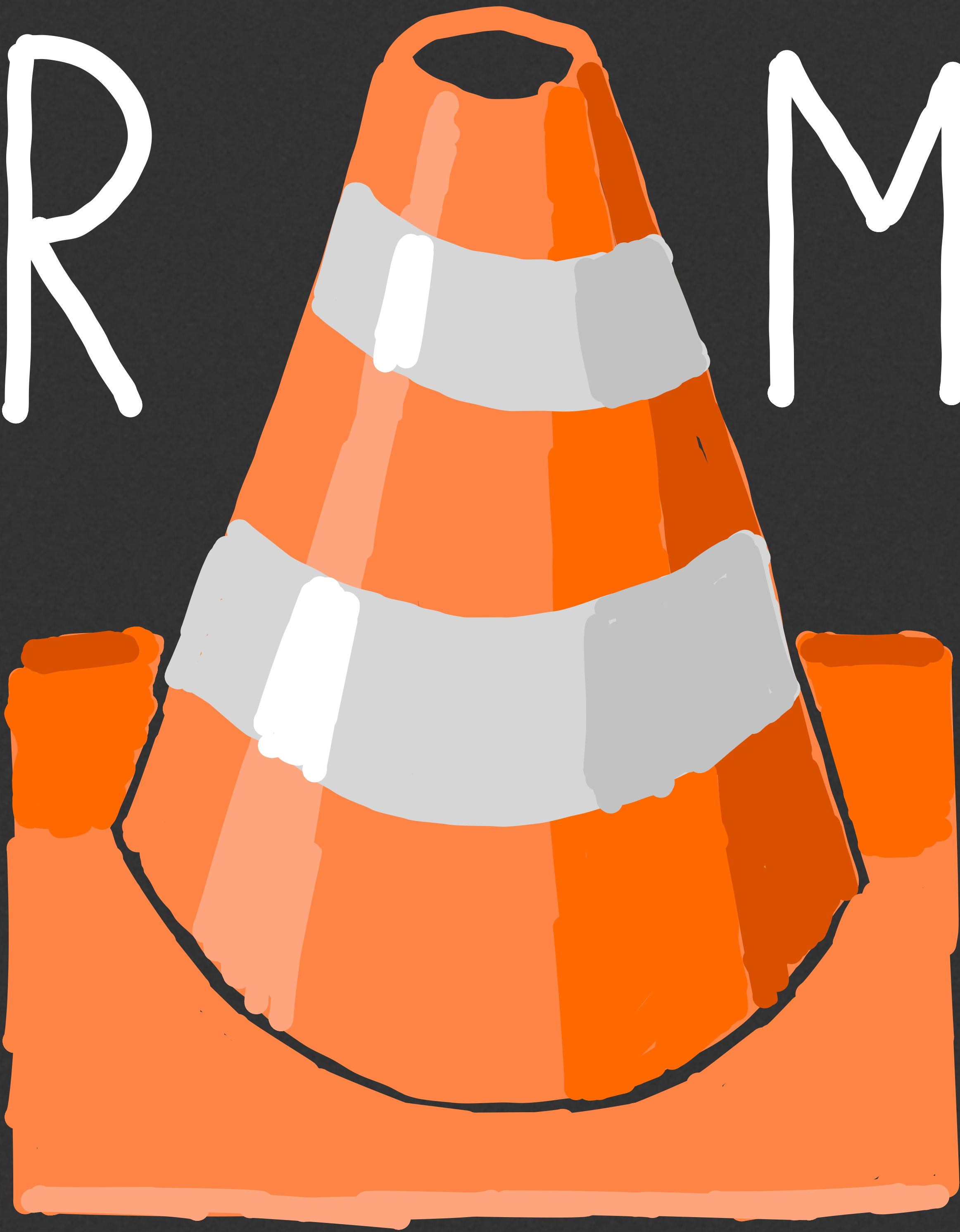
R M



rm -rf /bin/life

Rasterske slike

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

Shranjevanje barv

- Direkten način - predstava z n piksli
  - vsak piksel je 3x8bitov → R G B
- Indeksni način
  - imamo slovar kamor zapisujemo vse unikatne barve slike ↗ m bitna predstavitev barve
  - v matriki se zapiše referenca na barvo v slovarju → n bitov na piksel
  - koristno kadar je n<m
  - lahko predstavimo 2^n odtenkov iz nabora 2^m barv

