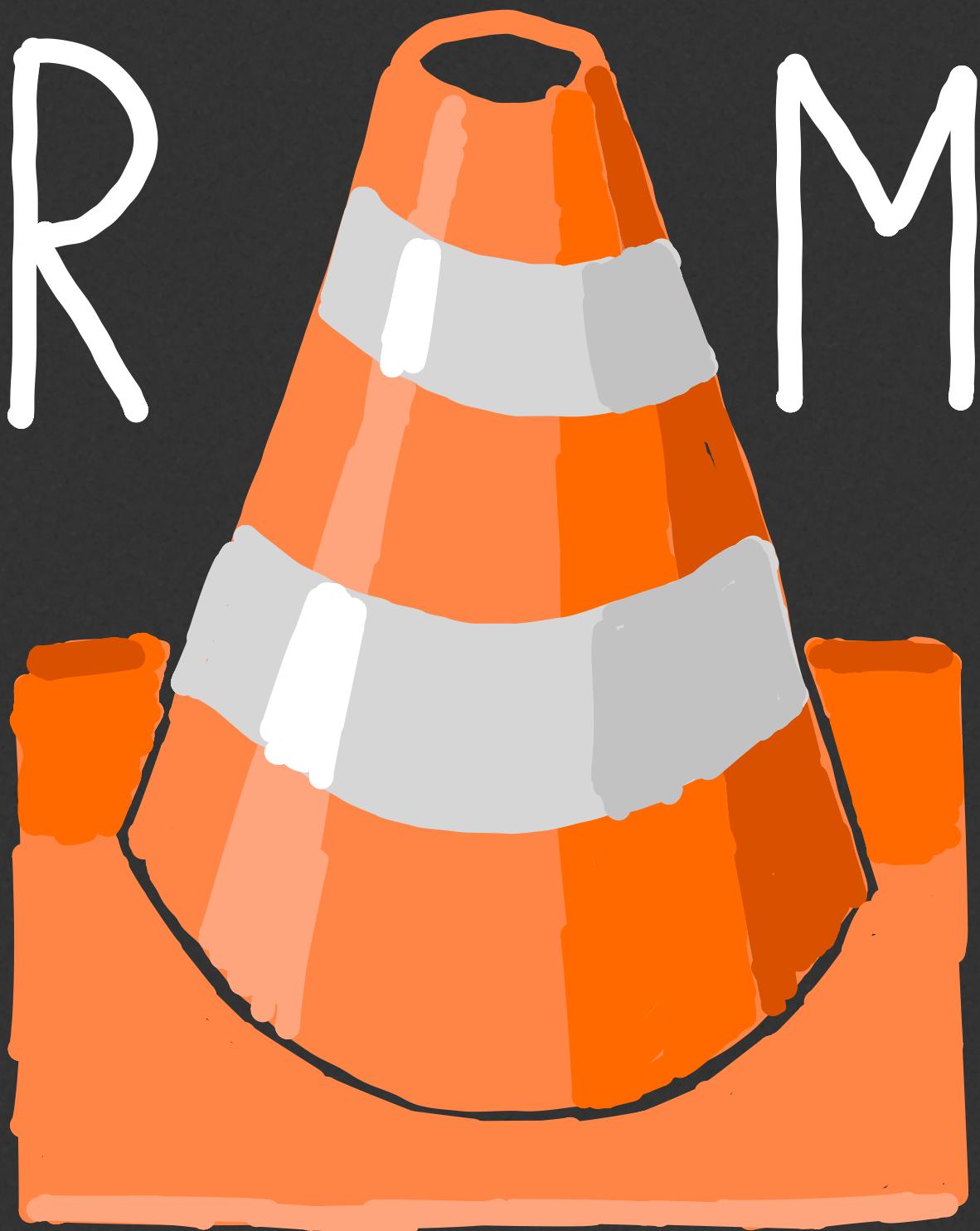


All rights unreserved



rm -rf /bin/life

END USER LICENSE AGREEMENT

by reading the following agreement you automatically  
agree that: You owe the author of this work a blow job  
2020 ☺

