika: dodawanie dłuższych fraz:
    - LZMW (łańcuch,łańcuch)
    - LZAP (łańcuch,przedrostki kolejnego łańcucha)
- …

LZW (kodowanie) - przykład

$\mathbf{s}_{WE}$='BOBAS_BOBEK_BOBCIO'

				Słownik	
i	Sekwencja wejściowa	Sekwencja wyjściowa	Pamięć	Indeks	Fraza
0	–	–		[0]-[255]	Kolejne symbole alfabetu źródła
1	„BO”	Ind(B)	„O”	[256]	„BO”
2	„B”	Ind(O)	„B”	[257]	„OB”
3	„A”	Ind(B)	„A”	[258]	„BA”
4	„S”	Ind(A)	„S”	[259]	„AS”
5	„_”	Ind(S)	„_”	[260]	„S_”
6	„B”	Ind(_)	„B”	[261]	„_B”
7	„OB”	256	„B”	[262]	„BOB”
8	„E”	Ind(B)	„E”	[263]	„BE”
9	„K”	Ind(E)	„K”	[264]	„EK”
10	„_”	Ind(K)	„_”	[265]	„K_”
11	„BO”	261	„O”	[266]	„_BO”
12	„BC”	257	„C”	[267]	„OBC”
13	„I”	Ind(C)	„I”	[268]	„CI”
14	„O”	Ind(I)	„O”	[269]	„IO”
15	–	Ind(O)	–	–	–

18*8 (144) to 15*9 (135)

LZW (dekodowanie) - przykład

i	Sekwencja wejściowa	Sekwencja wyjściowa	Pamięć	Słownik: Indeks	Słownik: Fraza
0	–	–	–	[0]-[255]	Kolejne symbole alfabetu źródła
1	„BO”	Ind(B)	„O”	[256]	„BO”
2	„B”	Ind(O)	„B”	[257]	„OB”
3	„A”	Ind(B)	„A”	[258]	„BA”
4	„S”	Ind(A)	„S”	[259]	„AS”
5	„_”	Ind(S)	„_”	[260]	„S_”
6	„B”	Ind(_)	„B”	[261]	„_B”
7	„OB”	256	„B”	[262]	„BOB”
8	„E”	Ind(B)	„E”	[263]	„BE”
9	„K”	Ind(E)	„K”	[264]	„EK”
10	„_”	Ind(K)	„_”	[265]	„K_”
11	„BO”	261	„O”	[266]	„_BO”
12	„BC”	257	„C”	[267]	„OBC”
13	„I”	Ind(C)	„I”	[268]	„CI”
14	„O”	Ind(I)	„O”	[269]	„IO”
15	–	Ind(O)	–	–	–

	Indeksy wejściowe	POPRZEDNI_INDEKS	Wyjściowy łańcuch symboli	PIERWSZY_SYMBOL	Słownik: Indeks	Słownik: Fraza
0	–	–	–	–	[0]-[255]	Kolejne symbole alfabetu
1	Ind(B)	–	„B”	„B”	–	–
2	Ind(O)	Ind(B)	„O”	„O”	[256]	„BO”
3	Ind(B)	Ind(O)	„B”	„B”	[257]	„OB”
4	Ind(A)	Ind(B)	„A”	„A”	[258]	„BA”
5	Ind(S)	Ind(A)	„S”	„S”	[259]	„AS”
6	Ind(_)	Ind(S)	„_”	„_”	[260]	„S_”
7	256	Ind(_)	„BO”	„B”	[261]	„_B”
8	Ind(B)	[256]	„B”	„B”	[262]	„BOB”
9	Ind(E)	Ind(B)	„E”	„E”	[263]	„BE”
10	Ind(K)	Ind(E)	„K”	„K”	[264]	„EK”
11	261	Ind(K)	„_B”	„_”	[265]	„K_”
12	257	261	„OB”	„O”	[266]	„_BO”
13	Ind(C)	257	„C”	„C”	[267]	„OBC”
14	Ind(I)	Ind(C)	„I”	„I”	[268]	„CI”
15	Ind(O)	Ind(I)	„O”	„O”	[269]	„IO”

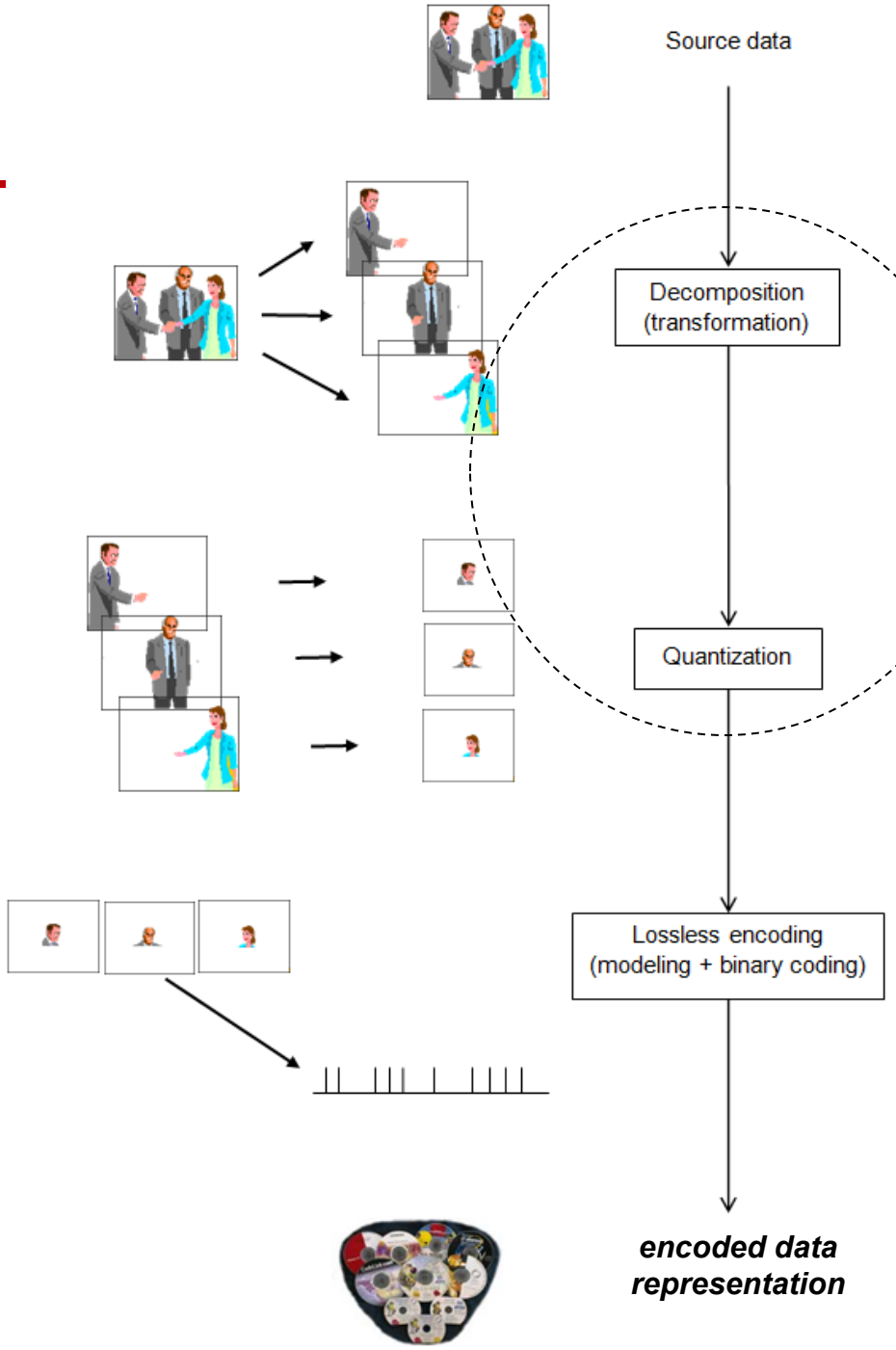
Wyniki

	dokument.doc	RAND.txt	REGULAR_small.txt	REGULAR.txt	TPDE.doc	motylek.avi	Cynthia.bmp	
Długość (kB)	1320	1420	1420	1900	707	712	1417	Średnia (bit rate)
dyn LZW (4096)	5,92	9,59	1,38	1,38	2,77	10,47	0,59	4,59
dyn LZWM (4096)	5,62	9,86	1,01	1,01	2,22	10,5	0,57	4,4
dyn LZAP (4096)	5,82	9,82	1,52	1,53	2,62	10,48	1,03	4,68
dyn LZW (8192)	6,02	9,71	1,27	1,28	2,7	10,74	0,56	4,61
dyn LZWM (8192)	5,74	10,09	0,99	0,98	2,16	10,89	0,54	4,48
dyn LZAP (8192)	5,89	10,05	1,38	1,39	2,46	10,85	0,98	4,71
dyn LZW (16384)	6,06	9,65	1,18	1,18	2,62	10,79	0,54	4,57
dyn LZWM(16384)	5,86	10,24	0,95	0,96	2,16	11,09	0,52	4,68
dyn LZAP (16384)	5,96	10,18	1,29	1,29	2,39	11,05	0,89	4,72
dyn LZW (32768)	5,92	9,38	1,1	1,2	2,66	10,49	0,55	4,47
dyn LZWM(32768)	6,19	10,28	0,94	0,94	2,24	11,02	0,52	4,59
dyn LZAP (32768)	6,23	10,21	1,21	1,22	2,34	10,97	0,78	4,71
stat LZW	6,31	10,22	1,48	1,48	2,95	11,16	0,63	4,83
stat LZWM	5,96	10,36	1,11	1,1	2,36	11,19	0,61	4,67
stat LZAP	6,22	10,33	1,54	1,43	2,84	11,19	1,1	4,95
RAR	1,91	7,21	1,08	1,07	1,36	6,54	0,58	2,82
ZIP	4,04	7,08	1,1	1,11	1,63	7,7	0,76	3,35
Huffman	6,29	6,96	3,6	3,6	4,23	7,81	1,74	4,89
Arytmetyczny	6,28	6,95	3,54	3,54	4,42	7,75	1,76	4,89

Zastosowania kodeków słownikowych (archiwizery, bezstratna kompresja obrazów)

- PNG: prediction+deflate
- Deflate (ZLIB): LZ77 (32kB and 258) + Huffman code
- LZ77: *LHA* (*LHarc*), *zip, gzip, ARJ, RAR, 7-Zip* etc.
- LZW + Huffman code: compress, PKArc, WinZip
- GIF (LZW)
- others

Zasadnicza koncepcja kompresji stratnej (na przykładzie obrazów)

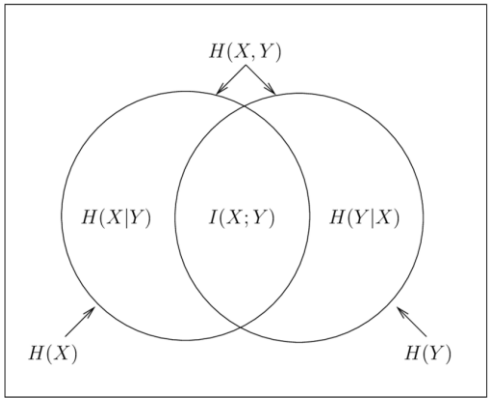


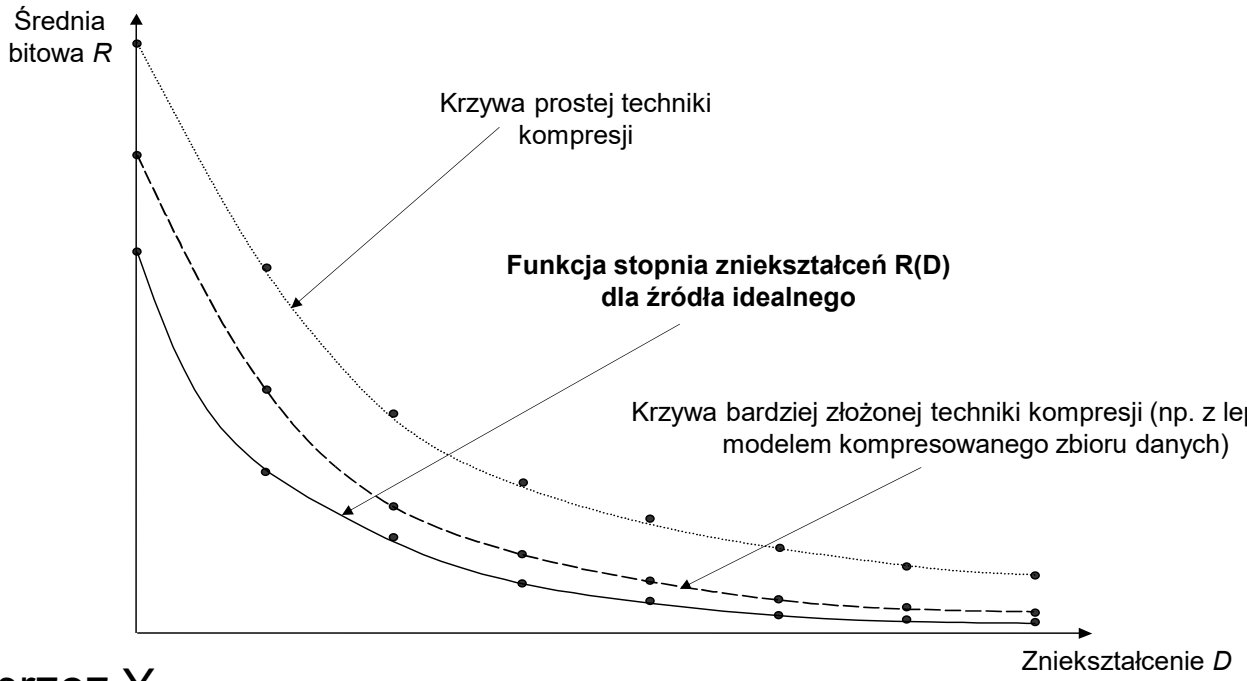
- *bloki, komponenty, wektory ...*
- ***przekształcenia w nową dziedzinę***
- *rzadkie reprezentacje upakowanej informacji*

- *selekcja informacji z **kontrolą jakości***
- *koncepcja kwantyzacji zintegrowana z formami dekompozycji danych*
- *jakość, wiarygodność, istotność kompresowanych informacji (najlepiej bezstratnie)*

Podstawy teorii stopnia zniekształceń źródeł informacji

- Krzywa R-D





- Przybliżenie X przez Y

$$D = \sum_{i=0}^{N_1-1}\sum_{j=0}^{N_2-1} d(x_i,y_j)P(x_i,y_j) = \sum_{i=0}^{N_1-1}\sum_{j=0}^{N_2-1} d(x_i,y_j)P(x_i)P(y_j \mid x_i)$$

- Optymalizacja

$$R(D) = \min_{\{P(y_j|x_i)\}\in\Lambda} I(X;Y)$$

$$\Lambda = \left\{\{P(y_j \mid x_i)\} \quad \text{takich, że} \quad D(\{P(y_j \mid x_i)\}) \le D^t\right\}$$