Vrste rasterskih slik

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

Stiskanje rasterskih slik

- Želimo predstaviti z čim manj bajti
- Novejši pristopi odstranjajo čim več redundanc hkrati
- Vrste postopkov:
  - brezizgubne (lossless)
  - izgubne (lossy)
  - skoraj brezizgubne (near lossless)

1. RLE (run length encoding)

- prebiramo piksle in zapišemo št. ponovitev vsake barve
- dolžine ponovitev dodatno stisnemo s variable length code (VLC)
- primerno za monokromatske slike
  AAABBBBBBAACCCCCC ⇒ 3A5B2A6C
- Kontekstni: RLE - koda za kodiranje določimo glede na sosednje piksle

2. Stiskanje sivinskih slik z n ravninami

- Stiskanje n-bitnih ravnin na podlagi predhodne metode n-kvantizacija piksla
  - predpostavimo, da imajo sosedni px podobne vrednosti:
  - problem, če je npr. 63 in 64 (1000000 in 0111111), rešitev Grayeve kode

Grayeve kode

- Problem, podobne vrednosti, vendar zelo različna bitna predstavitev
  7 => 0111   8 => 1000
- Grayeve zrcalne kode n => št. bitov za število

n=1		n=2		n=3
0 →	0	00	0	000
1 →	1	01	1	001
	1	11	2	011
	0	10 ⇒	3	010 ...
	zrcaljeno		4	110
			5	111
			6	101
			7	100

3. Napovedi

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

Golomb-Rice

- Unarne kode:
  0 - 1
  1 - 01
  2 - 001
  3 - 0001
  ⋮
- Koda Rice
  - MSB je predznak napake Δ
  - iz Δ izločimo n LSB bitov, postanejo LSB kode
  - preostalo kodiramo z unarno kodo

Primer:

$\Delta = -46$ $n=4$

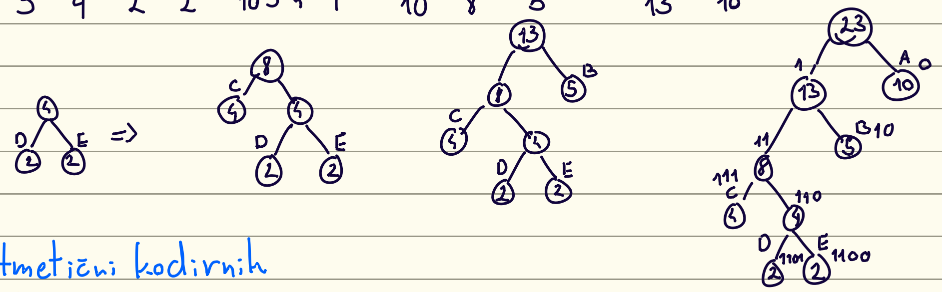
sign=1   101110

sign
1|001|1110

- Kontekstno odvisna napoved
- glede na okolico izberemo različno funkcijo za napoved - horizontalno prelivanje druga funkcija ko vertikalno

Huffman encoding

A	B	C	D	E	⇒	A	B	C	DE	⇒	A	CDE	B		CDEB	A
10	5	4	2	2		10	5	4	4		10	8	5		13	10



Aritmetični kodirnik

Kodiranje

	A	B	C
F_{vz}	2	1	1
%	0,5	0,25	0,25
interv.	[0, 0.5)	[0.5, 0.75)	[0.75, 1)

Ponovi!