Copyright© ME BITCH

## Rasterske slike

- Matrika pikselov - Raster
- število pikselov v rasterju: ločljivost
- dovolj visoka ločljivost - kadar ne vidimo piksel
- piksel ima določeno barvo
- Vzorčenje - nam pove št. pikselov (visina in širina)
- Kvantizacija - pove št. bitov na vzorec oz. piksel oz. barvo
- Barva piksla je razdeljena na več vrednosti: vsaka vrednost je v svojem kanalu
- Barvni model pove prestihavo vrednosti v končno barvo
- RGB, sRGB, CYMK, YCbCr, HSV

## Shranjevanje barv

- Direkten način - predstava z n pikseli:
  - vsak piksel je  $3 \times 8$  bitov  $\rightarrow$  R G B
- Indeksni način
  - imamo slovar katerim zapisujemo vse unikatne barve slike  $\rightarrow$  m bitna predstavitev barve
  - v matriki se zapise referenca na barvo v slovarju  $\rightarrow$  n bitov na piksel
  - koristno kadar je  $n < m$
  - lahko predstavimo  $2^n$  odtenkov iz nabora  $2^m$  barv

## Vrste rasterskih slik

- Enobarvne - monohromatske
- Sivinske - se lahko zapisuje kot vektor bitnih ravnin, bitna globina n - odtenkov  $2^n$
- Barvne
  - Zvezne (slike z zveznimi barvnimi toni): Fotografije
  - Diskretné slike: skice, risanke

## Stiskanje rasterskih slik

- Želimo predstaviti z čim manj bajti:
- Novejši pristopi odstranijo čim več redundanci likrat:
- Vrste postopkov:
  - brezizgubne (lossless)
  - izgubne (lossy)
  - skoraj brezizgubne (near lossless)

## 1.RLE (run length encoding)

- prebiramo piksele in zapisemo št ponovitev vsake barve
- dolžine ponovitev dodatno stisnemo s variable length code (VLC)
- primerno za monohromatske slike  
 $AAA BBBB BAA CCCCCC \Rightarrow 3A5B2A6C$
- Kontekstu: RLE - koda za kodiranje določimo glede na sosednje piksele

## 2. Stiskanje sivinskih slik z n ravnicami

- Stiskanje n-bitnih ravnin na podlagi predhodne metode n-kvantizacija piksla
  - predpostavimo, da imajo sosednji px podobne vrednosti:
  - problem, če je npr. 63 in 64 (1000000 in 011111), rešitev Grayeve kode

### Grayeve kode

- Problem, podobne vrednosti, vendar zelo različna bitna predstavitev  
 $7 \Rightarrow 0111$     $8 \Rightarrow 1000$

- Grayeve zrcalne kode  $n \Rightarrow$  st. bitov za število

$n=1$	$n=2$	$n=3$
$0 \rightarrow 0$	00	0000
$1 \rightarrow 1$	01	1001
	11	1011
	0	10 $\Rightarrow$ 3010 ...
		$\uparrow$ zrcaljeno 4110
		5111
		6101
		7100

## 3. Napovedi

- Na podlagi konteksta se napove piksel P  
npr. A = avg. sosednjih pikslov piksla P
- Izračunamo napako  $\Delta = P - A$
- Zakodiramo A z VLC
- Pogosto je  $\Delta$  za  $P$  blizu laplacove porazdelitve
- Tonji so pričakovane napake manjše, kar pomeni, da lahko zapisemo z manjšim številom bitov (npr. Golomb-Rice)

### Golomb-Rice

- Uncarne kode:

0-1  
1-01  
2-001  
3-0001  
⋮

- Koda Rice

- MSB je predznak napake  $\Delta$
- iz  $\Delta$  izločimo n LSB bitov, postanijo LSB kode
- preostalo kodiramo z uncarne dode

Primer:

$$\begin{array}{l} \Delta = -46 \quad n=4 \\ \text{sign}=1 \quad 101110 \\ \text{sign} \\ 1|0011110 \end{array}$$

- Kontekstno odvisna napoved

- glede na okolico izberemo različno funkcijo  
za napoved - horizontalno prelivanje druge funkcije ko vertikalno



#### 4. Transformacije

- Kodiramo transformirane vrednosti.
- Diskretna kosinusna transformacija, Fourierjeva transformacija, diskretna valčna transformacija.