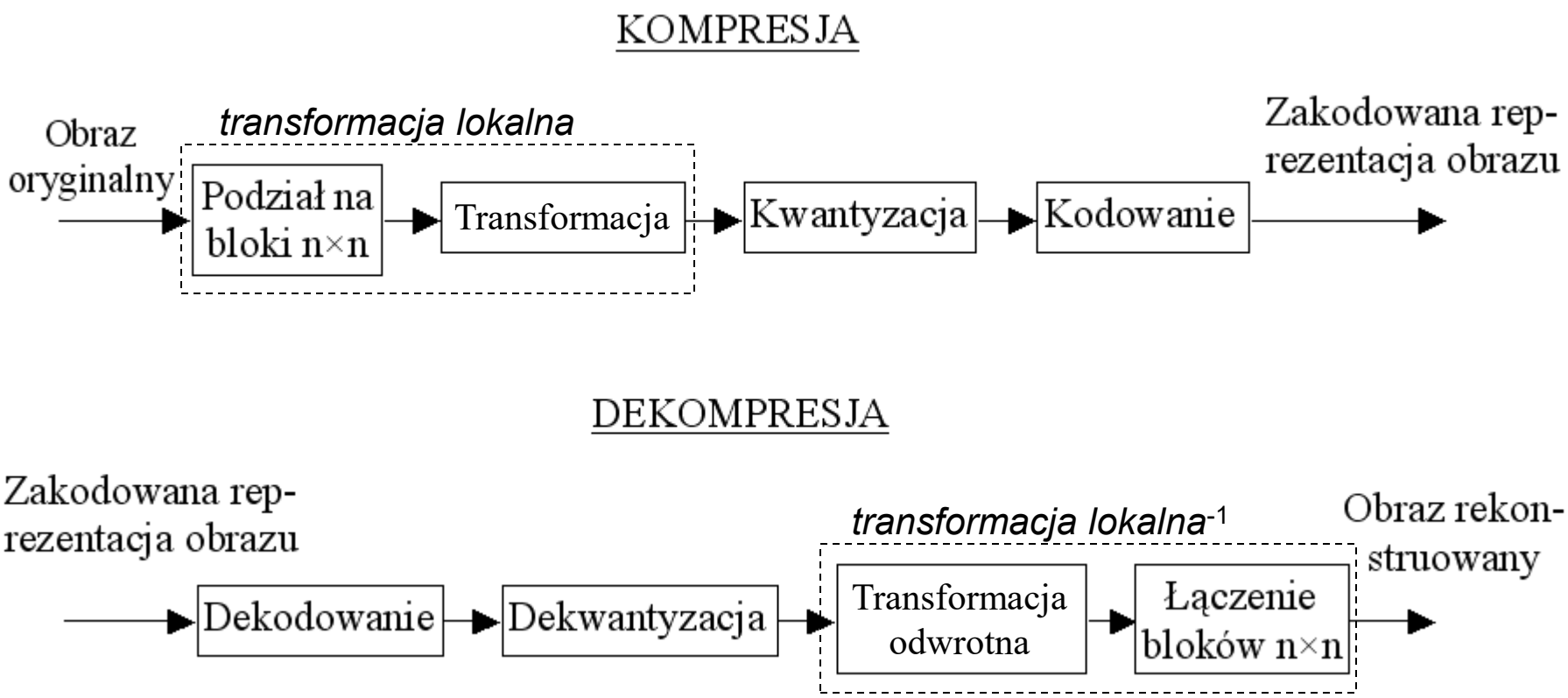
Informacja wzajemna:

$$I(X;Y) = \sum_{i=1}^{N_1}\sum_{j=1}^{N_2} P(x_i,y_j)\log\left[\frac{P(x_i,y_j)}{P(x_i)P(y_j)}\right]$$

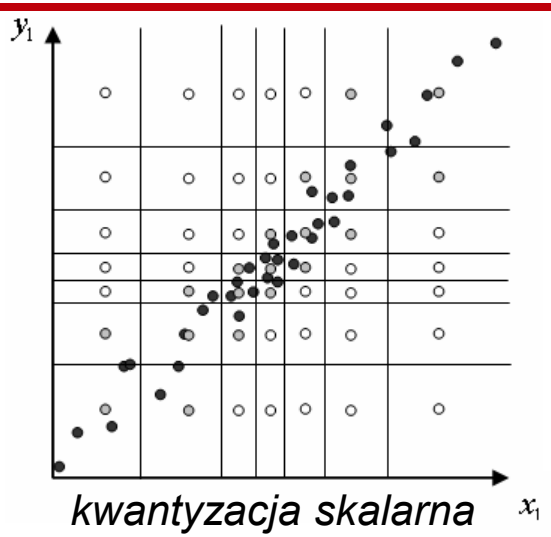
$$I(X;Y) = H(X) + H(Y) - H(X,Y) = H(X) - H(X \mid Y) = H(Y) - H(Y \mid X)$$

entropia produktowa (łączna); entropie warunkowe

Paradygmat TK

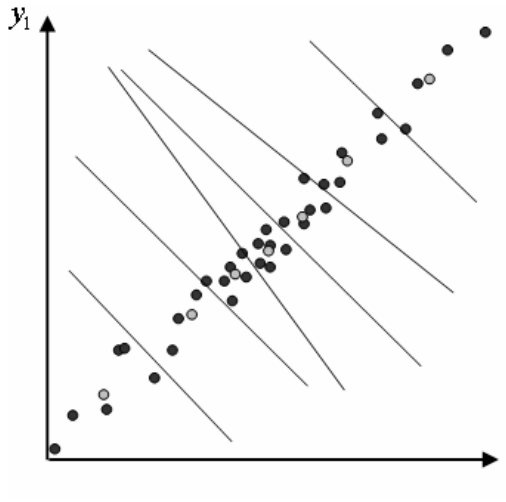


Koncepcja kodowania: uproszczenie kwantyzacji

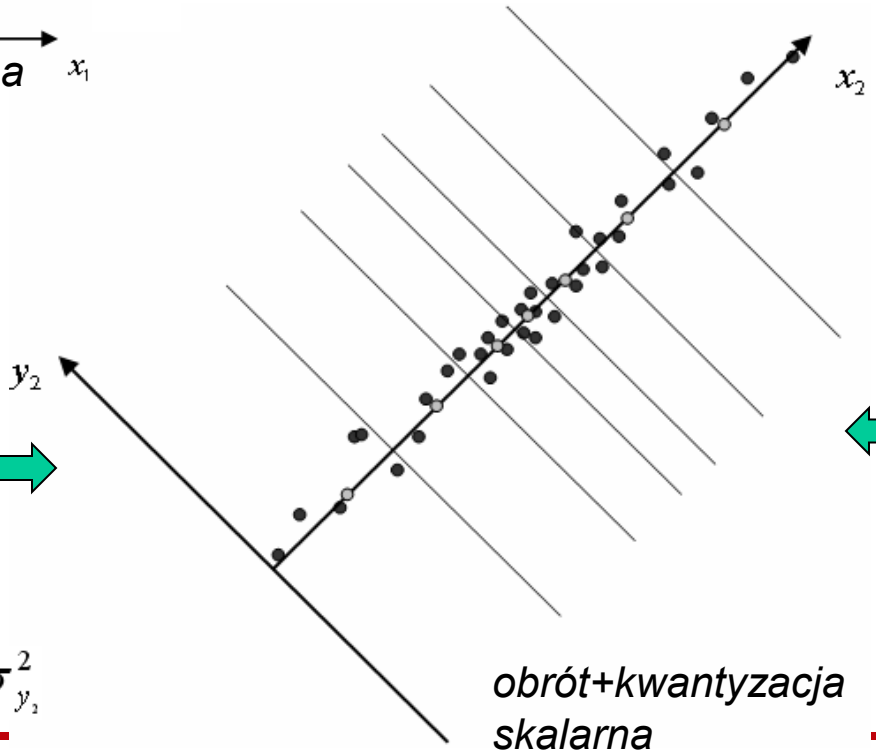


kwantyzacja skalarna

$$\sigma_{x_1}^2 \cong \sigma_{y_1}^2$$



kwantyzacja wektorowa w 2D



Transformacja

Upakowanie

$$\sigma_{x_2}^2 >> \sigma_{y_2}^2$$

$$\sigma_{x_1}^2 + \sigma_{y_1}^2 = \sigma_{x_2}^2 + \sigma_{y_2}^2$$

obrót+kwantyzacja skalarna

KLT (T, Karhunena-Loeve) – quasi-optymalna, ale niekorzystna w implementacji

- Wyznaczamy macierz kowariancji
- Diagonalizujemy macierz kowariancji
- Utrzymujemy zbiór wartości własnych i wektorów własnych
- Ustalamy zbiór wektorów własnych jako bazę transformacji
- Uzyskujemy zdekorelowany sygnał

$$\mathbf{R}_{\mathbf{XX}} = \mathbf{X}\mathbf{X}^T = \boldsymbol{\psi}(\mathbf{x}\mathbf{x}^T)\boldsymbol{\psi}^T = \mathrm{diag}(\lambda_1, \lambda_2, \ldots, \lambda_n)$$

Transformacja DCT (2W) – efektywna, łatwa w implementacji, ‘lokalna’ poprzez bloki

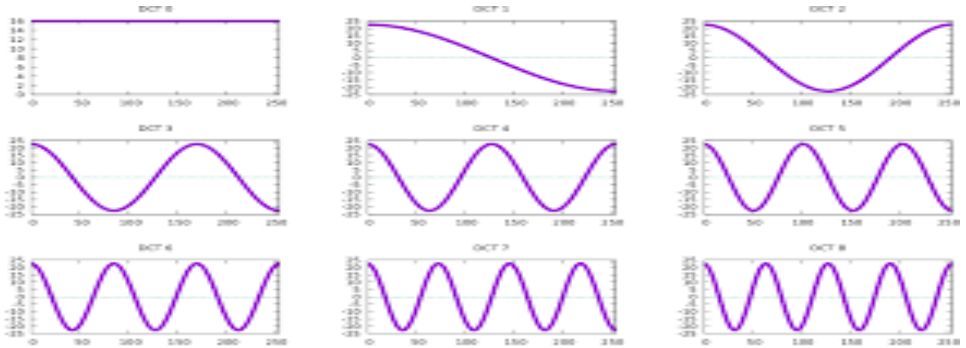
prosta

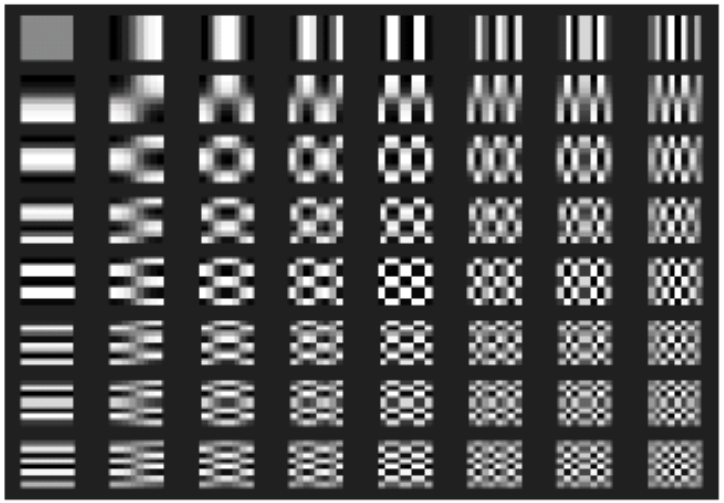
$$k(u,v) = \frac{1}{\sqrt{2n}} c_u c_v \sum_{x=0}^{n-1} \sum_{y=0}^{n-1} f(x,y) \cos \frac{(x+\frac{1}{2})u\pi}{n} \cos \frac{(y+\frac{1}{2})v\pi}{n}$$

$$c_u, c_v = \frac{1}{\sqrt{2}} \quad \text{dla } u, v = 0; \qquad c_u, c_v = 1 \quad \text{w p.p.}$$

odwrotna

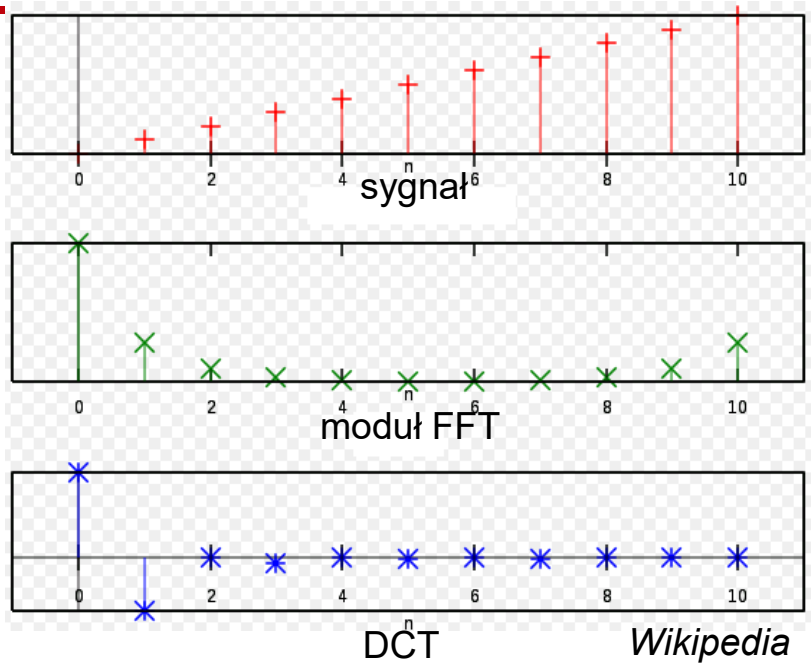
$$f(x,y) = \frac{1}{\sqrt{2n}} \sum_{u=0}^{n-1} \sum_{v=0}^{n-1} c_u c_v k(u,v) \cos \frac{(x+\frac{1}{2})u\pi}{n} \cos \frac{(y+\frac{1}{2})v\pi}{n}$$