$L_{i+1} = L_i + (H_i - L_i) \cdot L(\sigma_i)$
$H_i + 1 = L_i + (H_i - L_i) \cdot H(\sigma_i)$

AABC

$L_0 = 0$
$H_0 = 1$

1. A $L_1 = 0$, $H_1 = 0{,}5$

2. A $L_2 = 0{,}0$, $H_2 = 0{,}25$

3. B $L_3 = 0{,}125$, $H_3 = 0{,}1875$

4. C $L_1 = 0{,}125 + (0{,}1875 - 0{,}125) \cdot 0{,}75$
$H_1 = 0{,}125 + (0{,}1875 - 0{,}125) \cdot 1$
$L_1 = 0{,}171875$
$H_1 = 0{,}1875$

Izberemo vrednost med L in H

npr. $v = 0{,}18$

Dekodiranje 1. način

A	B	C
2	1	1
0,5	0,25	0,25
[0, 0.5)	[0.5, 0.75)	[0.75, 1)

$v_{i+1} = (v_i - L(\sigma_i)) / (H(\sigma_i) - L(\sigma_i))$

$v = 0{,}18$

1. znakje:

1. $v_0 = 0{,}18$ $0 \leq v < 0{,}5$ ⇒ A
2. $v_1 = (0{,}18 - 0) / (0{,}5) = 0{,}36$; $0 \leq 0{,}36 < 0{,}5$; AA
3. $v_2 = (0{,}36 - 0) / (0{,}5) = 0{,}72$; $0{,}5 \leq 0{,}72 < 0{,}75$; AAB
4. $v_3 = (0{,}72 - 0{,}5) / (0{,}75 - 0{,}5) = 0{,}88$; $0{,}75 \leq 0{,}88 < 1$; AABC

$v_4 = (0{,}88 - 0{,}75) / (1 - 0{,}75) = 0{,}52$

Dekodiranje: 2. način

v normaliziramo

$v_1 = 0{,}18$ ⇒ A
$v_2 = 0{,}36$ ⇒ A L_2 H_2
$v_3 = (0{,}18 - 0) / (0{,}25 - 0) = 0{,}72$ ⇒ B
$v_4 = (0{,}72 - 0{,}125) / (0{,}1875 - 0{,}125) = 0{,}88$ ⇒ C
$v_5 = (0{,}88 - 0{,}171875) / (0{,}1875 - 0{,}171875)$

4. Transformacije

- Kodiramo transformirane vrednosti
- Diskretna kosinusna transformacija, furierjeva 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 sliko hranimo samo referenco ta del.

7. Fraktalno stiskanje

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

8. Podvzorčenje

- Brišemo vsako 2. vrstico in stolpec
- Nadgradna - povprečenje

9. Kvantizacija

- Zmanjšamo št. bitov za zapis pikslov
- Izboljsava - vektorska kvantizacija
  - v kodirno knjigo vpišemo najbolj pogoste piksle (npr. 4 zaporedne)
  - zakodiramo sliko kot zaporedje indeksov

Merjenje podobnosti med slikami

za merjenje podobnosti uporabimo RMSE (root mean square error) in PSNR - signal to noise ratio

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

$$MSE = \frac{1}{n} \sum_{i=1}^{n} (X_i - Y_i)^2$$

n – št. px

$$RMSE = \sqrt{MSE}$$

$$SNR = 20 \log_{10} \frac{\sqrt{\frac{1}{n}\sum_{i=1}^{n} P_i^2}}{RMSE}$$

SSIM - (structural similarity index metric)

- PSNR ni za vidno zaznavo prilagojen
- SSIM - ocena vizualne podobnosti, tak ko človek vidi
  - x, y - isti okni zotraj X in Y
  - vrednost med -1 in 1 (1 je identično)
- Delnje na barvnim prostorom YCbCr

- Po sliki se premika z oknom in računa povprečne vrednosti
  - ponavadi 8x8 ali 11x11

$$\text{varianca } \sigma_x = \frac{1}{n} \sum_{0}^{n} (x_i - \mu_x)^2$$

μ_x – povprečna vrednost

μ_x, μ_y - avg okenj

σ_x, σ_y - varianca vrednost v oknih - $\frac{1}{n-1} \sum_{i=0}^{n} (x_i - \mu_x)^2$

