Metody kompresji i przechowywania obrazów

Obrazy – ogromnymi zbiorami danych:

Np. Fotografia 24mm x 36 mm
→ 10^7 punktów; rozdzielczość 0.01 mm
→ 256 poziomów; >10 MB

Na komputerze
→ 640 x 480 pikseli 900 kB
→ 1280x1024 pikseli 3.84 MB

Transmisja dla 9600 bps → > 20 minut

Dla obrazów animowanych → około 30obrazów/sek

Sposoby zmniejszania wielkości plików graficznych:

- **Zmniejszanie rozdzielczości**
- **Redukcja liczby kolorów**
- **Kompresja plików**
  - **usuwanie redundantnej informacji z obrazu....**
  - **niemożliwa przy braku zależności między pikselami !**

Uwagi:

- **Techniki kompresji obrazów <u>częściowo</u> wykorzystują metody kodowania i kompresji danych nieobrazowych, typu ciągów symboli**

- **Efektywność metod kompresji zależy od rodzaju obrazu (*zawsze większa dla obrazów z regularnymi kształtami, jednolitymi powierzchniami, niż dla tych z nieregularnymi, zróżnicowanymi,...)***

Problem zasadniczy:

JAKA JEST MINIMALNA ILOŚĆ DANYCH NIEZBĘDNA DO ODTWORZENIA OBRAZU?

Ilościowe określenie efektywności kompresji:

- **Współczynnik kompresji:**

  $C_R = N1/N2,$

  gdzie N1 liczba bitów dla oryginalnego obrazu
  N2 - dla skompresowanego

- **Względna redundancja danych:**

  $R_D = (N1-N2)/N1 = 1 - 1/ C_R$

Dla obrazów istotne trzy podstawowe typy redundancji:

- **REDUNDANCJA KODU**
  usuwanie → zmienna długość kodu

- **REDUNDANCJA INTER-PIKSELOWA**
  usuwanie → odpowiednie transformacje

- **REDUNDANCJA PSYCHOWIZUALNA**
  usuwanie → mapowanie ograniczające
  np. ilość kolorów

Ilustracja redundancji kodu dla obrazów :

Niech zmienna losowa r_k, w przedziale [0,1], reprezentuje poziomy kwantyzacji na obrazie i niech r_k wstępuje z p-stwem $p_r(r_k)$

Z definicji histogramu dla obrazu:

$$p_r(r_k)=n_k/n \qquad k=1,2,....L-1,$$

L jest ilością poziomów kwantyzacji
n_k oznacza liczbę poziomów k na obrazie
n jest ilością wszystkich pikseli na obrazie

Ilustracja redundancji kodu dla obrazów :

Jeśli liczba bitów, reprezentująca wartość r_k , wynosi $l(r_k)$ to średnia ilość bitów, reprezentująca każdy piksel wynosi:

$$L_{śr} = \sum_{k=0}^{L-1} l(r_k)\, p_r(r_k)$$

Całkowita liczba bitów do zakodowania obrazu MxN wynosi więc: $MNL_{śr}$

Ilustracja redundancji kodu dla obrazów :

Przykład dwóch różnych kodów **Code 1** (Lśr=3)
oraz **Code 2** (Lśr=2,7) dla obrazu o 8 poziomach szarości

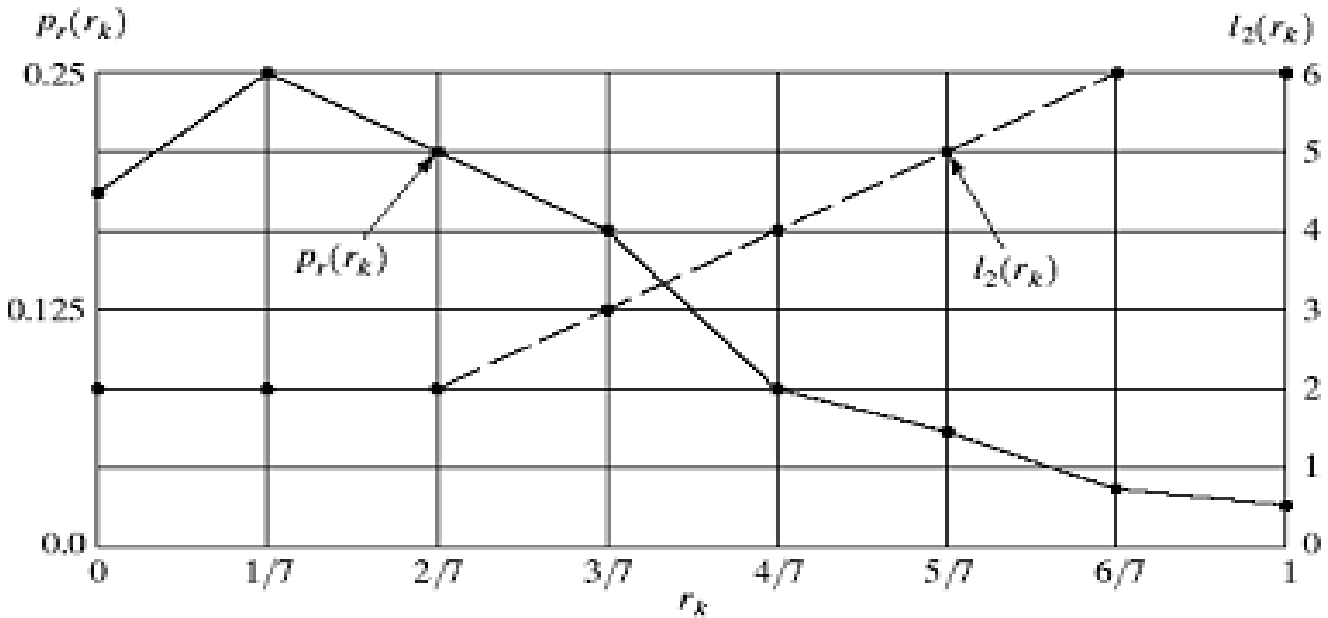
r_k	$p_r(r_k)$	Code 1	$l_1(r_k)$	Code 2	$l_2(r_k)$
$r_0 = 0$	0.19	000	3	11	2
$r_1 = 1/7$	0.25	001	3	01	2
$r_2 = 2/7$	0.21	010	3	10	2
$r_3 = 3/7$	0.16	011	3	001	3
$r_4 = 4/7$	0.08	100	3	0001	4
$r_5 = 5/7$	0.06	101	3	00001	5
$r_6 = 6/7$	0.03	110	3	000001	6
$r_7 = 1$	0.02	111	3	000000	6