#### 5. Ločeno kodiranje kanalov barv

#### 6. Slovarji:

- Posamezni deli slike se lahko ponavljajo npr. črke  
ideja je, da shranimo ta del slike v slovar in  
v slike hranimo samo referenco ta del.

#### 7. Fraktalno stiskanje

- Nadgradnja stiskanja s slovarjem
  - delimo sliko na dele
  - kodiranje transformacije enega dela slike  
v drugi del slike

#### 8. Podvzorčenje

- Brisemo vsato 2. vrstico in stolpec
- Nadgradna - povprečenje

#### 9. Kvantizacija

- Zmanjšamo št. bitov za zapis pikselov
- Izboljšava - vektorska kvantizacija
  - v kodirno knjigo upišemo najbolj pogoste piksele (npr. 4 zaporedne)
  - zakodiramo sliko kot zaporedje indeksov

#### Merjenje podobnosti med slikami

za merjenje podobnosti uporabimo  $\text{RMSE}$  (root mean square error)  $\text{PSNR}$  - signal to noise ratio  $\text{MSE} = \frac{1}{n} \sum_{i=1}^n (X_i - Y_i)^2$

$$\text{PSNR} = 20 \log_{10} \frac{\max_i(X_i)}{\text{RMSE}}$$

$$\text{RMSE} = \sqrt{\text{MSE}}$$

#### SSIM - (structural similarity index metric)

- PSNR ni za vidno zaznavo prilagojen
- SSIM - ocena vizualne podobnosti, tako ko človek vidi.
  - $x, y$  - isti ohni znotraj  $X$  in  $Y$
  - vrednost med -1 in 1 (1 je identično)
- Deluje na barvnem prostoru YCbCr

$$\text{SNR} = 20 \log_{10} \frac{\sqrt{\sum_{i=1}^n P_i}}{\text{RMSE}}$$

- Po sliki se premika z očeh in računa povprečne vrednosti
  - počevadi  $8 \times 8$  ali  $11 \times 11$

$$\text{varianca } \sigma_x^2 = \frac{1}{n} \sum_{i=1}^n (x_i - \bar{x})^2 \quad \text{povprečna vrednost } \bar{x}$$

$\mu_x, \mu_y$  - avg okenj

$$\sigma_x, \sigma_y - \text{varianca vrednost v oknu: } \sigma_x^2 = \frac{1}{n-1} \sum_{i=0}^{n-1} (x_i - \mu_x)^2$$

$$G_{xy} - ko varianca x in y = \frac{1}{n-1} \sum_{i=0}^{n-1} (x_i - \mu_x)(y_i - \mu_y)$$

$$c_1 = (0,01 \cdot (2^{bPP-1}))^2$$

$$c_2 = (0,03 \cdot (2^{bPP-1}))^2$$

$$\text{SSIM}(x, y) = \frac{(2\mu_x\mu_y + c_1)(2G_{xy} + c_2)}{(\mu_x^2 + \mu_y^2 + c_1)(G_x^2 + G_y^2 + c_2)}$$

## Osnovne transformacije

### Nad pikselov

- Zaporedne pare pikselov smatramo kot točke na ravnini.
  - Poiščemo premico, ki gre skozi točko.
  - Točke zarotirajo na x osi, npr za  $45^\circ$  z rotacijsko matriko
- $$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} \cos(\theta) & -\sin(\theta) \\ \sin(\theta) & \cos(\theta) \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} \quad R^{-1} = R^T$$
- $\overset{\pi}{R}$

Izračun križne korelacije:

$$CCR = \frac{1}{n} \sum_{i=1}^n x_i y_i$$

### Kodiranje točk:

- najprej x nato y ker je y običajno niče ga lahko kodiramo z manj biti.
- pri manjših vrednostih y sploh ne zapisujemo y ga zaokrožimo navzdol.

- možno tudi v 3D uporabimo 3 zaporedne piksele kot 3D točko
- uporabimo 2  $3 \times 3$  rotacijski matriki, da transformiramo na eno os 2 koordinati bosta imeli majhne vrednosti.

## Ortogonalne transformacije

- manjše št. bitov za zapis
- večino energije v zgolj nekaj koeficientih

### Linearne transformacije

$$C = WD$$

$c_i$  - transformirane vrednosti.