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

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

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

$$SSIM(x,y) = \frac{(2\mu_x\mu_y + c_1)(2\sigma_{xy} + c_2)}{(\mu_x^2 + \mu_y^2 + c_1)(\sigma_x^2 + \sigma_y^2 + c_2)}$$

Osnovne transformacije

Nad pari pikslov

- Zaporedne pare pikslov smatramo kot točke na ravnini
- Poiščemo premico, ki gre skozi točk
- Točke zarotiraš na x os. npr za 45° z rotacijsko matriko

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \underbrace{\begin{bmatrix} \cos(\varphi) & -\sin(\varphi) \\ \sin(\varphi) & \cos(\varphi) \end{bmatrix}}_{R} \begin{bmatrix} x \\ y \end{bmatrix} \qquad R^{-1} = R^T$$

Izračun križne korelacije

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

- Kodiranje točk:
  - najprej x nato y ker je y okoli ničle ga lahko kodiramo z manj biti
  - pri manjših vrednostih y sploh ne zapisujemo y ga zaokrožimo navzdol.

- možno tudi v 3D uporabimo 3 zaporedne piksle kot 3D točko
  - uporabimo 2 3x3 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

$w_{i,j}$ - uteži baznih vektorjev

d_i - zaporedne vrednosti pikslov

- W določimo tako, da imamo v prvem c_i čim več energije
- prvi c_i določa osnovno frekvenco ostali višje harmonike
- prva vrstica naj omogoča čim večji $c_1 \rightarrow w_{1j} > 0$ ostali c_i naj bodo manjši polovica $w_{xj} > 0$, polovica $w_{xj} < 0$

$$W = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 \\ 1 & -1 & 1 & -1 \end{bmatrix}$$

Vrstice naj bodo parona ortogonalne

- Ker je energija c zelo visoka ga normaliziramo z $\frac{1}{2} \rightarrow W = \frac{1}{2} * W$

$\vec{d} \cdot \vec{d} = d_i^2 + d_{i+1}^2 + \ldots + d_n^2$

- kolikšen delež energije prinese d_i k $\vec{d}$ $\left(\vec{d} \cdot \vec{d} - (d_j^2 + d_{j+1}^2 + \ldots + d_n^2 - d_i^2)\right) / (\vec{d} \cdot \vec{d})$
- inverz od $W \Rightarrow W^{-1} = W^T$

Primer za dvodimenzionalno transformacijo

- imamo 2D matriko ki je enake dimenzije kot W

$C' = W \cdot D$

$C = C'W^T = WDW^T = WDW$ ← Transformacija

$D = W^T C W = WCW$ ← Inverzna transformacija

Walsh-Hadamardova transformacija (WHT)

- Rekurzivno definirano
- H_m je ortogonalna in ortonormalna $H_m = H_m^{-1}$

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

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

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

$\vdots$

Diskretni WHT

- Optimizacija skalarni faktor kvadriramo oz. šiftamo bite
- Vrstice in stolpci so ortogonalni samo H^{-1} ni več trivialen

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

Kosinusna transformacija in diskretna (DCT)

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

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

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

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

2D DCT

- Uporabljeno na slikah
- Pogosto $n=m=8$, DCT izvedemo nad vrsticami in nato stolpci
  v praksi dodamo novo koordinato podvojimo izraz C in $\cos(\dots)$
- Možno predstaviti tudi z $8 \cdot 8$ matriko 8×8 W matrik

$$F(u,v) = \sqrt{\frac{2}{mn}}\, C(u) C(v) \sum_{x=0}^{n-1}\sum_{y=0}^{m-1} f(x,y) \cdot \cos\left(\frac{\pi(2x+1)u}{2n}\right) \cos\left(\frac{\pi(2y+1)v}{2m}\right); \quad f(x,y) = \sqrt{\frac{2}{mn}} \sum_{u=0}^{n-1}\sum_{v=0}^{m-1} C(u)C(v)F(u,v) \cdot \cos\left(\frac{\pi(2x+1)u}{2n}\right) \cos\left(\frac{\pi(2y+1)v}{2m}\right)$$