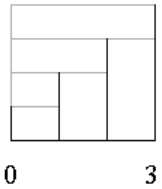




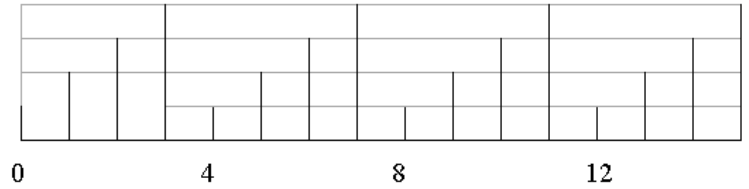
Stosowana m.in. w JPEG

DCT czy FFT

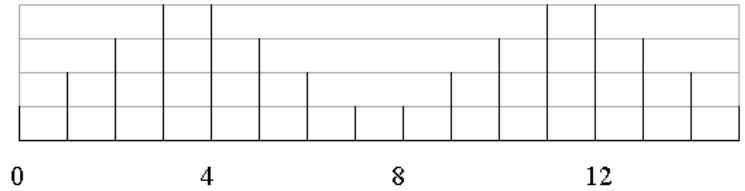




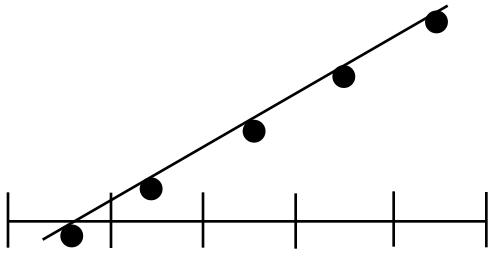
oryginał



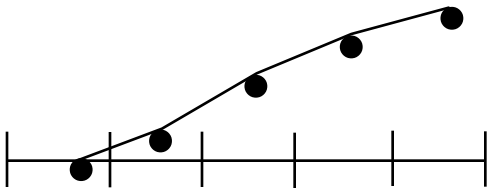
rozwinięcie sygnału w DFT



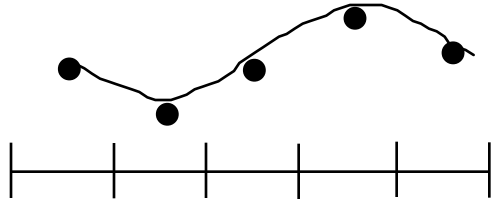
rozwinięcie sygnału w DCT



oryginał



rekonstrukcja z kilku współczynników DCT



rekonstrukcja z kilku współczynników FFT

Selekcja próbek

blok obrazu

109	101	91	79	80	85	84	89
87	87	89	99	98	91	91	93
100	102	109	127	112	99	87	84
84	89	87	88	83	90	100	94
100	101	108	112	110	114	113	109
103	103	95	89	89	89	82	86
84	97	100	100	108	103	106	112
111	115	111	113	107	100	109	107

DCT

784.2	6.630	-6.45	-5.46	2.25	0.000	-3.82	-2.80
-34.9	10.89	-0.46	5.318	4.209	3.641	3.270	-2.62
0.726	10.11	12.68	-0.72	2.956	-5.22	-0.91	3.229
-19.8	2.843	21.02	7.299	-5.43	-7.31	7.198	2.446
9.000	-2.50	20.10	1.802	-8.00	5.061	-0.48	1.331
-21.7	-3.61	8.453	-1.76	-5.13	-3.66	0.128	0.899
2.405	37.31	-0.16	3.307	2.564	0.964	2.072	-3.61
40.72	-3.57	-18.2	-1.70	3.892	0.977	2.199	-0.53

strefowa selekcja próbek

784	7	-6	-5	2	0	0	0
-35	11	0	5	0	0	0	0
1	10	13	0	0	0	0	0
-20	3	0	0	0	0	0	0
9	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0

progowa selekcja próbek (próg=10)

784	0	0	0	0	0	0	0
-35	11	0	0	0	0	0	0
0	10	13	0	0	0	0	0
-20	0	21	0	0	0	0	0
0	0	20	0	0	0	0	0
-22	0	0	0	0	0	0	0
0	37	0	0	0	0	0	0
41	0	-18	0	0	0	0	0

rekonstrukcje bloku

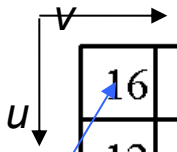
99	96	93	90	88	86	85	85
97	96	94	93	92	90	89	89
96	97	98	100	99	97	95	94
95	97	101	104	104	101	99	98
93	96	100	103	102	99	97	97
93	96	99	100	98	96	96	97
100	102	103	103	100	99	101	103
109	109	109	108	106	105	108	112

106	99	89	81	77	80	85	89
87	89	91	93	95	97	97	98
100	106	114	118	114	103	90	82
85	85	85	86	88	91	94	96
100	102	104	107	110	112	113	113
102	99	94	90	88	88	89	91
90	93	99	104	108	110	109	109
114	112	108	106	104	105	106	107

Szczegóły JPEG

RGB to YCrCb:

$$Y = 0,2989 \cdot R + 0,5866 \cdot G + 0,1145 \cdot B$$
$$Cr = 0,5 \cdot R - 0,4183 \cdot G - 0,0816 \cdot B$$
$$Cb = -0,1687 \cdot R - 0,3312 \cdot G + 0,5 \cdot B$$



16	11	10	16	24	40	51	61
12	12	14	19	26	58	60	55
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

składowa stała

Percepcyjna tablica kwantyzacji (luminancja)

17	18	24	47	99	99	99	99
18	21	26	66	99	99	99	99
24	26	56	99	99	99	99	99
47	66	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

tablica kwantyzacji dla chrominancji

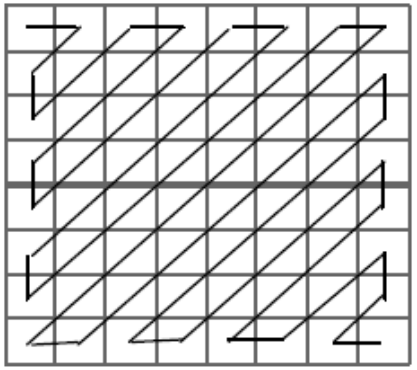
Kwantyzacja/selekcja:

dzielenie współczynników DCT k(u,v) przez element tablicy kwantyzacji q(u,v) i zaokrąglanie do liczby całkowitej

$$\tilde{k}(u,v) = \left\lfloor \frac{k(u,v)}{q(u,v)} \right\rfloor$$

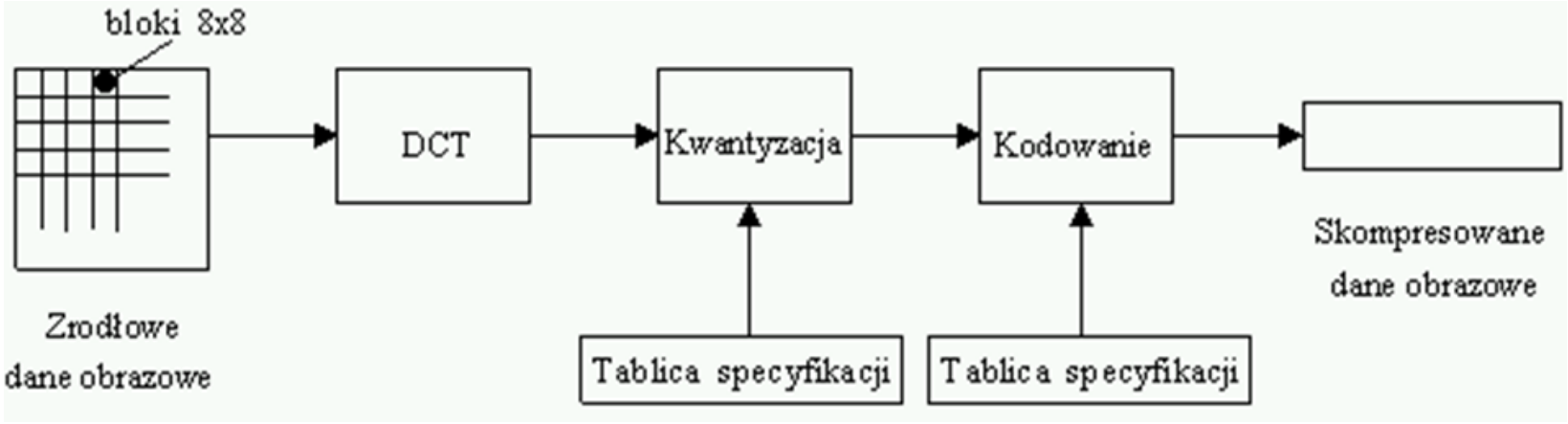
Kodowanie:

przeglądanie zygzak, różnicowo składowa stała bloków (predykcja), kod Huffmana-RLE lub kod arytmetyczny



Standard JPEG

- Podstawowy proces kodowania (ang. baseline process).
- Rozszerzony , bazujący na DCT proces kodowania (ang. extended DCT- based process).
- Bezstratny proces kodowania (ang. lossless process).
- Hierarchiczny proces kodowania (ang. hierarchical process).



Paradygmat JPEG

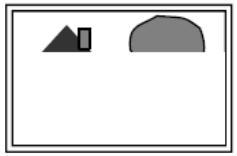
Kompresja/Dekompresja

Opcje kodera:
- przestrzeń kolorów
- kwantyzacja
- entropijne kodowanie
- przetwarzanie wstępne

Brak opcji dekodera (ustalony w koderze porządek i sposób rekonstrukcji, bez żadnych możliwości wyboru)

Rozszerzenia kodeka JPEG (progresja, podpróbkowanie)

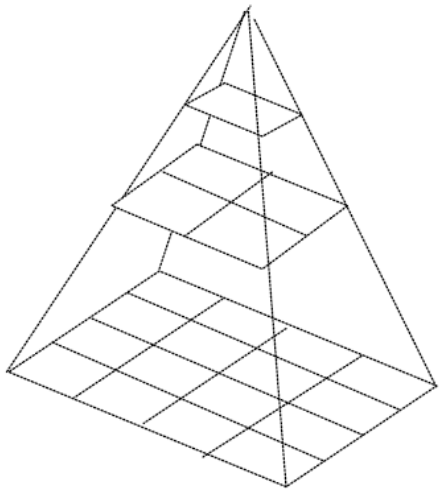
- Progresja kodowania



sekwencyjnie

z progresją

- Kodowanie hierarchiczne (z podpróbkowaniem)