$w_{ij}$  - teži baznih vektorjev

$d_i$  - zaporedne vrednosti pikselov

Ker je matrika ortogonalna in ortonormalna je  $W^T = W$

- W določimo tako, da imamo v prvem c: Čim več energije
- pri c: določa očitno frekvenca ostali višje harmonike  $W = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 \\ 1 & -1 & 1 & -1 \end{bmatrix}$  Vrstice naj bodo parne ortogonalne
- prije vrstica naj omogoča čim večji:  $c_1 \rightarrow w_{11} > 0$
- ostali ci naj bodo manjši polovicu  $w_{11} > 0$ , polovicu  $w_{11} < 0$
- Ker je energija c zelo visoka ga normaliziramo z  $\frac{1}{2} \rightarrow W = \frac{1}{2} \cdot W$
- količinški delež energije prinese d:  $h \cdot \hat{d} = (\hat{d} \cdot \hat{d} - (d_1^2 + d_2^2 + \dots + d_n^2 - d_c^2)) / (\hat{d} \cdot \hat{d})$
- inverz od W  $\Rightarrow W^{-1} = W^T$

### Primer za dvodimenzionalno transformacijo

- imamo 2D matriko ki je enake dimenziji kot W
- $C' = W \cdot D$

$$C = C' W^T = W D W^T = W D W \leftarrow \text{Transformacija}$$

$$D = W^T C W = W C W \leftarrow \text{Inverzna transformacija}$$

$$\hat{d} \cdot \hat{d} = d_1^2 + d_2^2 + \dots + d_n^2$$

## Walsh-Hadamardova transformacija (WHT)

- Rekurenčno definirano
- $H_m$  je ortogonalna in orthonormalna  $H_m = H_m^{-1}$

$$H_m = \frac{1}{\sqrt{2}} \begin{bmatrix} H_{m-1} & H_{m-1} \\ H_{m-1} & -H_{m-1} \end{bmatrix}$$

### Diskretni WHT

- Optimizacija skalarnih faktor kvadriramo oz. šitamo bite
- Vrstice in stolpci so ortogonalni samo  $H_m^{-1}$  ni več trivialen

$$H_1 = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}$$

$$H_2 = \frac{1}{2} \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & -1 & 1 & -1 \\ 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 \end{bmatrix}$$

⋮

$$H_m = \frac{1}{2} \begin{bmatrix} H_{m-1} & H_{m-1} \\ H_{m-1} & -H_{m-1} \end{bmatrix} \quad |H_0| = 1$$

## Kosinusna transformacija in disfuretka (DCT)

- Temelj izgubnega stiskanja
- DC koeficient vsebuje večino energije ( $u=0$ )
- AC koeficient prvih par se vsebuje nekaj energije ostali bolj malo
- V praksi primerni  $n=8$
- Temelj za izgubno stiskanje
- možno predstaviti z matriko za linearno transformacijo za  $n=8 \Rightarrow W 8 \times 8$  matrika

$$C(u) = \begin{cases} \frac{1}{\sqrt{2}} : u=0 \\ 1 : u>0 \end{cases}$$

$$F(u) = \sqrt{\frac{2}{n}} C(u) \sum_{x=0}^{n-1} f(x) \cdot \cos\left(\frac{\pi(2x+1)u}{2n}\right) \leftarrow \text{Trans.}$$

$$f(x) = \sqrt{\frac{2}{n}} \sum_{u=0}^{n-1} C(u) F(u) \cos\left(\frac{\pi(2x+1)u}{2n}\right) \leftarrow \text{inv.trans.}$$

## 2D DCT

- Uporabljeno na slikah
- Pogosto  $n=m=8$ , DCT izvedemo nad vrsticami in nato stolpcem v praksi dodamo novo koordinato podvojimo izraz  $C$  in  $\cos(...)$
- Možno predstaviti tudi z  $8 \times 8$  matriko  $8 \times 8$  W matriki

## JPEG

- najbolj popularen datotečni format za brezizgubno stiskanje
- JIFF - datotečni format (specs), JPEG: postopek stiskanja
- možno določati stopnjo stiskanja, izgubno in brezizgubno (alčen postopek)
- Podprtji načini stiskanja JPEG:
  - Zaporedno stiskanje
  - Naprednjoče stiskanje: vsak naslednji prehod vsebuje več informacij, primerno za prenos po počasnih omrežjih
  - Lobsless
  - Hierarhično