Tu:
R_D=0,099

Ilustracja redundancji kodu dla obrazów :

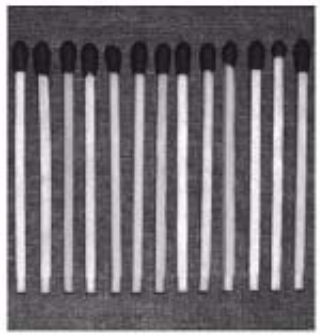
Graficzna analiza kompresji

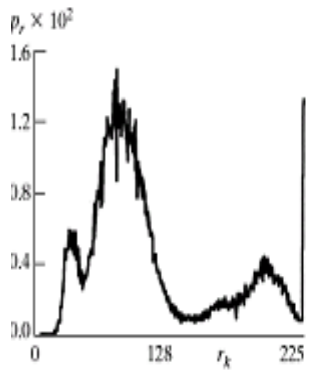
- ilustracja $p_r(r_k)$ oraz $l_2(r_k)$ versus r_k

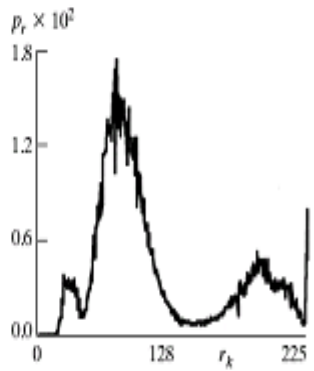


Ilustracja redundancji inter-pikselowej dla obrazów:

Przykład dwóch różnych obrazów o identycznych histogramach

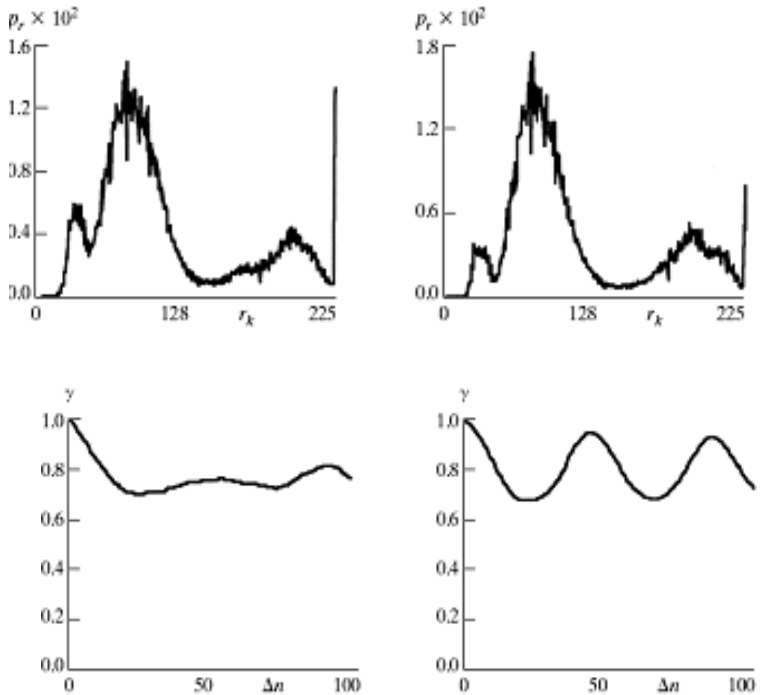






Ilustracja redundancji inter-pikselowej dla obrazów :

Histogramy oraz współczynniki znormalizowanych autokorelacji dla przykładowych obrazów wzdłuż wybranej linii