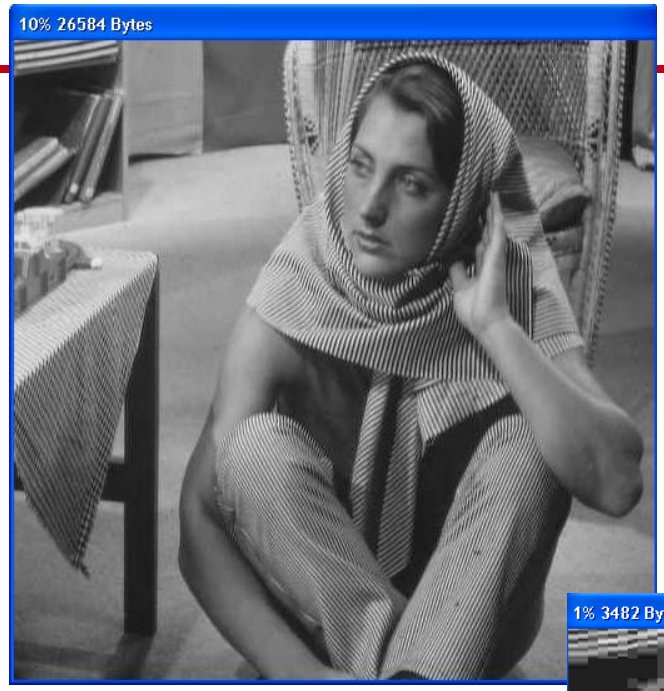
Efekty JPEGowe

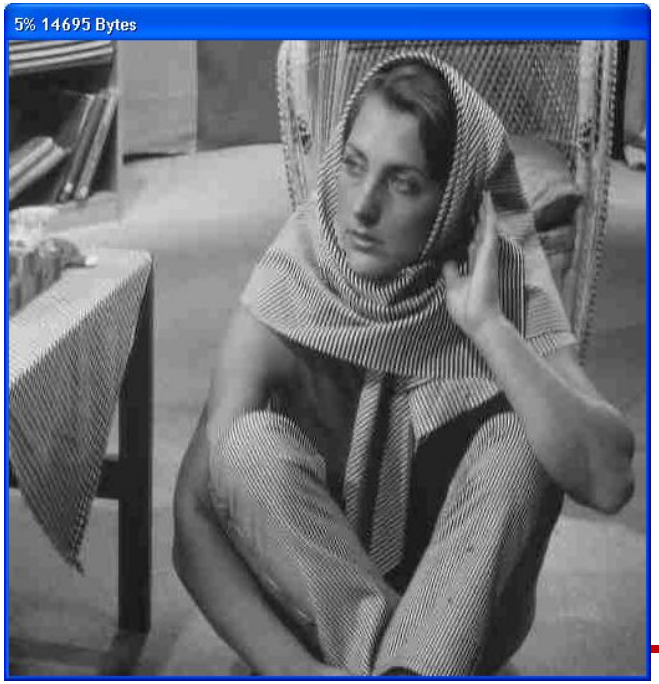
Oryginał

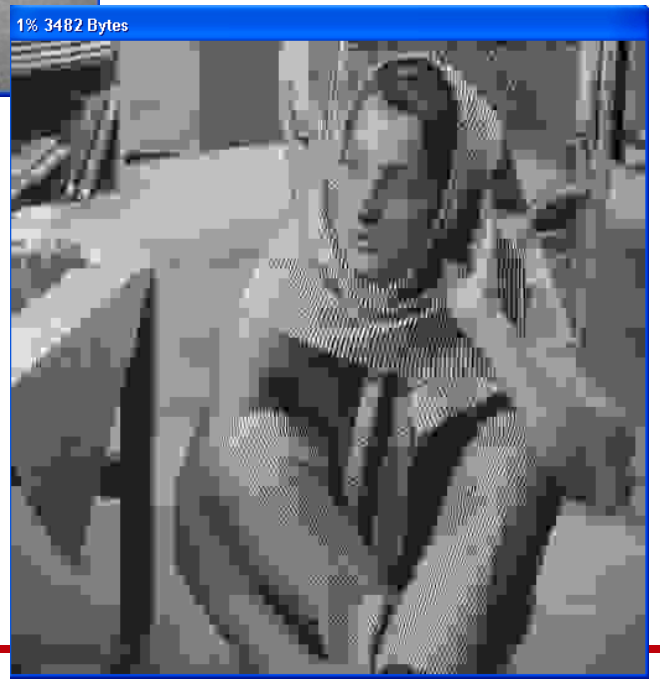
12:1

43:1

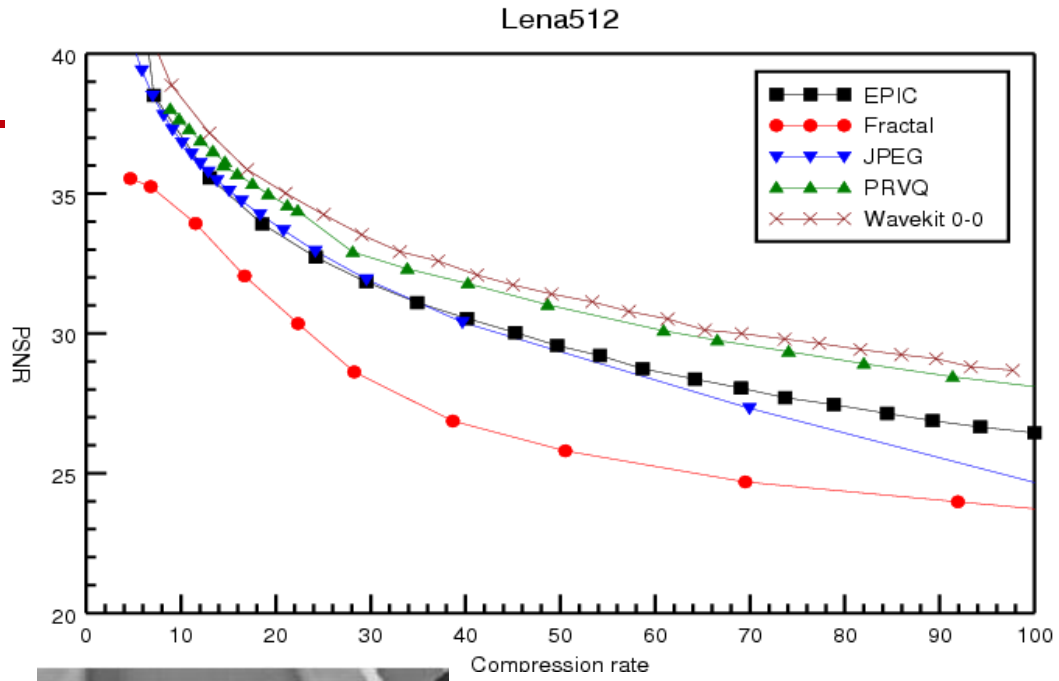
Efekty JPEGowe

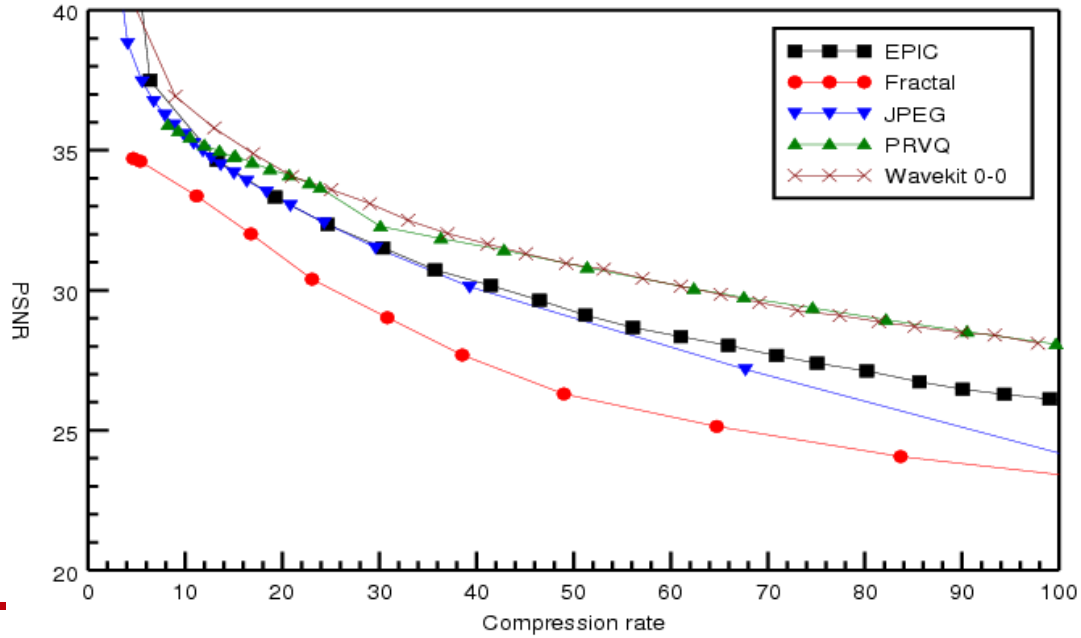






Testy porównawcze



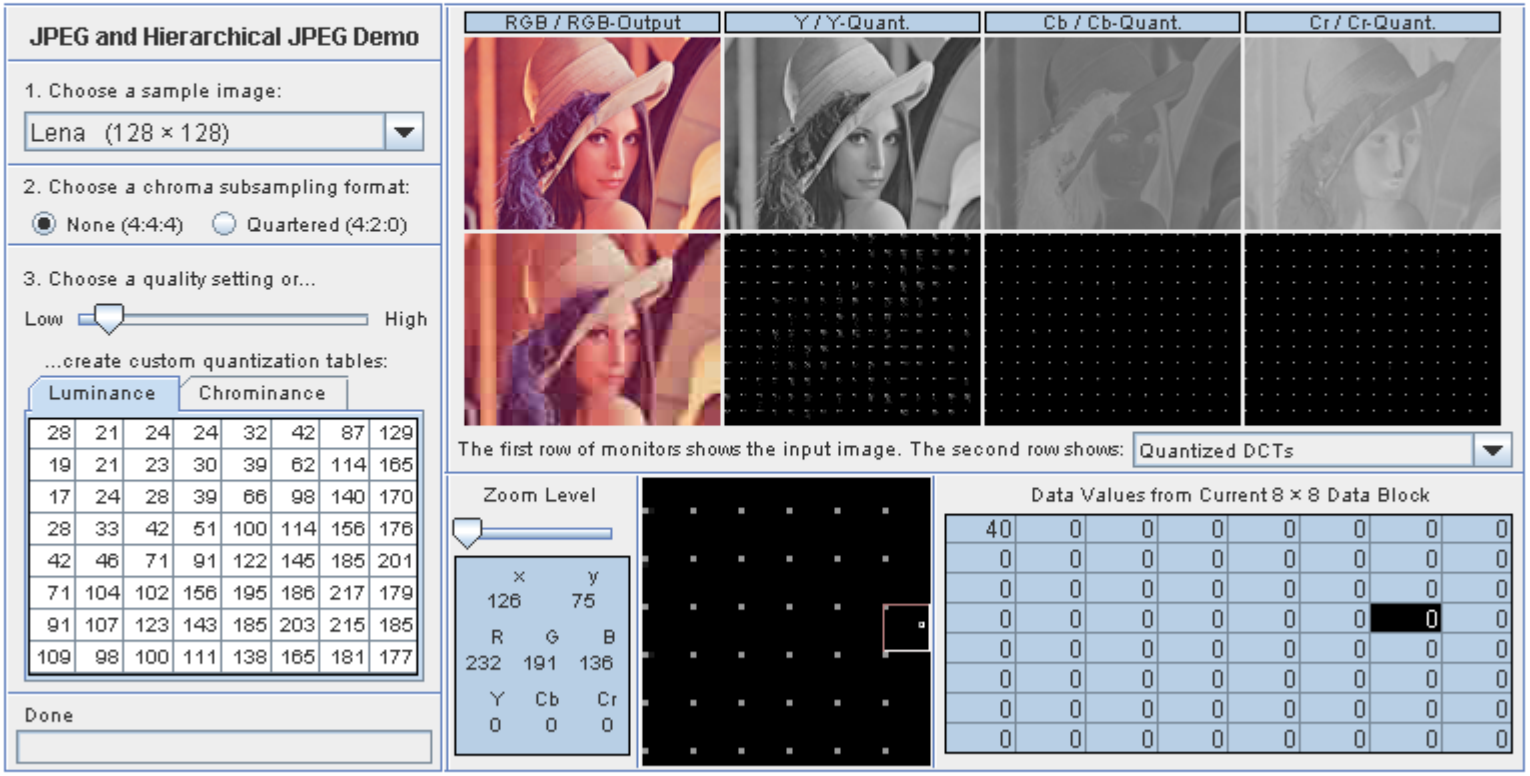


Polecam zabawę z JPEG

http://www.sfu.ca/~cjenning/toybox/hjpeg/index.html

Transform of each channel. Notice that you can make out the patterns of little 8x8 boxes. Remember that the upper-left corner of those boxes is the low-frequency part, the other corner is high-frequency. Brighter pixels mean more information, *i.e.* more of the image was of that kind of frequency.

5. Now choose the next drop down item (Quantized DCTs). Whoosh! The lower corners of each of the 8x8 boxes just went dark. The values in each box were just divided by the values in the corresponding cells (throwing away any remainders) of the quantization tables (Step 3), so they now have less information. The next drop down box shows the reconstructed DCTs, *i.e.* what we get back when we decompress the image by multiplying by the values in the tables. Compare the original DCT of the colour planes to the reconstructed versions. A *lot* of colour information has been lost, but it is not very noticeable in the final decompressed picture.

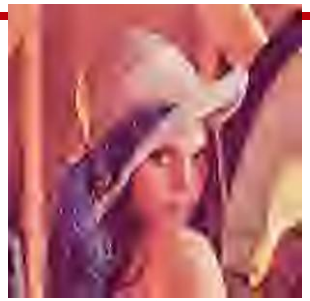


: applet requires that your browser support at least the Java 2 runtime. If the applet above doesn't work properly, this is almost certainly the blem. You can download the latest Java runtime from Sun to correct this.

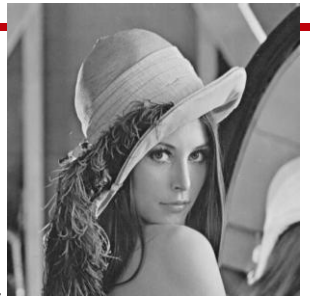
Paradygmat JPEG2000

Opcje kodera:
- format danych: obrazy ze skalą szarości i obrazy binarne
- dzielenie na części (podobrazy, ang. *tiling*)
- kompresja stratna-bezstratna
- pozostałe jak w paradygmacie JPEG

Kompresja/Dekompresja



skalowanie jakości

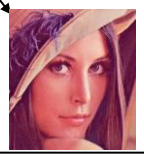


odtworzenie wybranego komponentu

kompresja bezstratna

skalowanie rozdzielczości

rekonstrukcja wybranego ROI



progresja z wybranym ROI

Opcje dekodera:
- rozdzielczość obrazu
- jakość obrazu
- ustalanie dokładnego rozmiaru reprezentacji
- bitowej dekodowanej informacji
- składniki (komponenty)
- regiony zainteresowań
- kompresja stratna-bezstratna

Dodatkowe możliwości schematu kompresji:
- zwiększona odporność na zakłócenia
- adnotacje o prawach autorskich
- XML-owa struktura informacji dodatkowych
- osadzony strumień bitowy (progresywne dekodowanie i skalowalność SNR)
- reprezentacja wielorozdzielcza
- bezpośredni dostęp do kodowanego strumienia i przetwarzanie
- interaktywny protokół (JPIP) – wzrost użyteczości
- kompresja danych 3D etc.

Porównanie JPEG2000 vs JPEG

(rozmycie vs efekty blokowe)

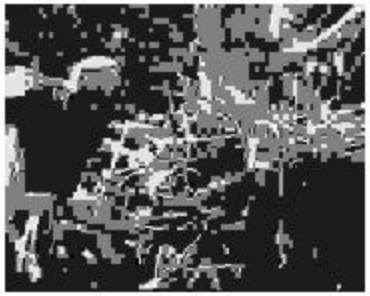
Testy JPEG2000 – JPEG

JPEG vs JPEG2000

Compression Ratio => 310:1 (top row), 130:1 (bottom row)

Original Image
(600x480x24)

BMP (844 KB)



JPEG (3 KB, 2751 bytes)

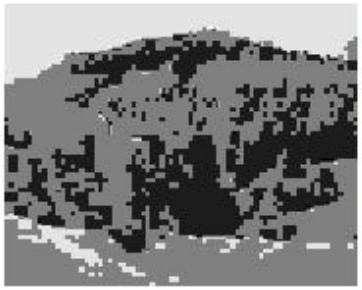
JPEG2000 (3 KB, 2781 bytes)

JPEG (7 KB, 6220 bytes)

Compression Ratio => 350:1 (top row), 215:1 (bottom row)

Original Image
(600x480x24)

BMP (844 KB)



JPEG (3 KB, 2391 bytes)

JPEG2000 (3 KB, 2406 bytes)

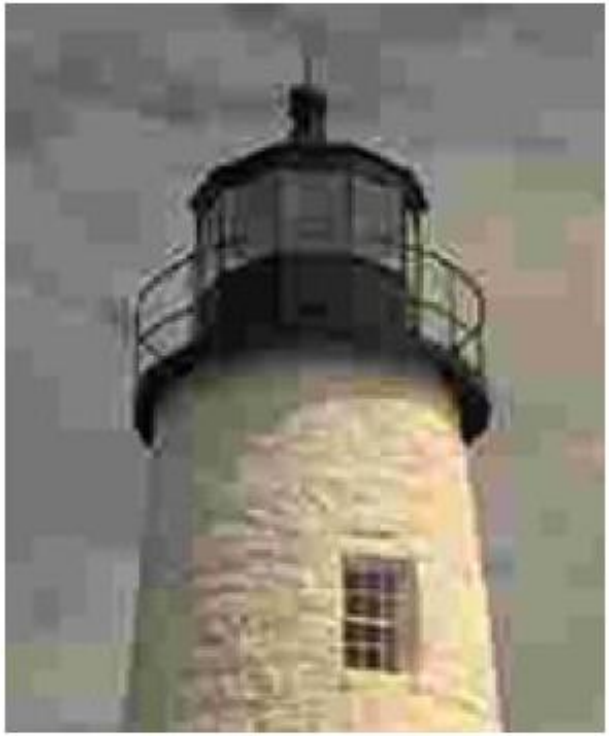
JPEG (4 KB, 3994 bytes)