## Zaporedno stiskanje

- osnova ideja lossy stiskanja: močneje stisnemo manjše podrobnosti, podrobnosti določimo s frekvenčno analizo (DCT)

### 1. Korak

- Pretvorimo iz RGB v YCbCr (razen z. grayscale)
- Y-svetlost, Cb Cr - kromatičnost, v nadaljevanju stiskamo vsak kanal posebej

### 2. Korak

- Podvzorcejmo Cb in Cr kanale
- Barve lahko bolj stisnemo, ker je človeško oko bolj občutljivo na svetlost
- Najpogostejsi zapis podvzorečenja  $J: a:b$
- $J$  - št. komponent svetlost (ponavadi  $J=4$ )
- a - št. komponent v h-smer, b - št. spremenih v barvnih komponentih v drugo vrstico

- podprtosti: 4:4:4, 4:4:0, 4:2:2, 4:2:0, 4:1:1

1	2	3	4
1	2	3	4
1	2	3	4
1	2	3	4

1	2
1	2

1	2
1	2

### 3. Korak

- Slike razdelimo na bloke, počevši od velikih 8x8 pikselov. Stiskamo po blokih neodvisno od drugega.
- če ni deljivo z 8 lahko uporabimo različne strategije razteganja slike

### 4. korak

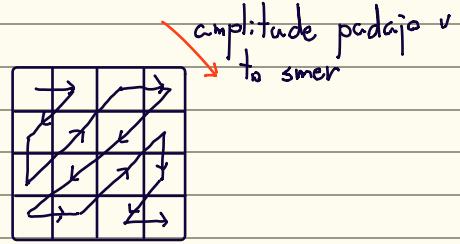
- transformiramo bloke z DCT, levo zgoraj DC komponenta, ostalo AC
- imamo malo izgubo zaradi zaščitve na celo št.
- predhodno odštejemo 128 od vrednosti;

### 5. korak

- višje frekvence predstavimo z manjšo natančnostjo, uporabimo kvantizacijske koeficiente
- s stopnjo kvantizacije dolečimo stopnjo stiskanja

### 6. koraku

- DC in AC stiskamo ločeno
- vrednosti pašemo v kodirniki v cik-cah zaporedju →
- uporaba RLE in huffman / aritmetični (tega bolj redko zaradi patentov)



## Bolj podrobni pogled na DCT

1. Bloku odštejemo 128
2. Nad blokom izvedemo DCT
3. Istočne elemente iz bloka delimo z elementi v kvantizacijski tabeli:
4. Pretvorimo blok v zig-zag sekvenco
5. DC iz trenutnega bloka je napovedan z DC iz prejšnjega bloka:  $DC_i - DC_{i-1}$  (razen za prvi blok)

### 6. Kodiramo DC in AC koeficiente

DC: (velikost v bitih) (amplituda)

AC: (št. nikel, velikost v bitih) (amplituda)

### 7. Kodiranje simbolov z huffmanom

tabelu zapisana v specu

Amplitude kodiramo kot števila spremenljive dolžine (VLI)

## Naprednje stiskanje

- Kodiranje vsakega kanala v več prehodih
  - spektralna izbira koeficientov DCT
  - zaporedna aproksimacija, skupine bitov od MSB do LSB

## Hierarhično stiskanje

- shranimo sliko v več ločljivostih
- da dekodiramo nižjo ločljivost ne rabimo dekodirati višje ločljivosti
- višje ločljivosti uporabljajo podatke iz nižje ločljivosti
- skupna velikost slike manjša, kot če b:  
vsaka ločljivost posebej shranili
- Vschaša ločljivost lahko koristi naprednije stiskanje
- Koraki:
  - kodiranje nižje ločljivosti
  - kodiranje 2x prejšnje ločljivosti,  
kodiranje razlik z enim od preostalih načinov kodiranja JPEG
  - repeat