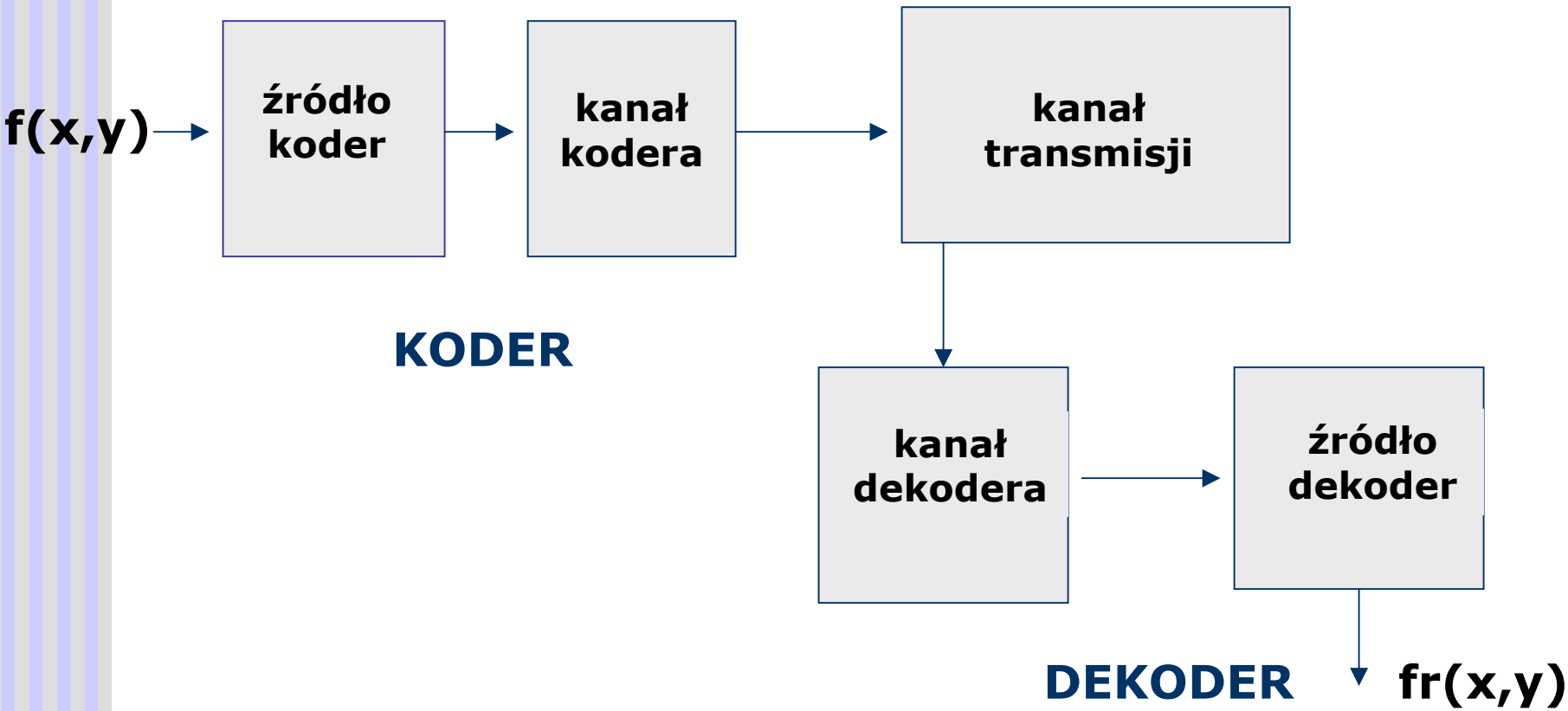
Tu:
Informacje o wartości danego piksela można przewidywać na podstawie analizy sąsiednich....

Ilustracja redundancji psycho-wizualnej dla obrazów :

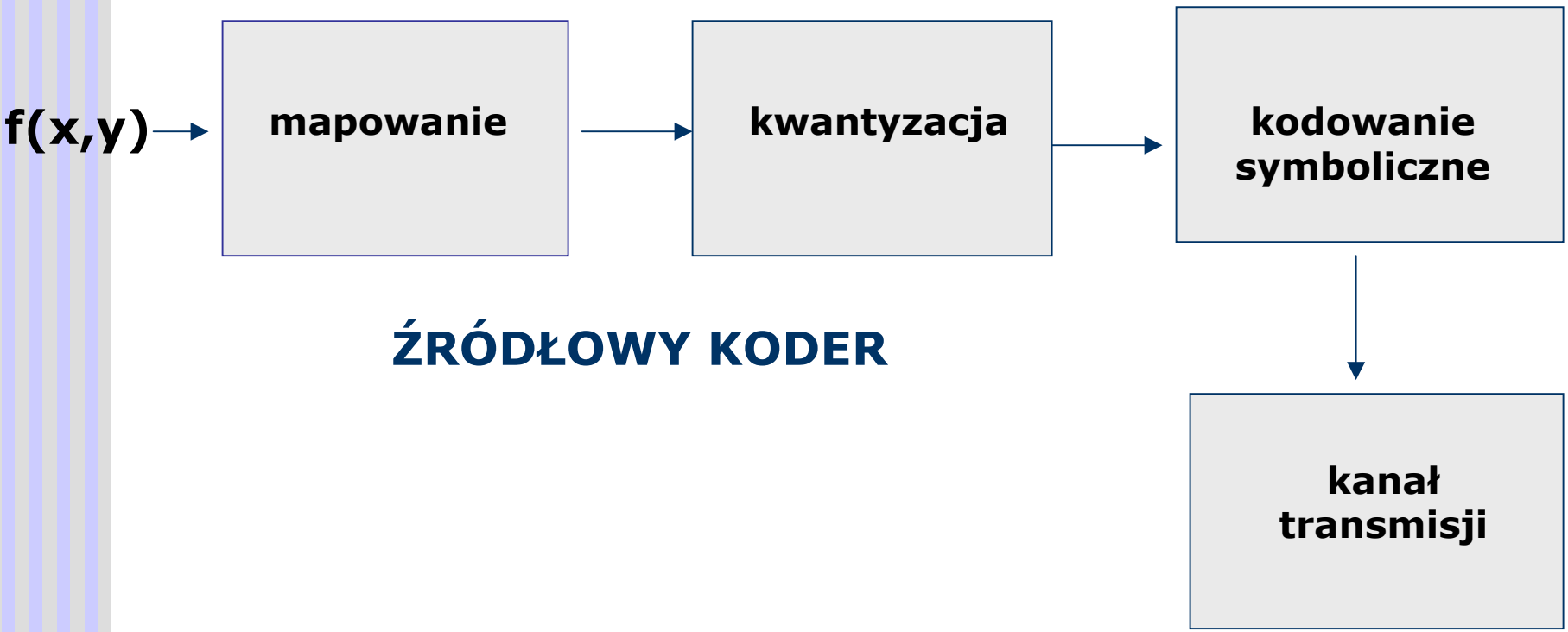
Przykładowe obrazy (a) 256 poziomów szarości, (b) 16 poziomów, (c) 16 poziomów + procedura wygładzania