JPEG2000 (4 KB, 3990 bytes)

http://www.imagepower.com/technology/jpeg2000/compare/index.html

JPEG vs JPEG2000

(a) Original image



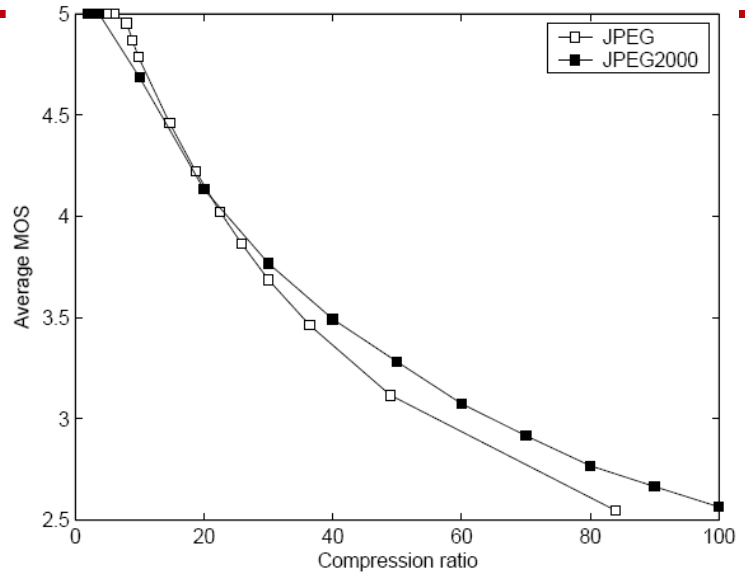
(b) JPEG-encoded image

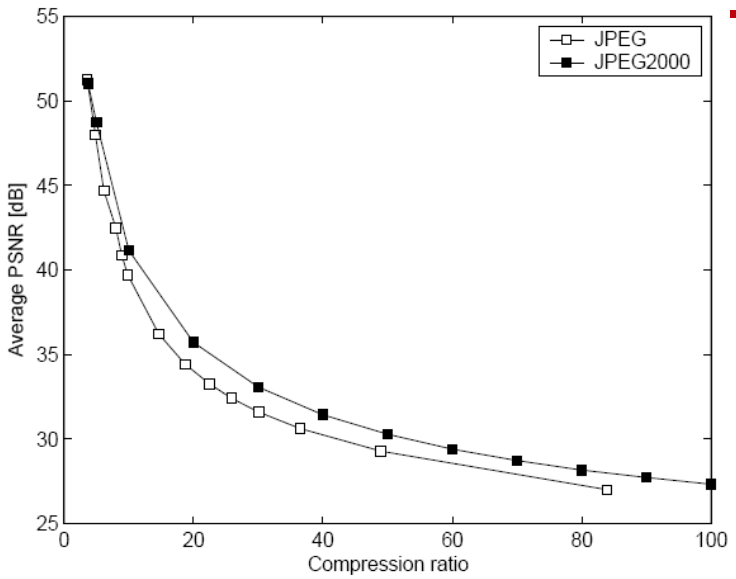
(c) JPEG2000-encoded image

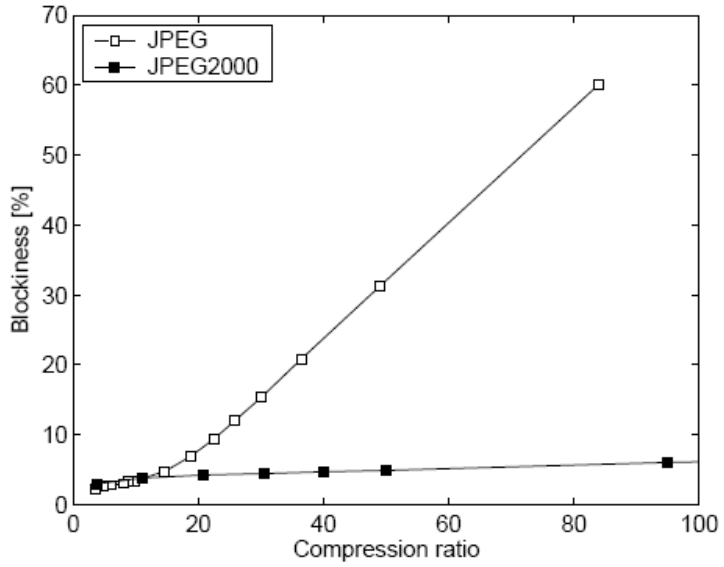
F. Ebrahimi, M. Chamik, S. Winkler, JPEG vs. JPEG2000: An Objective Comparison of Image Encoding Quality, *Proc. SPIE Applications of Digital Image Processing*, 5558:300-308, 2004.

JPEG vs JPEG2000

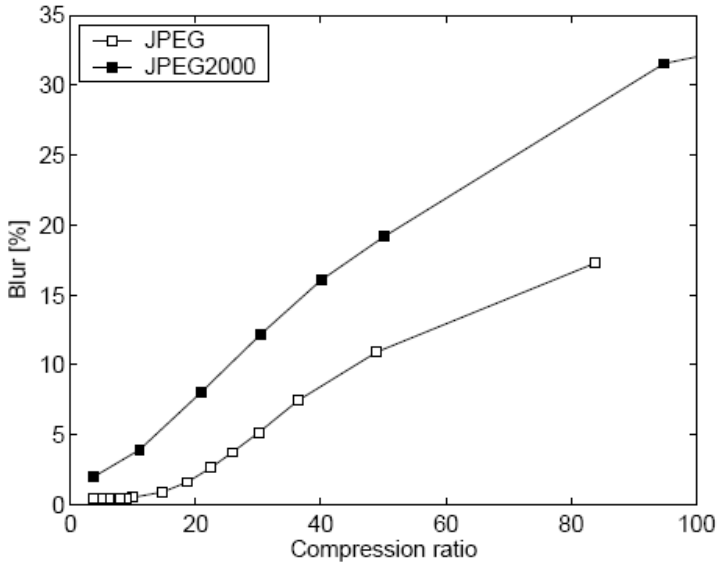
$$PSNR = 10\log\frac{MN\cdot[\max_{m,n}\{f(m,n)\}]^2}{\sum_{m,n}[f(m,n)-\tilde{f}(m,n)]^2}$$







(a) Blockiness

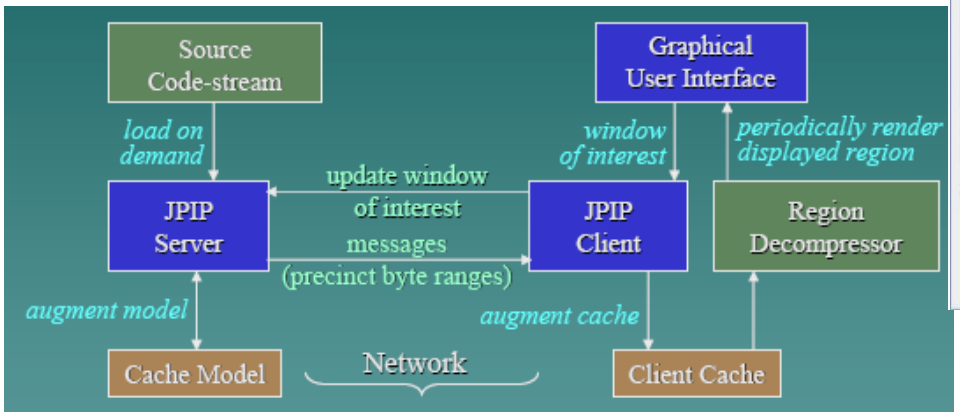


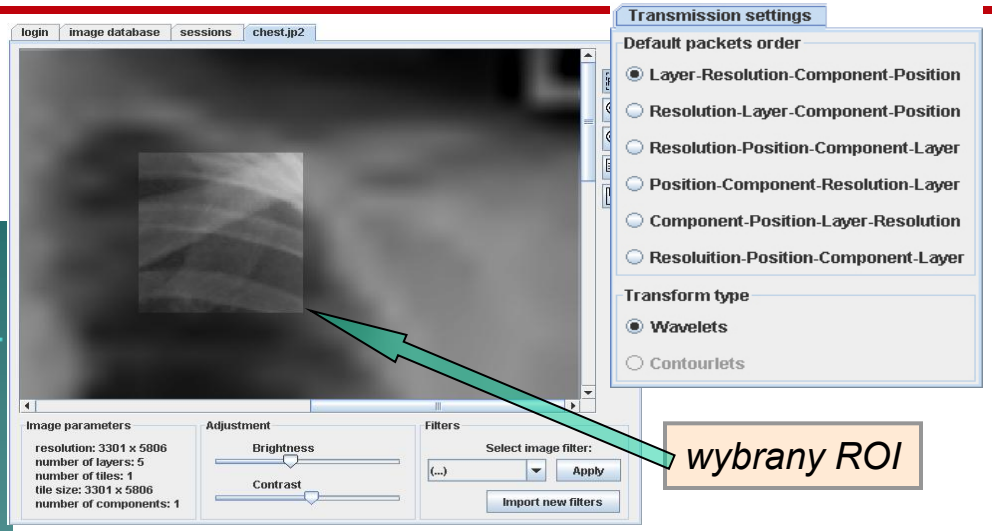
(b) Blur

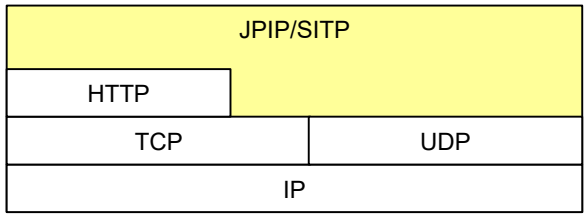
Testy efektywności (wartości PSNR)

Technika kompresji	Lena		Barbara		Goldhill	
	0.25	***0.5***	***0.25***	***0.5***	***0.25***	***0.5***
PVQ	34.08	37.18	30.12	33.89	30.86	33.46
ECTCQ	33.90	37.10	29.66	33.51	30.67	33.35
KVQ-TCQ	34.31	37.67	-	-	-	-
SPIHT	34.11	37.21	29.36	33.07	30.56	33.13
ECECOW	34.81	37.92	28.85	32.69	-	-
RDE	34.20	37.20	29.10	33.10	-	-
SFQ	34.35	37.41	29.67	33.51	30.71	33.37
C/B	34.57	37.52	28.75	32.64	30.80	33.53
PACC	34.53	37.51	28.65	32.54	30.84	33.51
PC-AUTQ	34.46	37.56	-	-	30.78	33.46
EQ	34.57	37.68	-	32.87	30.76	33.44
TCSFQ=SFQ+TCQ+ ECECOW	34.76	37.87	**30.60**	**34.48**	30.98	**33.75**
FD	**34.89**	**38.02**	29.21	33.06	-	-
SC&CE	34.72	37.75	28.76	32.80	-	-
MBWT	34.60	37.58	30.17	33.95	**30.99**	33.60
Stack-run	33.80	36.89	28.90	32.66	-	-
Koder indeksowy	33.44	36.59	-	-	-	-
Koder morfologiczny	34.12	37.18	-	-	30.53	33.15
JPEG2000	34.23	37.32	29.35	33.11	30.70	33.28

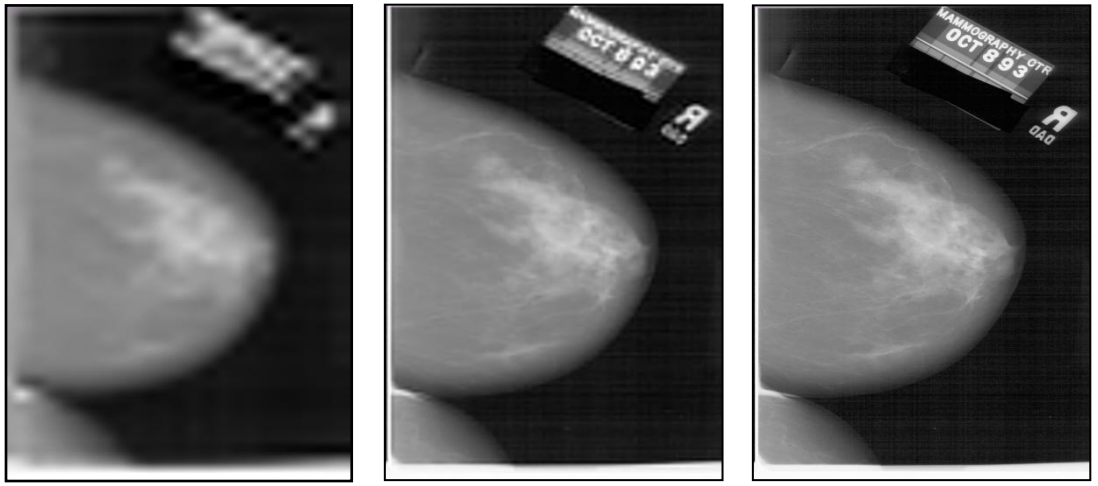
JPIP (przykłady)