## Brez izgubno stiskanje

- uporaba napovednega stiskanja na podlagi sosedov, 8 možnih napovedi:  
 $x = X, A, B, C, A+B-C, A+(B-C)/2,$   
 $B+(A-C)/2, (A+B)/2$
- Napake stiskamo glede na napoved (Huffman, Aritmetični kodirnik)
- slabše stiskanje 2:1, redko uporabljen

	C	B	D	
A	X			

## Omejitve

- Visoka časovna zahtevnost DCT
- Pri večjih stopnjah stiskanja pride do vidnih blokov v sliki
- slabo delovanje lossless
- slabo za risbe in tekst (robovi)
- le 8 bitov na kanal (možno 12 v razšir. fvi)  
16 z lossless
- Ni transparentnost;
- Ni primerno za urejanje, ponovno stiskanje lahko povzroči akumulacijo napak

## Nadgradnje JPEG

### JPEG XT

- Več kvantizacijskih nivojev, HDR, α kanal, DCT v lossless, backwards compatibility,

### JPEG XL

- Namenjeno kot dolgoročna zamenjava za JPEG
- 60% boljše stiskanje, boljše lossless (35%), več kanalov (do 100), bitdepth do 32 podpora za decimalke, animacije, možna pretvorba JPEG → XL, plastičice, 360° slaba podpora
- Ločeni cevovi d za lossless
- Ločen feature encoding, napovedno kodiranje, LZ77 in ANS/Huffman

### JPEG-LS (lossless)

- brezgubno ali skoraj brezgubno kodiranje
- Dva načina stiskanja glede na kontekst
  - navaden način: napovedno kodiranje, golombovo za zapis napak
  - način s ponavljanjem
- Nadgradnja
  - aritmetični kodirnik
- Uporaba LOCO-1 (Low cOMplexity lossless COmpression of Images)

### Posplošen Golomb-Rice

- Kodiramo samo pozitivna števila  $g$ , k-dolžina kod
- $g$  - kodirano z umerno večjim kodo;  $g=3 \Rightarrow 1110$
- Kodiranje r s prisekanom modificirano binarno kodo
  - odprava redundanco zapisa  $n=k$  števil z enakim številom bitov u
  - kodiranje  $n=k$  simbolov s prisekanom modificirano binarno kodo
  - V primeru negativnih št. uporabimo preslikavo  $g^+$

$$g = 3 \\ h = h = 5$$

$$g = 7 \\ r = h$$

$$\lceil \log_2 5 \rceil = 3$$

$$1111110 : 111$$

$$g^+ = \begin{cases} 2g, & g \geq 0 \\ 2g+1, & g < 0 \end{cases}$$

0	00
1	01
2	11
3	110
4	111

### JPEG-LS osnovni koncepti:

- Skenirno prebiranje pikselov, ločeno stiskanje barvnih kanalov
- uporaba konteksta za določitev načina kodiranja (napovedi ali ponavljanje)
- napovedi:
  - napovedi glede na okolico (metrikalni robovi)
  - korekcija napovedi, da se izbalansira entropijski kodirnik
  - popravek na 0
  - uporaba golomb kod
- Ponavljanje - poseben način kodiranja ponavitev

## JPEG-LS modeliranje konteksta

- Modeliramo kontekst glede okolice piksla  $x$
- če nismo ačd vzememo  $b$ , če smo v prvi vrstici:  $c=b=d=0$
- vodenje stanja kodirnika glede na kontekst  $Q_i$ 
  - uporaba 3x7 dolgega polja:
  - A - vsota abs vrednosti napak
  - B - pristnansost - vsota napak po kontekstih;
  - C - kontekstna napovednih vrednosti;
  - N - humulativno št. pojavitev konteksta
  - Posodabljenje polja glede na kontekst
  - Uporaba polj za učinkovitejše kodiranje (konteksto)