(a) **(b)** **(c)**

Ogólny model kompresji:

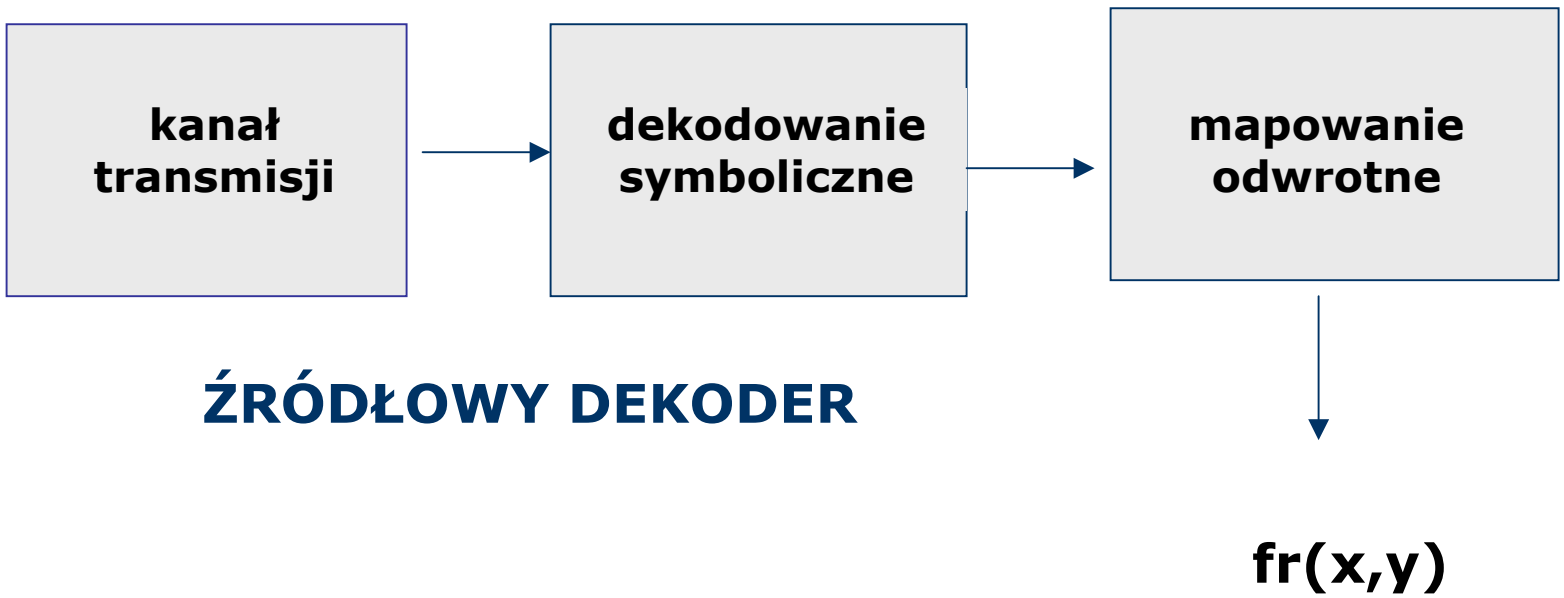


Model kodera:



ŹRÓDŁOWY KODER

Model dekodera:

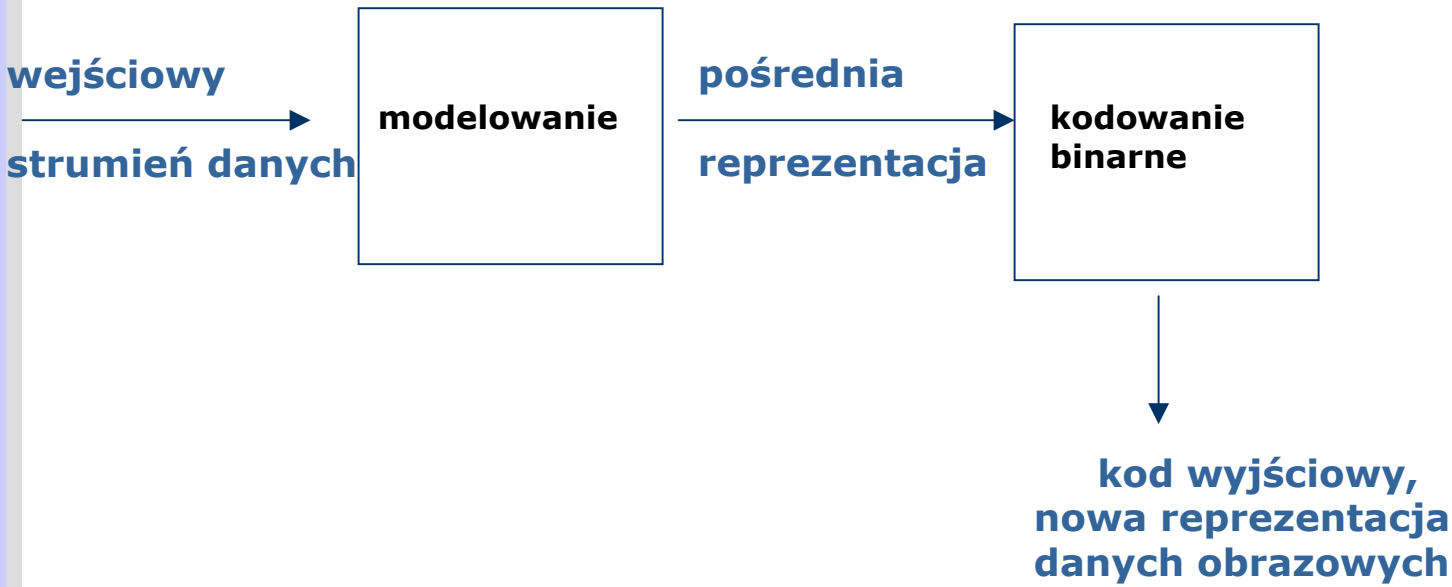


Metody kompresji:

I. BEZ STRAT INFORMACJI (*lossless*)
- bezstratne

II. ZE STRATAMI INFORMACJI (*lossy*)
- stratne, modyfikujące

Schemat bezstratnych kompresji:



Schemat stratnych kompresji:

Ad. I. Metody bezstratne:

I.1 KODOWANIE CIĄGÓW SYMBOLI

run-lengh-encoding, konturowe

I.2 METODY STATYSTYCZNE

kodowanie Huffmana

I.3 METODY SŁOWNIKOWE

LZ , LZW

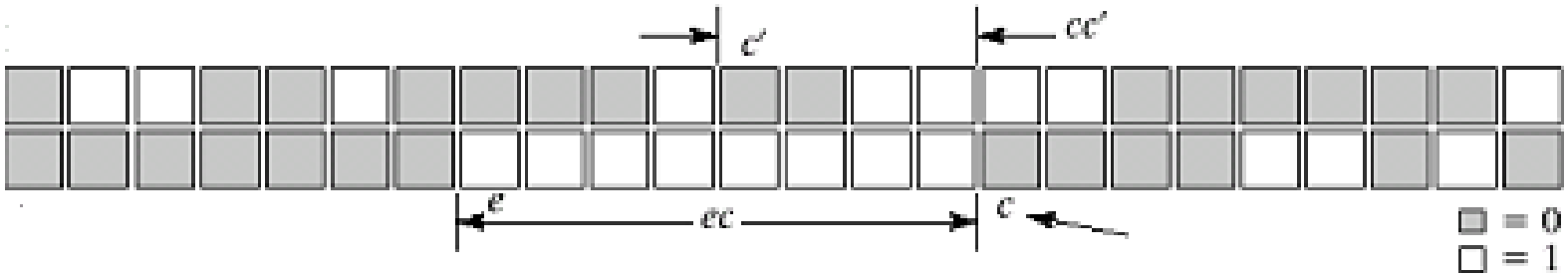
Ad. II. Metody stratne:

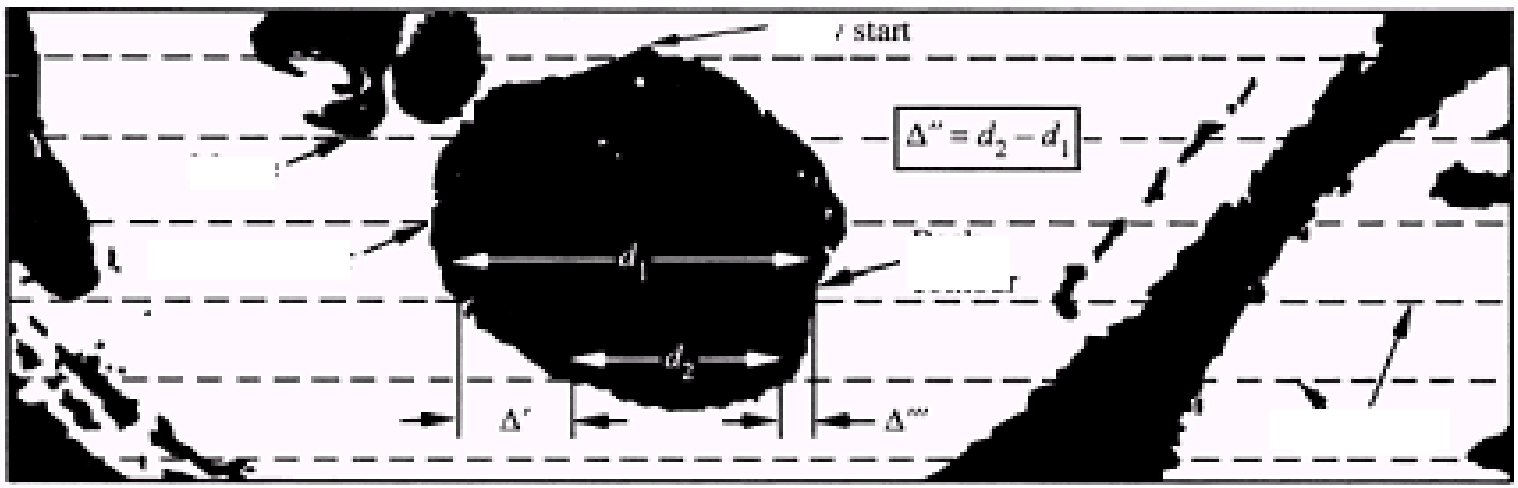
II.1 ZMIANA MODELU KOLORÓW

II.2 METODY FRAKTALNE

II.3 METODY TRANFORMACYJNE

Ad I.1 Kodowanie ciągów i konturów:





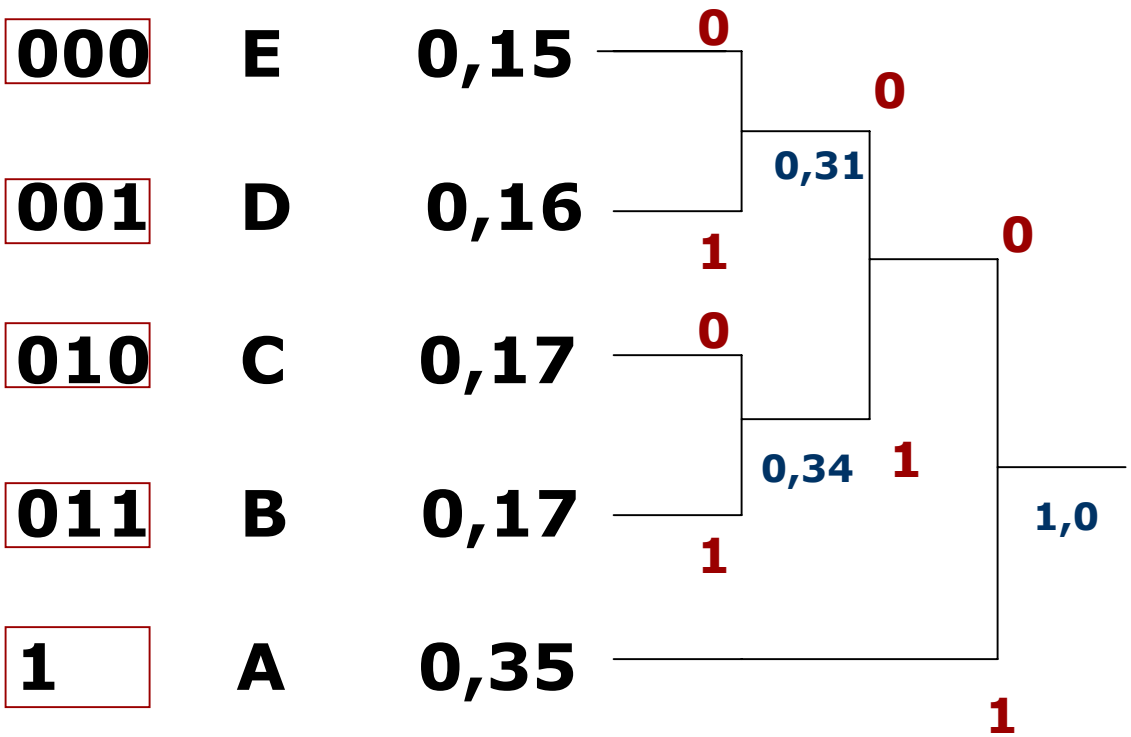
Ad I.2. Kodowanie Huffmana

- pierwsza optymalna,statystyczna metoda kompresji danych, zapewniająca redukcję średniej długości kodu dla liter alfabetu

D. A. Huffman
A Method for the Construction of Minimum Redundancy Codes
Institute of Radio Engineers Proc. 40, 1098-1101 (1952)

Ad I.2. Kodowanie Huffmana

Przykład drzewa



Ad I.2 Algorytm Huffmana:

dla każdej litery utwórz drzewo złożone tylko z korzenia i ustaw te drzewa w malejącym porządku prawdopodobieństwa użycia danej litery;

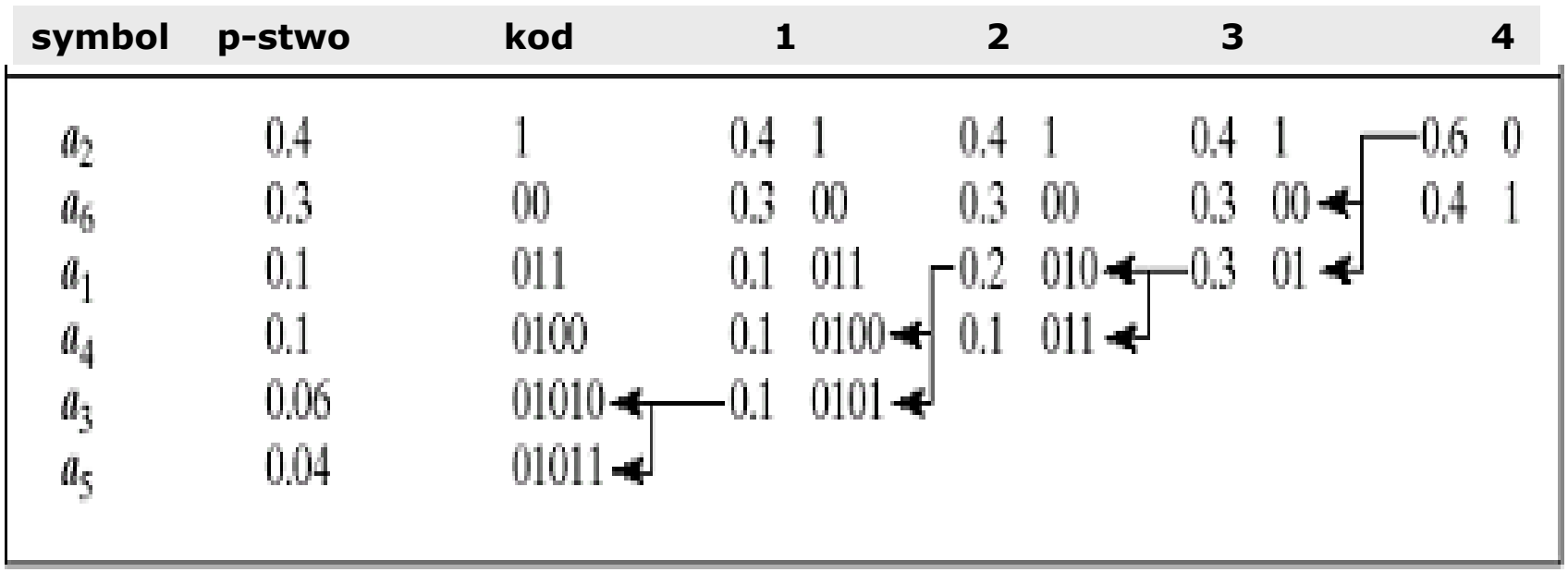
while istnieją przynajmniej dwa drzewa t1 i t2 o najmniejszych p-stwach p1 i p2 utwórz drzewo zawierające w korzeniu p-stwo p1+p2 i mające t1 i t2 jako lewe i prawe poddrzewo;

przypisz **0** każdej lewej krawędzi drzewa i **1** każdej prawej krawędzi;

utwórz słowo kodu dla każdej litery przechodząc drzewo od korzenia do gałęzi zawierającej p-stwo stowarzyszone z tą literą i łącząc napotkane zera i jedynki;