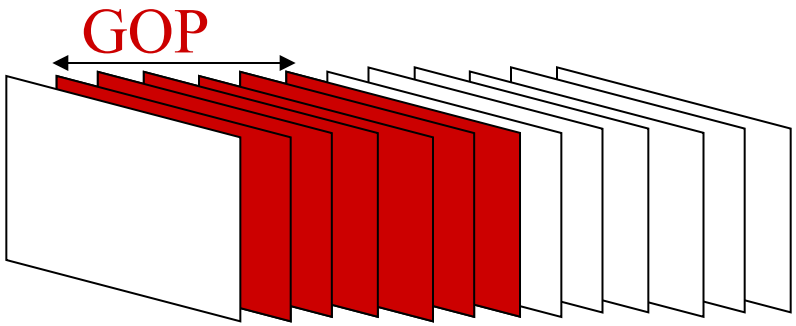


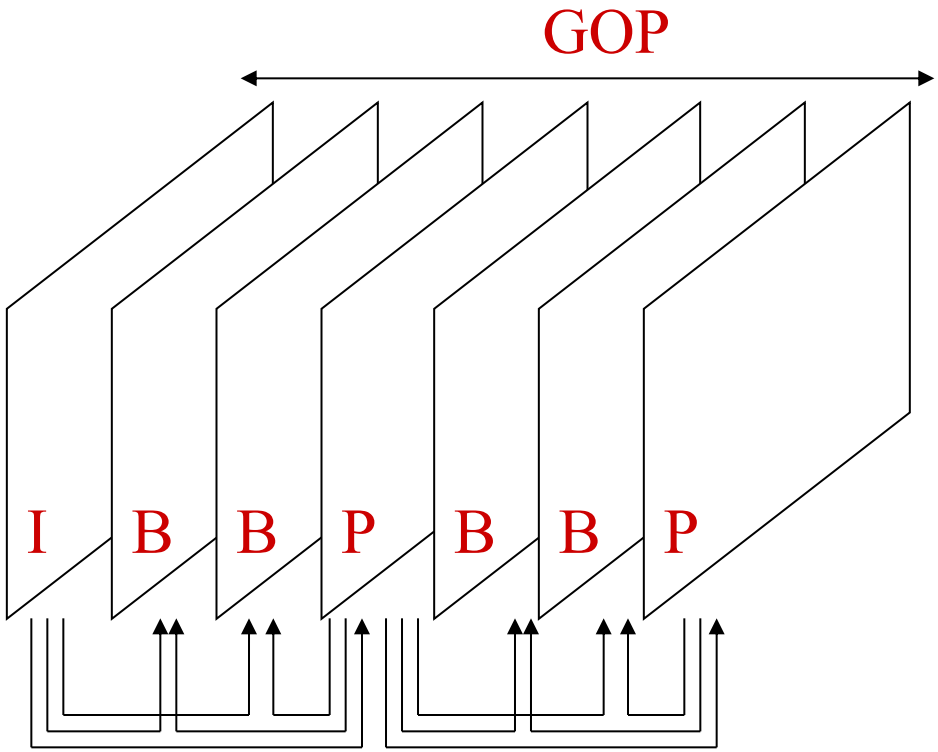
protokół



0.5% 2% 100%

Predykcja INTER (międzyobrazowa) w MPEG

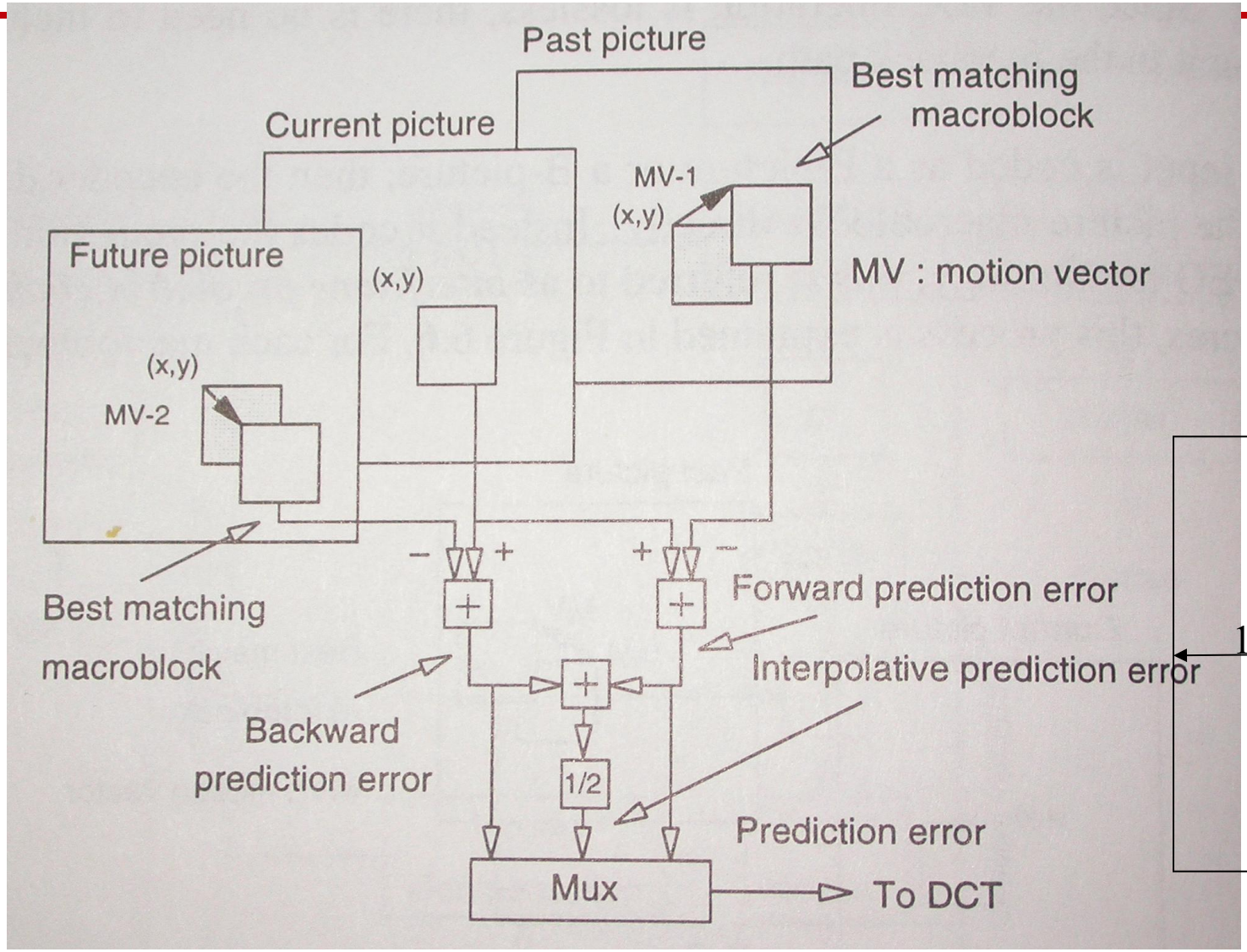


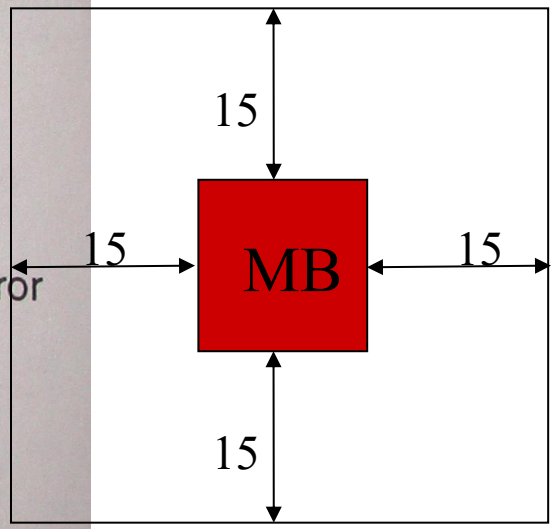


Porządek kodowania: I P B B P B B

Porządek wyświetlania: I B B P B B P

Estymacja ruchu

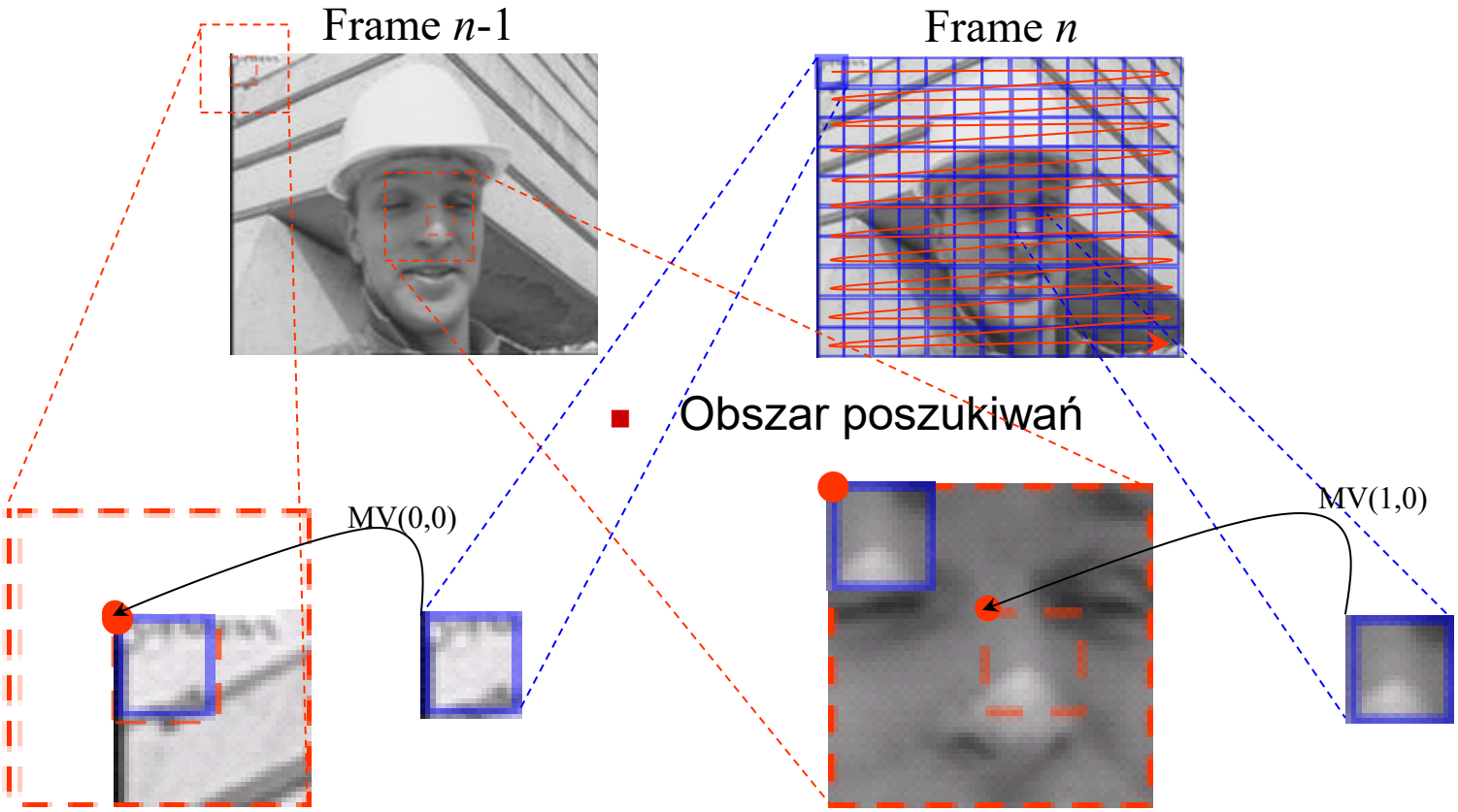




Makroblok 16x16

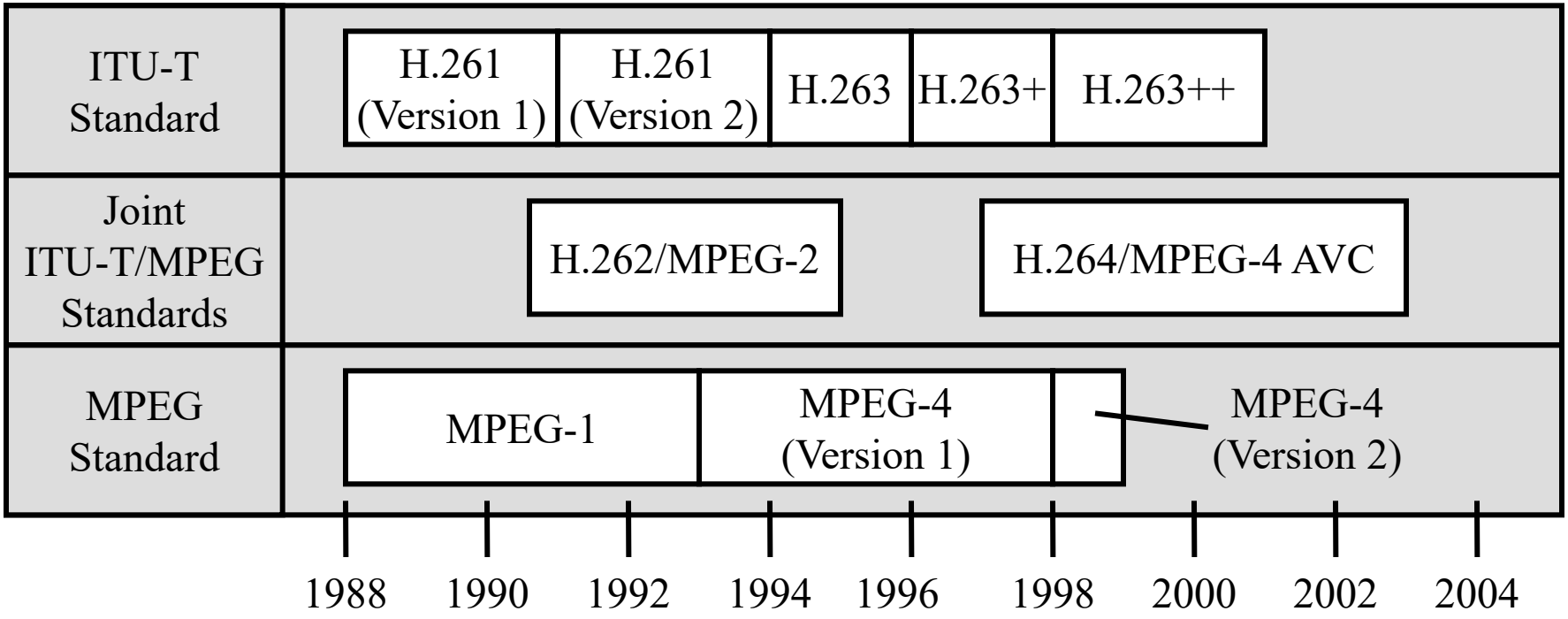
Estymacja ruchu

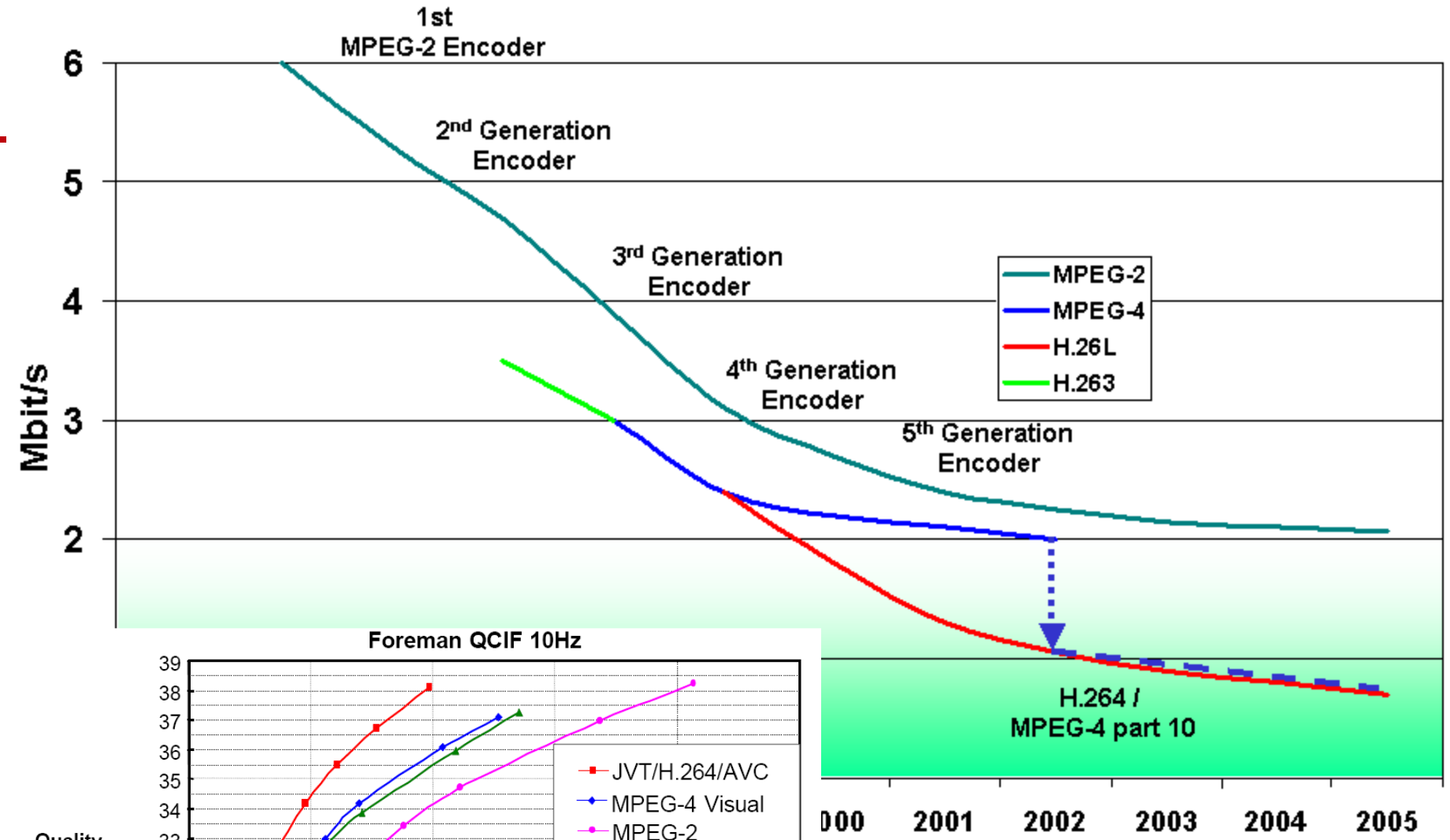
- Po wierszach

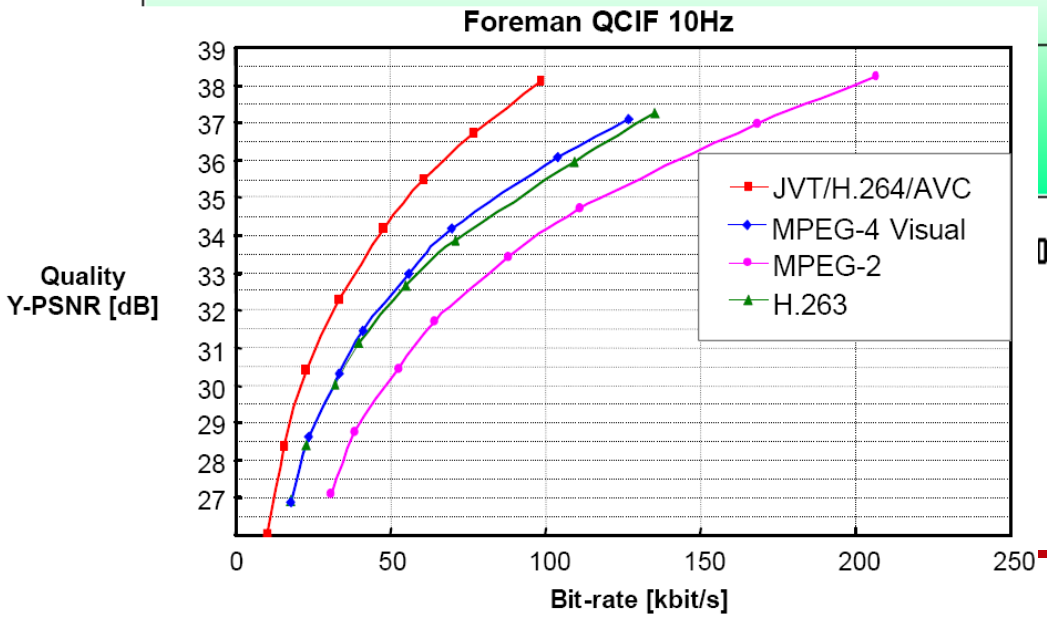


- Obszar poszukiwań

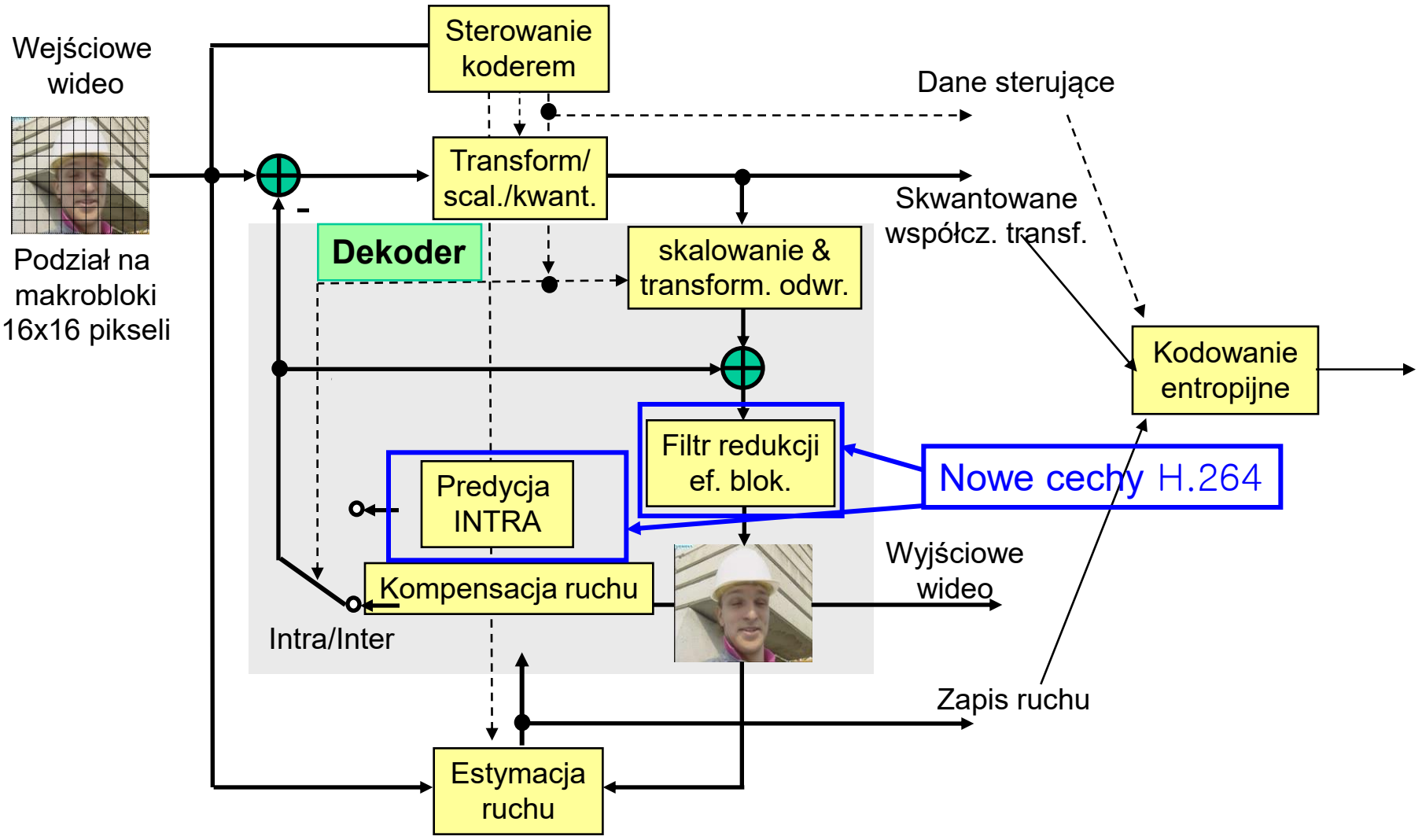
Standardy rodziny MPEG i pokrewne




1st
MPEG-2 Encoder
2nd Generation
Encoder
3rd Generation
Encoder
4th Generation
Encoder
5th Generation
Encoder
MPEG-2
MPEG-4
H.26L
H.263
Mbit/s
6
5
4
3
2
H.264 /