## JPEG-LS izbirat kodiranja

- računam gradiente
 
$$g_1 = d - b$$

$$g_2 = b - c$$

$$g_3 = c - a$$
- $\text{če } g_1 = g_2 = g_3 = 0 \rightarrow$  prehlop kodirnika  
sicer napovedni način

		c	b	d		
		a	x			

## Napovedni način

- Fiksna napoved  $x$  (MED - median edge detector)
 
$$x' = \min(a, b, c) + \max(a, b, c) - c$$
  - ideja: napočen rob (svetel ali temen) levo od  $x: b$   
vodoraven rob (svetel ali temen) nad  $x: a$   
ali interpolacija
- Izračun in kodiranje napake:  $\epsilon = x - x'$ , če je večino napak okoli 0 (geometrijsko porazdelitev) bo golombovo kodiranje učinkovito, vendar ponavadi ni tako
- Rešitev: naredimo kontekst  $\Rightarrow \epsilon' = \epsilon - \mu$ 

$$\epsilon' = x' + \mu \quad \text{Poprawek napovedi}$$
- u razbijemo na celo-R (bias) del in decimalni-s (shift) del
- sin R složeno izračuna glede kontekst k (parameter za golom-nice) se tudi izračuna glede kontekst

## → Dolčevanje konteksta $Q$ glede $g_1, g_2$ in $g_3$

- kvantiziramo  $g_1, g_2, g_3$  v 9 regij med -4 in 4
- privzetca kvantizacija za 8-bitne vrednosti  $g_i \rightarrow g'_i$ 
  - $-\{0\} \rightarrow 0$
  - $-\{\pm 1, -2\} \rightarrow \pm 1$
  - $-\{\pm 3, -6\} \rightarrow \pm 2$
  - $-\{\pm 7, -13\} \rightarrow \pm 3$
  - $-\{\pm 15, \pm 21\} \rightarrow \pm 4$

- dolčimo kontekst  $Q = g'_1 g'_2 g'_3$  isto polje kot opisano na začetku poglavje o modeliranju konteksta
- izvedemo kontekst glede polje  $C[Q] = \mu$
- izračun kja za golomb kodirnik  $k = \lceil \log_2(A[Q]/N[Q]) \rceil$
- posodobitve polji
- ko obdelamo piksel posodobimo polja

$$\begin{aligned} N[Q] &= N[Q] + 1 \\ A[Q] &= A[Q] + |E| \\ B[Q] &= B[Q] + E \end{aligned}$$

za C in N:

```

B = B + e; /* accumulate prediction residual */
N = N + 1; /* update occurrence counter */
/* update correction value and shift statistics */
if ( B ≤ -N ) {
    C = C - 1; B = B + N;
    if ( B ≤ -N ) B = -N + 1;
}
else if ( B > 0 ) {
    C = C + 1; B = B - N;
    if ( B > 0 ) B = 0;
}

```

• Ě zapišemo z golombom

### Način pojavljivanja

- Pomicanje po pikstih, dokler so piksti enaki in smo v isti vrstici, ker je vrednost x že znana kodiramo samo št. ponovitev, ki je lahko tudi 0 (preko  $g_1, g_2, g_3$  smo prišli v način pojavljivanja, a  $x$  ne vjema več)
- kodiramo št. ponavljaj m, zapišemo bit 0 dokler se pojavlja, ob prekinitvu pojavljanja zapišena bit 1.
- Po prekinitvi zapisanega vzorca pikstila neujemanja in se vrnemo nazaj na izbiro načina kodiranja, pri zapisu vzorca uporabimo hotekste  $Q=365$  in  $366$  za zapis vrednosti pikstila (uporaba polj A in N)

### → Zapis števila ponovitev

- m - št. ponovitev, ideja je, da m zapišemo s vsoto potenc števila 2
- potence so del polja  $J = \{0, 0, 0, 0, 1, 1, 1, 1, 2, 2, 2, 2, 4, 4, 5, 5, 6, 6, 7, 7, 8, 9, 10, 11, 12, 13, 14, 15\}$
- $i=0$
- dokler je  $m > 2^{J[i]}$ :
  - ne izhod zapisi 1, m zmanjšaj za  $2^{J[i]}$ , povečamo i (največ do zadnjega indeksa J)
- če pride do prekinitve zaradi nove vrstice:
  - če je  $m > 0$  zapišemo 1 na izhod, za novi vzorec spet izberemo način kodiranja
- če nič od zgornjih dveh izpišemo 0, in ostane m zapišemo z  $J[i]$  biti:
- če je  $i > 0$  zmanjšamo za 1