Ad I.2. Kodowanie Huffmana

Przykładowe etapy przypisywania kodów



Uwagi:

- **z techniką kodowania Huffmana związana też wcześniejsza metoda Shannona-Fano *(prawie optymalna)***

C. E. Shannon,*A Mathematical Theory of Communication,*
Bell System Technical Journal 27, 379-423,623-656 (1948)

- **tzw. kanoniczne drzewo Huffmana (w formie kompaktowej) stosowane w większości archiwów (**pkzip, lha, arj..)

Ad I.2 Algorytm Shannona-Fano:

ustaw litery alfabetu źródłowego w ciąg ***S*** uporządkowany zgodnie z prawdopodobieństwem ich użycia;

if S zawiera dwie litery dołącz **0** do słowa kodu jednej litery i **1** do słowa kodu drugiej litery;

else if S zawiera więcej niż dwie litery podziel ***S*** na dwa podciągi ***S1*** i ***S2*** tak, by różnica między sumą prawdopodobieństw liter w podciągach była najmniejsza;

dołącz **0** do słów kodu dla liter w ***S1*** i **1** do słów kodu w ***S2***;

Ad. I.3 Metody słownikowe:

- kodowanie ciągów symboli (pikseli) za pomocą odwołań do słownika zawierającego już takie ciągi

- im dłuższy ciąg uda się zastąpić indeksem do słownika tym większy stopień kompresji

- w metodach adaptacyjnych tworzony słownik zmienia się w trakcie kodowania obrazu

Abraham Lempel i Jakob Ziv and → LZ77, LZ78
+ Terry Welch →LZW84 (gif zagrożony patentem'97 ...-png)

Ad. I.3 Algorytm kompresji LZW:

wprowadź do słownika wszystkie symbole wejściowe ;

ciąg **c =** pierwsza litera wejściowa;

while kodowanie niezakończone wczytaj symbol **s**;

if c+s jest w słowniku ***c=c+s***;

else zwróć słowo kodu odpowiadające ciągowi ***c***;

włącz słowo **c+s** do słownika; **c=s**;

zwróć słowo kodu odpowiadające ciągowi **c**;

Ad. II. Metody stratne:

II.1 ZMIANA MODELU KOLORÓW

II.2 METODY FRAKTALNE

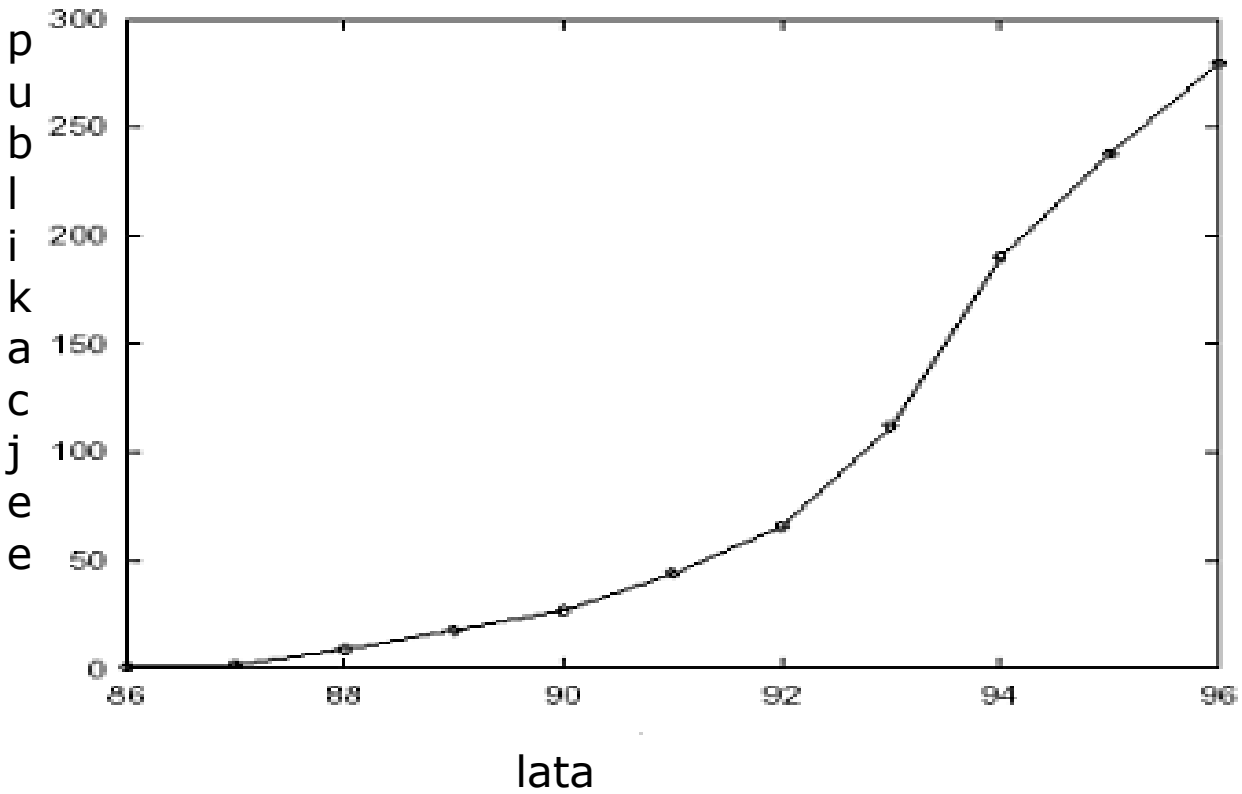
II.3 METODY TRANFORMACYJNE

Ad. II.2 Metody fraktalne:

- Benoit Mandelbrot 1975, Fractals in Nature....
- John Hutchinson 1981, Iterated Function Theory.
- Michael Barnsley 1988 , Fractals Everywhere...

(Dowód dla *Collage theorem* - jakie warunki musi spełniać IFS, aby poprawnie reprezentować obraz...→ Iterated Functions Company 1991)

Ad. II.2 Metody fraktalne:



Liczba publikowanych prac nt. kompresji fraktalnych

Uwagi:

- ✓**Fraktalami w metodach fraktalnej kompresji są układy IFS** *(Iterated Function Systems)*

- ✓**Kompresja (stratna!) jest bardzo wolna, dekompresja natomiast bardzo szybka**

- ✓**Szczegóły technologii kompresji patentowane** *(owiane tajemnicą...)*

Kompresja fraktalna

Nadzieje związane z kompresją fraktalną związane głównie z następującymi obserwacjami:

1. Wiele naturalnych obiektów przyrody ma cechy samopodobieństwa...
   (Benoit Mandelbrot 1975 *self similarity, selfaffinity*)

2. Można znaleźć układy IFS bardzo dobrze przybliżające obrazy świata przyrody (liść paproci u Barnsley'a – przybliżany przy pomocy zaledwie 4 transformacji...)

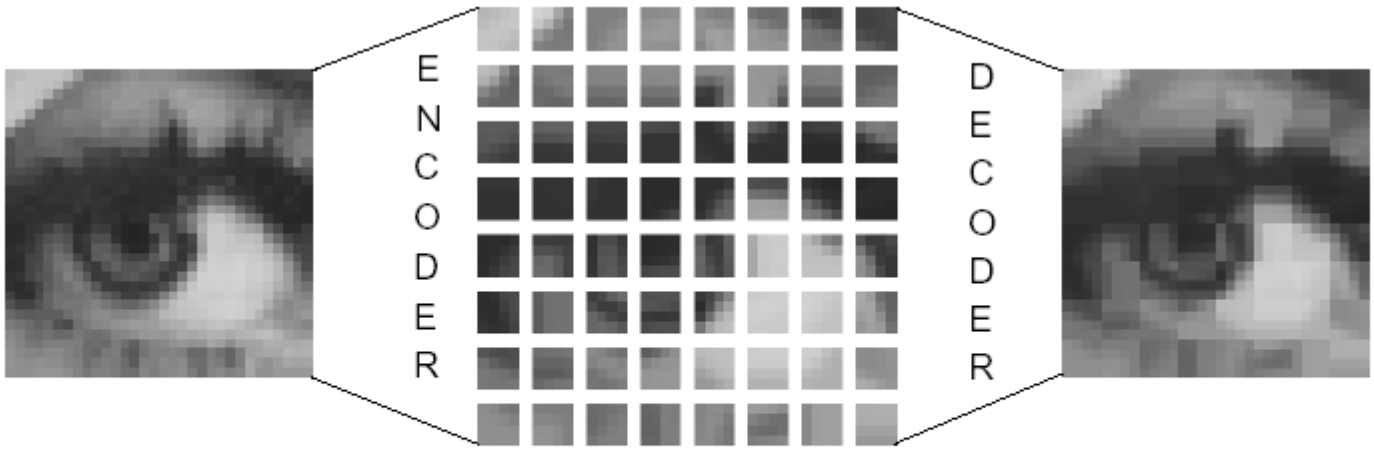
Kompresja fraktalna

Szukanie odpowiednich, dobrych IFS dla danego obrazu wymaga następujących procedur:

1. Podział obrazu na odpowiednie bloki/regiony

2. Wybór odpowiednich transformacji /przekształceń afinicznych

Kompresja fraktalna



Kompresja fraktalna

Próby automatyzacji pewnych procedur

-bez powodzenia!

Wg Michaela Barnsley'a (1988)

"Complex color images require about 100 hours each to encode and 30 minutes to decode on the Masscomp [dual processor workstation]. That's 100 hours with a _person_ guiding the process..... "

Ale programy kompresji fraktalnych z Windows DLL można kupić .....(!!!)

Kompresja fraktalna

Próby rozwiązania problemu odwrotnego:

tzn. dla danego obrazu

znaleźć optymalne transformacje

(tzw. transformacje fraktalne)

podejmowane w wielu laboratoriach

- od stosowania algorytmów genetycznych

po ...algorytm GSA...(!)

...problem pozostaje nierozwiązany!

CO DALEJ??....

Na dziś cała nadzieja(!)
w transformacjach falkowych
dla kompresji obrazów...

(np. JPEG 2000)

Generalnie, wciąż nierozwiązany problem jednolitych standardów dla plików graficznych...

Przykładowe formaty obrazów:

- **bezstratne**

  bmp → *bitmap*
  tiff → *tagged interchange file format* 1992
  png →*portable network group* 1997
  pnp →*portable network phtographs 2002*

- **stratne**

  gif → *graphics interchange format* 1987
  jpeg →*joint photographic expert group 1995*