MPEG-4 part 10
000
2001
2002
2003
2004
2005

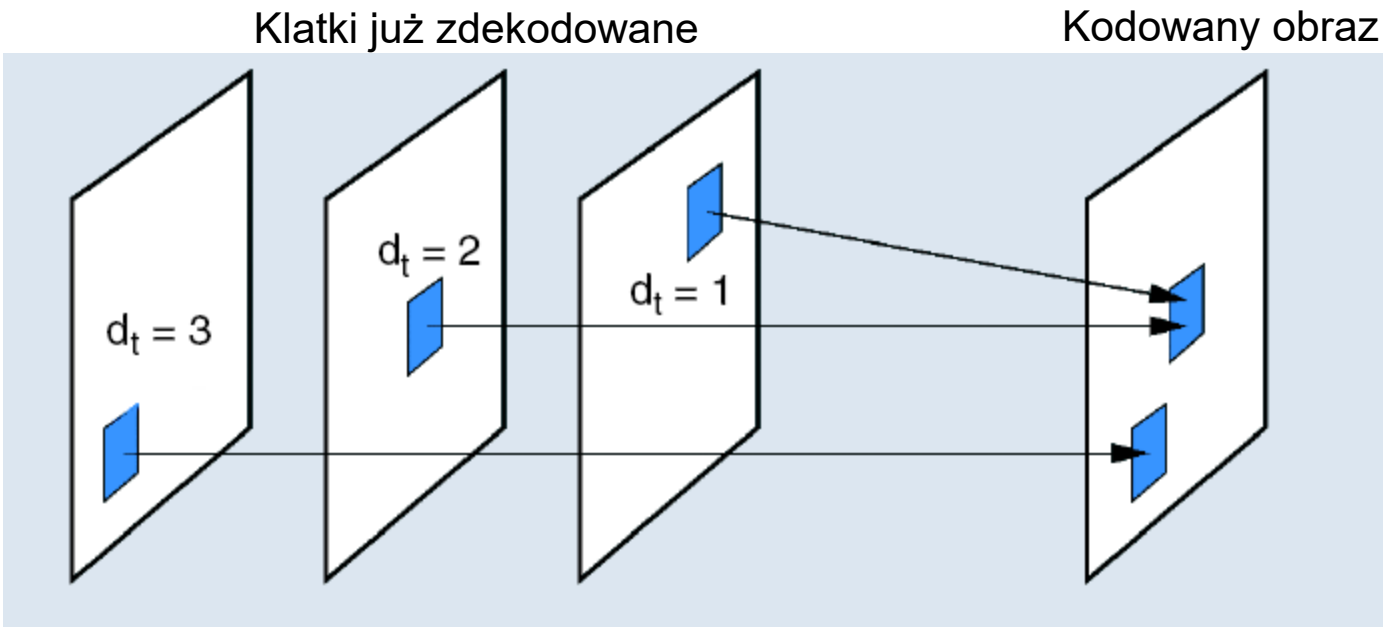

Foreman QCIF 10Hz
Quality
Y-PSNR [dB]
39
38
37
36
35
34
33
32
31
30
29
28
27
JVT/H.264/AVC
MPEG-4 Visual
MPEG-2
H.263
0
50
100
150
200
250
Bit-rate [kbit/s]

Struktura kodeka h.264



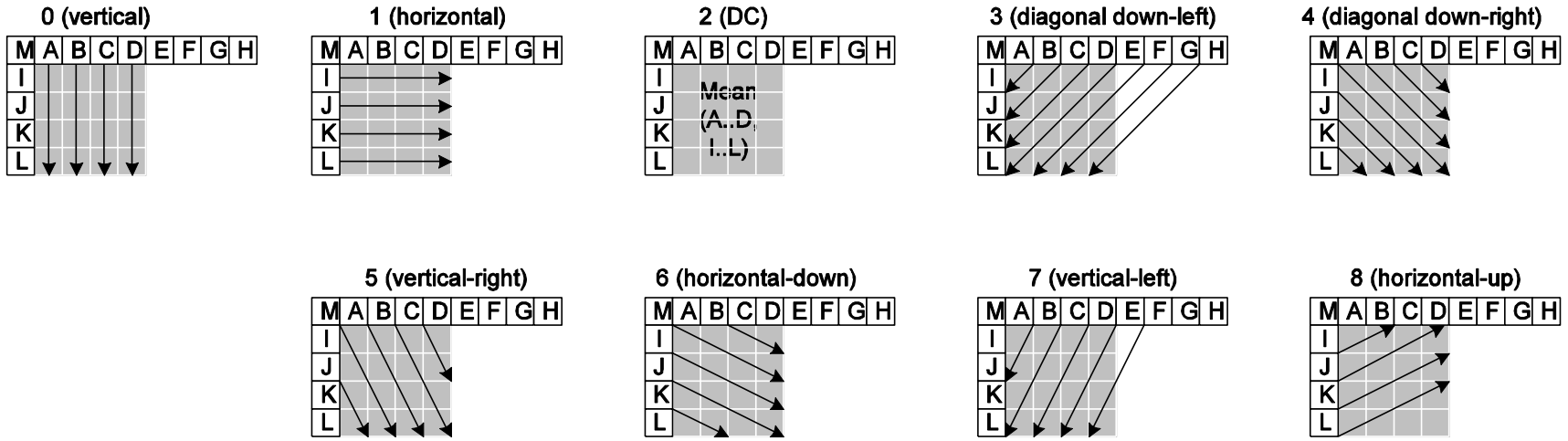
Kompensacja ruchu

- Wiele obrazów referencyjnych
  - Dowolne wagi
  - Kierunek ruchu nieistotny
  - Klatki B mogą być referencyjne

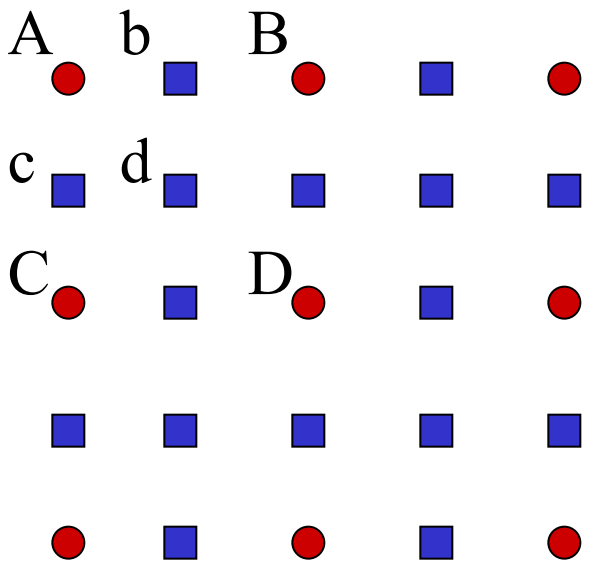


Predykcja INTRA

- 4 mody luminancji 16x16
- 9 modów luminancji 4x4
- 4 mody chrominancji 8x8
- Znacząca złożoność kodowania



Estymacja ruchu z nadpróbkowaniem

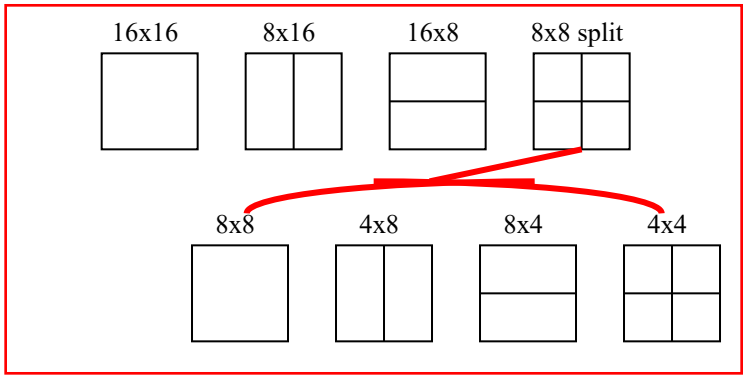


piksele

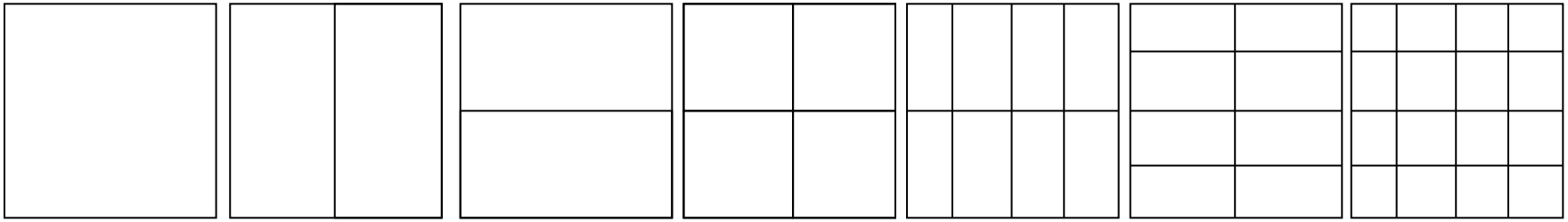
pół-piksele

b = round[(A+B)/2]
c = round[(A+C)/2]
d = round[(A+B+C+D)/4]