JPEG

- najbolj popularen datotečni format za izgubno stiskanje
- JFIF - datotečni format (specs), JPEG: postopek stiskanja
- možno določati stopnje stiskanja, izgubno in brezizgubno (a ločen postopek)
- Podprti načini stiskanja JPEG:
  - Zaporedno stiskanje
  - Napredujoče stiskanje: vsak naslednji prehod vsebuje več infota, primerno za prenos po počasnem omrežju
  - Lossless
  - Hierarhično

Zaporedno stiskanje

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

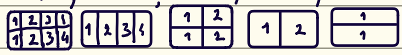
1. Korak

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

2. Korak

- Podvzorčimo Cb in Cr kanale
- Barve lahko bolj stisnemo, ker je človeško oko bolj občutljivo na svetlost
- Najpogostejši zapis podvzorčenja J:a:b
  J - št. komponent svetlost na vrstico (ponavadi J=4)
  a - št. komponent v h-smer, b - št. sprememb v barvnih komp iz prve v drugo vrstico

- podprtoi 4:4:4, 4:4:0, 4:2:2, 4:2:0, 4:1:1



3. Korak

- Sliko razdelimo na bloke, ponavadi veliki 8x8 pikslo. Stiskamo po blokih neodvisno od drugega.
- če ni deljivo z 8 lahko uporabimo različne strategije raztezanja slike

4. korak

- odštejemo 128 od vrednosti vsakega piksla (sinusoide prestavimo na interval [-128,127])
- transformiramo bloke z DCT, levo zgoraj DC componenta, ostalo AC
- imamo malo izgubo zaradi zaokrožitve na cela št.

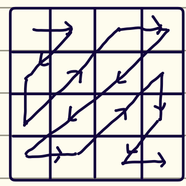
5. korak

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

6. korak

- DC in AC stiskamo ločeno
- vrednosti pašemo v kodirnik v cik-cak zaporedju →
- uporaba RLE in huffman / aritmetični (tega bolj redko zarad patentov)

amplitude padajo v to smer



Bolj podroben pogled na DCT

1. Bloku odštejemo 128
2. Nad blokom izvedemo DCT
3. Istoležne 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. ničel, velikost v bitih)(amplituda)
7. Kodiranje simbolov z huffmanom
   tabela zapisana v specu
   Amplitude kodiramo kot števila spremenljive dolžine (VLI)

Napredujoče stiskanje

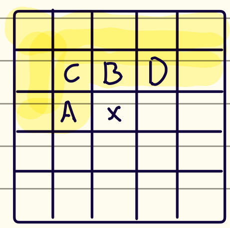
- Kodiranje vsakega kanala v več prehodih
  - spektralna izbira koeficientov DCT
  - zaporedna aproksimacija, skupine bitov od MSB do LSB

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 bi vsako ločljivost posebej shranili
- Vsaka ločljivost lahko koristi napredujoče 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



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 razširitvi) 16 z lossless
- Ni transparentnosti
- Ni primerno za urejanje, ponovno stiskanje lahko povzroči akumulacijo napak

Nadgradnje JPEG

JPEG XT

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

JPEG XL

- Namenjeno kot dolgoročna zamenjava za JPEG
- 60% boljše stiskanje, boljše lossless (35%), več kanalov (do 4100), bitdepth do 32 podpora za decimalke, animacije, možna pretvorba JPEG⇒XL, ploščice, 360°, slaba podpora
- ločeni cevovod za lossless
- ločen feature encoding, napovedno kodiranje, LZ77 in ANS/Huffman

JPEG-LS (lossless)