### Primer:

$$\begin{aligned}
 m &= 13 & := 0 & \quad 0 = 111111101 \\
 m &= 12 & i = 1 \\
 m &= 11 & i = 2 \\
 m &= 10 & i = 3 \\
 m &= 8 & i = 4 \\
 m &= 6 & i = 5 \\
 m &= 4 & i = 6 \\
 m &= 0 & i = 7
 \end{aligned}$$

## JPEG-LS skoraj brezizgubno stiskanje

- Pri vsakem pikslu dovolimo največjo sprememblo  $\delta$ :
  - V način ponavljanja prehlopimo, če so pikli znotraj tolerance:  $g \leq \delta$  (brezizguben način kadar  $\delta = 0$ )
  - Uporaba gradientov (?)
  - V načinu napovedi, kvantizacija napak napovedi v  $2\delta+1$  možnih vrednostih.
- JPEG-LS izvaja ponastavitev zaradi boljšega prilaganja lokalni okolci in omejitve implementacije
- $N_A$  in  $B$  se razpolovijo
- razpolovitev naredimo kader  $N=64$

Code segment A.12 – Variables update

```

B[Q] = B[Q] + Errval * (2 * NEAR + 1);
A[Q] = A[Q] + abs(Errval);
if (N[Q] == RESET) → RESET=64
  A[Q] = A[Q] >> 1;
  if (B[Q] >= 0)
    B[Q] = B[Q] >> 1;
  else
    B[Q] = -((1-B[Q]) >> 1);
  N[Q] = N[Q] >> 1;
}
N[Q] = N[Q] + 1;
  
```

## Osnove valčnih transformacij

### DCT - izboljšava

- Z uporabo okenjskih funkcij pri DCT lahko izvemo kje v signalu (čas) kdaj se pojavijo frekvence oz. kosinusne komponente
- Okenjska funkcija  $w(t) = \exp(-t^2)$  ← primer
- Okenjsko funkcijo pomnožimo z vhodnim signalom, ter transformirah signal dano v DCT, torej naredimo DCT samo nad tem kratkim časovnim oknom. Nato ohranimo pomaknemo naprej v času in ponovimo postopek.
- Tako dobimo spektrogram  $s(f, t)$
- Slabost je fiksna velikost okna
  - velika okna - dobra frekvenčna ločljivost, slaba časovna ločljivost
  - majhna okna - obratno

### Valčna transformacija

- Glavna nadgradnja nad STDCT, širša okna pri nižjih frekvencah, očja okna pri višjih frekvencah.
- Izbira osnovnega valjčka (funkcija  $\psi(t)$ ), vrednost funkcija ≠ 0 na omejenem intervalu, namenjena določanju frekvenc na določenem intervalu
- Dela se izračun korelacije med signalom in valjčkom pri različnih premikih in širinah (skaljah), valjček je v vlogi okenjske in bazne funkcije

#### Funkcije valjčkov:

- Meksikanski klobuk

$$\psi(t) = \frac{2}{\sqrt{3}\sigma\pi^{1/4}} \left(1 - \left(\frac{t}{\sigma}\right)^2\right) e^{-\frac{t^2}{2\sigma^2}}$$

- Morletov valček

$$\Psi_\sigma(t) = c_\sigma \pi^{-\frac{1}{4}} e^{-\frac{1}{2}t^2} (e^{i\omega t} - \kappa_\sigma)$$

$$\kappa_\sigma = e^{-\frac{1}{2}\sigma^2} \quad c_\sigma = \left(1 + e^{-\sigma^2} - 2e^{-\frac{3}{4}\sigma^2}\right)^{-\frac{1}{2}}$$

• lastnosti, ki jih morajo valjoki imeti:

$$\int_{-\infty}^{\infty} |\psi(t)| dt = 0 \quad \int_{-\infty}^{\infty} |\psi(t)|^2 dt < \infty$$

### Zvezni WT

• uporaba osnovnega  $\psi(t)$  za analizo signala

- analiza signala  $x(t)$  pri različnih  
skalah in intervalih

- splošen zapis primeren za analizo  
kratkih  $\psi \rightarrow$  lokalizirana energija

- zaradi veliko redundance

$$W_{\psi}(a, b) = \frac{1}{\sqrt{|a|}} \int_{-\infty}^{\infty} x(t) \psi\left(\frac{t-b}{a}\right) dt$$