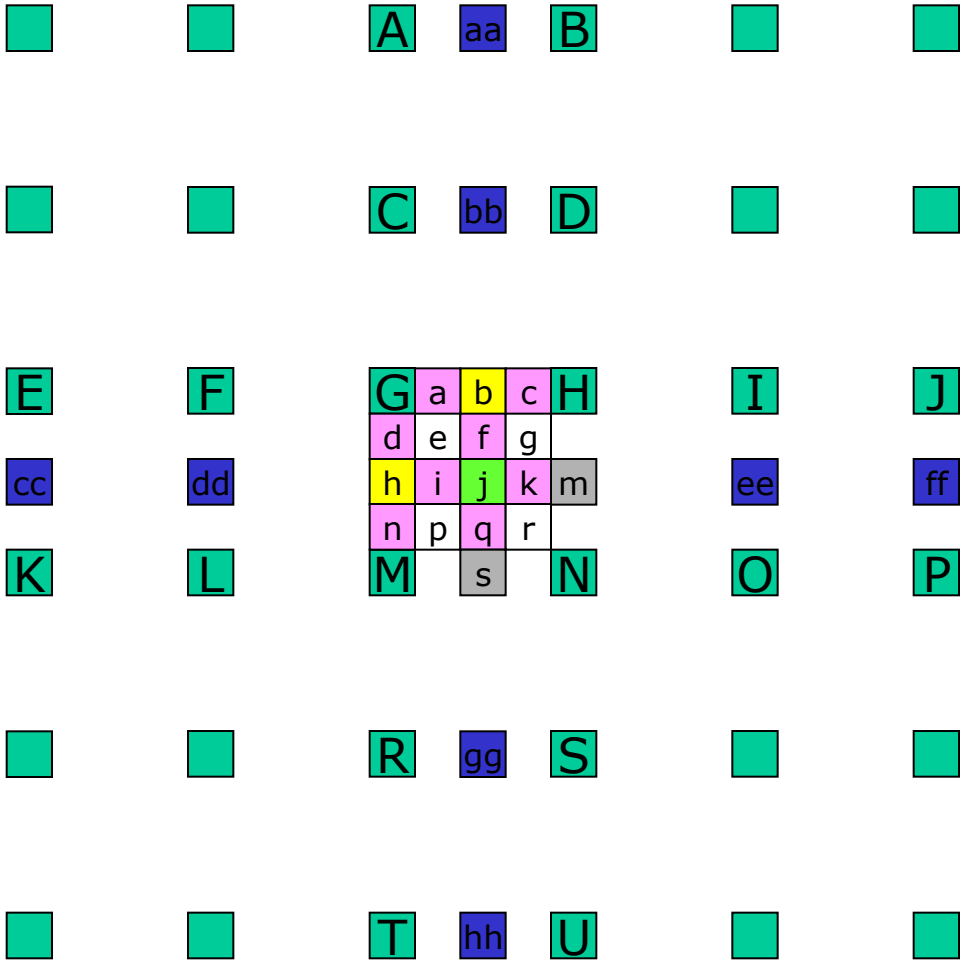
nadpróbkowanie



różna wielkość bloków



Dokładność kompensacji ruchu



$b_1=(E-5F+20G+20H-5I+J)$
$h_1=(A-5C+20G+20M-5R+T)$

$b=(b_1+16) >> 5$ clipped to
$h=(h_1+16) >> 5$ 0~255

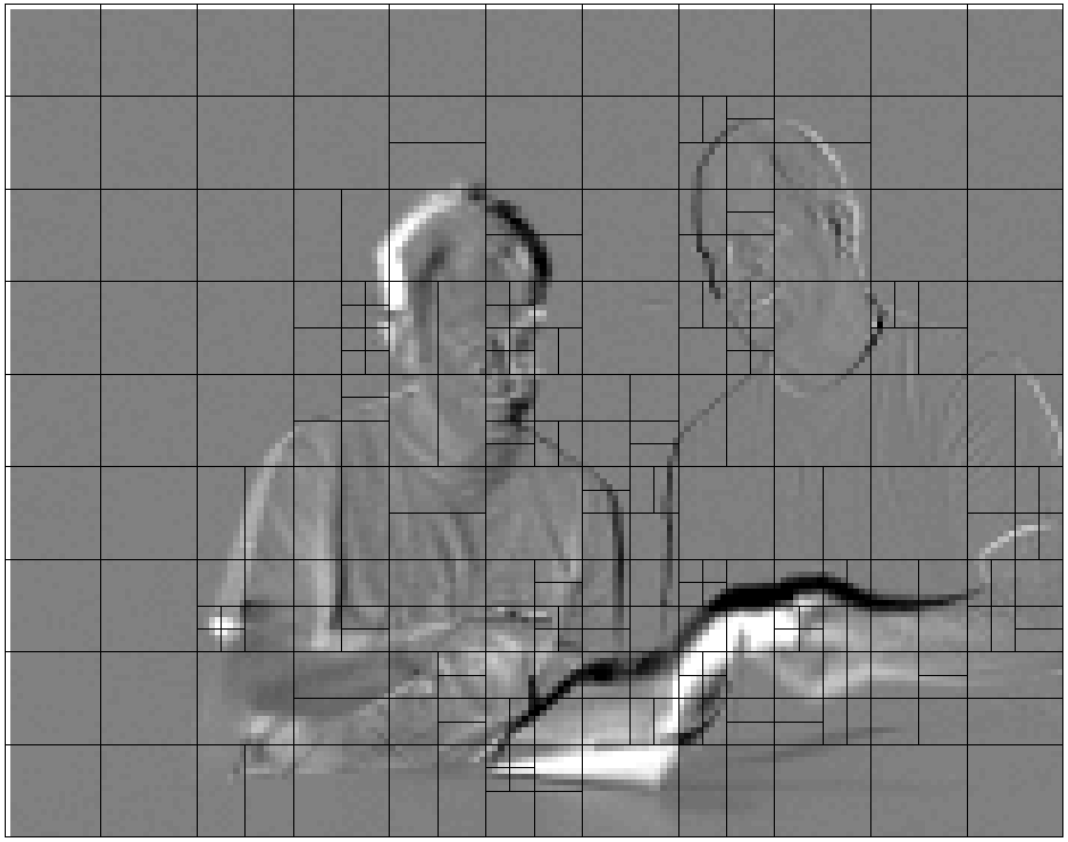
$j_1=cc-5dd+20h_1+20m_1-5ee+ff$

$j = (j_1+512) >>10$ clipped to 0~255

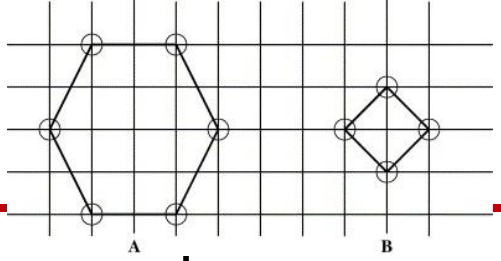
$a=(G+b+1) >>1$

$e=(b+h+1) >> 1$

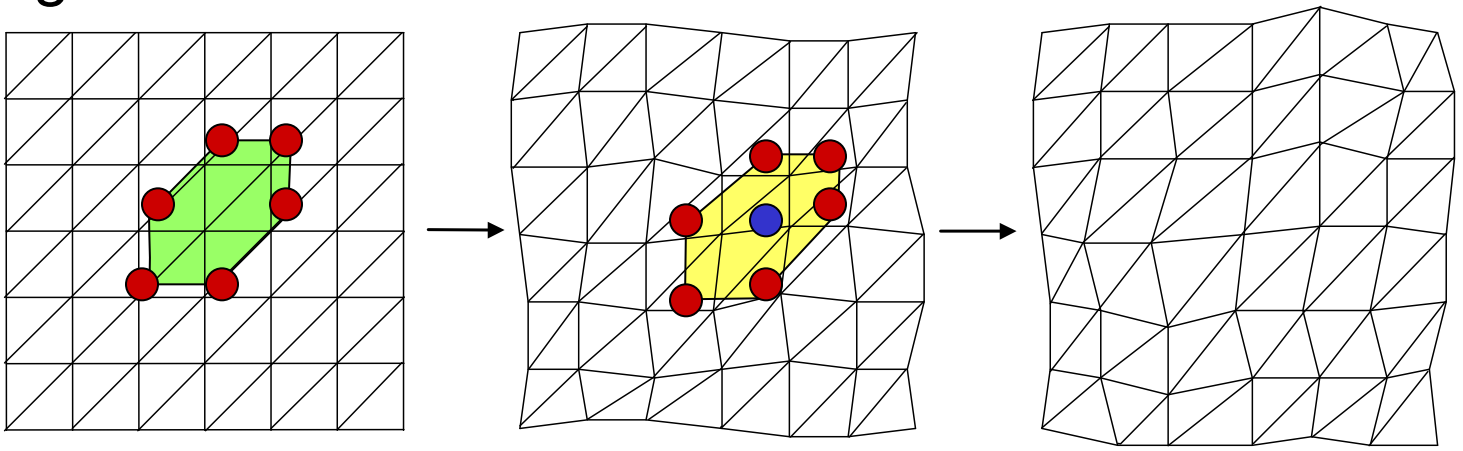
Przykład podziału na makrobloki



Estymacja ruchu

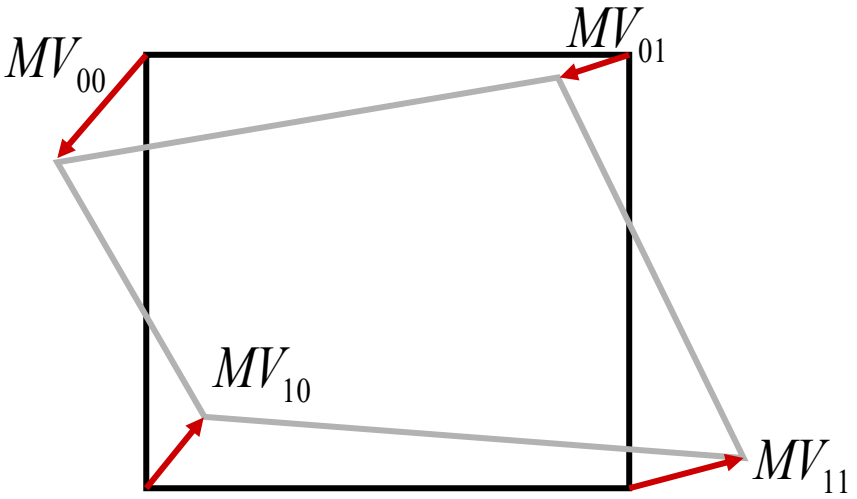


- Nieregularne siatki do modelowania obiektów w ruchu



Z. Chen, J. Xu, Y. He, , J. Zheng, **Fast integer-pel and fractional-pel motion estimation for H.264/AVC,** J Vis Com Im Rep 17(2): 264–290, 2006

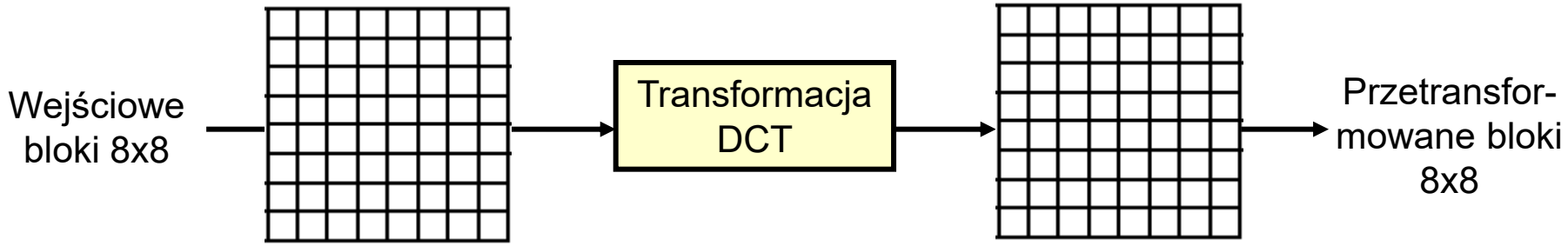
- Globalna estymacja ruchu



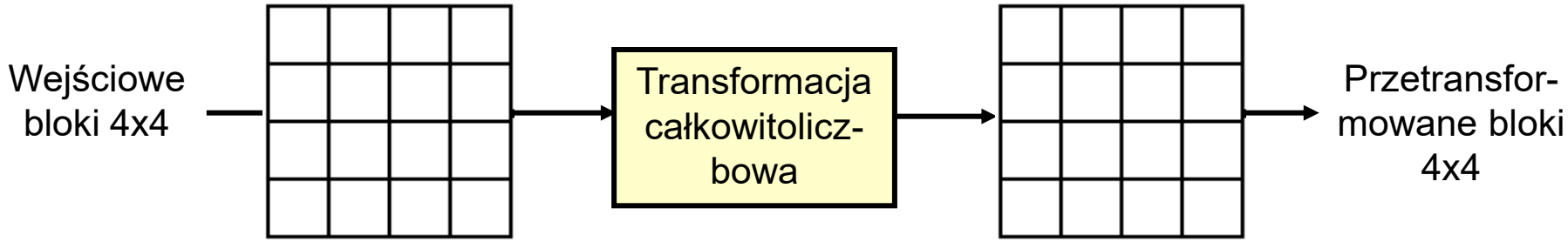
Estymacja dużych fragmentów obrazu z uwzględnieniem zmieniającej się perspektywy lub skali obiektów

Nowy rodzaj transformacji

- MPEG-2 / MPEG-4



- MPEG-4 AVC



Transformacje w AVC

$$T_{4x4} = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 2 & 1 & -1 & -2 \\ 1 & -1 & -1 & 1 \\ 1 & -2 & 2 & -1 \end{bmatrix}.$$

Rozszerzona na 8x8 dla Chroma za pomocą transformacji Hadamarda 2x2 współczynników DC
Rozszerzona na 16x16 dla Luma Intra16x16 za pomocą transformacji Hadamarda 4x4 współczynników DC

$$T_{8x8} = \begin{bmatrix} 8 & 8 & 8 & 8 & 8 & 8 & 8 & 8 \\ 12 & 10 & 6 & 3 & -3 & -6 & -10 & -12 \\ 8 & 4 & -4 & -8 & -8 & -4 & 4 & 8 \\ 10 & -3 & -12 & -6 & 6 & 12 & 3 & -10 \\ 8 & -8 & -8 & 8 & 8 & -8 & -8 & 8 \\ 6 & -12 & 3 & 10 & -10 & -3 & 12 & -6 \\ 4 & -8 & 8 & -4 & -4 & 8 & -8 & 4 \\ 3 & -6 & 10 & -12 & 12 & -10 & 6 & -3 \end{bmatrix}$$

$$H_{2x2} = \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}, \quad H_{4x4} = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 \\ 1 & -1 & 1 & -1 \end{bmatrix}.$$

Adaptacyjny filtr redukcji efektów blokowych

- Filtracja zależy od Intra/Inter, wektorów ruchu, różnic pomiędzy pikselami, QP
- Redukcja średniej bitowej o 6~9%
- Poprawa subiektywnej jakości oraz PSNR dekodowanych klatek

Bez filtru

Z filtrem