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

Posplošen Golomb-Rice

- Kodiramo samo pozitivna števila σ, k-dolžina kod — $q = \lfloor \frac{\sigma}{k} \rfloor$
- q - kodirano z unarno vejično kodo; $q=3 \Rightarrow 1110$ — $r = \sigma - qk$
- Kodiranje r s prisekano modificirano binarno kodo — $u = \lceil \log_2 k \rceil$
  - odprava redundance zapisa $n=k$ števil z enakim številom bitov u
- kodiranje $n=k$ simbolov s prisekano modificirano binarno kodo
- V primeru negativnih št. uporabimo preslikavo σ_i^+

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

$\sigma = 39$ $\quad q = 7$ $\quad u = \lceil \log_2 5 \rceil = 3$

$k = h = 5$ $\quad r = 4$

11111110:111

0 →	00
1 →	01
2 →	11
3 →	110
4 →	111

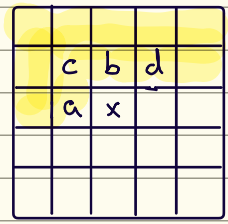
JPEG-LS osnovni koncepti

- skanirno prebiranje pikslov, ločeno stiskanje barvnih kanalov
- uporaba konteksta za določitev načina kodiranja (napovedi ali ponavljanje)
- napovedi:
  - napovedi glede na okolico (vetrikalni robovi)
  - korekcija napovedi, da se izboljša entropijski kodirnik popravek v okolico nič (pr: porazdelitv.)
  - uporaba golomb kod
- Ponavljanje - poseben način kodiranja ponovitev

JPEG-LS modeliranje konteksta

- Modeliramo kontekst glede okolico piksla x
- če nimamo a,c,d vzamemo b, če smo v prvi vrstici c=b=d=0
- vodenje stanja kodirnika glede na kontekst Q:
  - uporaba 367 dolgega polja:
    - A - vsota abs vrednosti napak
    - B - pristranskost - vsota napak po korekciji
    - C - korekcija napovedanih vrednosti
    - N - kumulativno št. pojavitev konteksta
  - Posodabljanje polja glede na kontekst
  - Uporaba polj za učinkovitejše kodiranje (korekcijo)

JPEG-LS izbira kodiranja



- računamo gradiente

  $g_1 = d - b$

  $g_2 = b - c$

  $g_3 = c - a$

- če $g_1 = g_2 = g_3 = 0 \rightarrow$ preklop kodirnika

  sicer napovedni način

Napovedni način

- Fiksna napoved x (MED - median edge detector)

  $x' = \min(a,b,c) + \max(a,b,c) - c$

  - ideja: navpičen 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 večina napak okoli 0 (geometrijska porazdelitev) bo golombovo kodiranje učinkovito, vendar ponavadi ni tako
- Rešitev: naredimo korekcijo $\Rightarrow \epsilon' = \epsilon - \mu$

  $x' = x' + \mu$ ← Popravek napovedi

- μ razbijemo na celi - R (bias) del in decimalni - s (shift) del
- s in R ločeno izračuna glede kontekst

  k (parameter za golom-rice) se tudi izračuna glede kontekst

→ Določanje konteksta Q glede g_1, g_2 in g_3

- kvantiziramo g_1, g_2, g_3 v 9 regij med -4 in 4
- privzeta kvantizacija za 8-bitne vrednosti $g_i \rightarrow g_i'$
  - $\{0\} \rightarrow 0$
  - $\pm\{1\text{-}2\} \rightarrow \pm 1$
  - $\pm\{3\text{-}6\} \rightarrow \pm 2$
  - $\pm\{7\text{-}21\} \rightarrow \pm 3$
  - $\pm\{g_i \mid g_i \geq 21\} \rightarrow \pm g_i$
- določimo kontekst $Q = 81 g_1' + 9 g_2' + g_3'$
- izvedemo korekcijo glede polje $C[Q] = \mu$ ← isto polje kot opisano na začetu poglavja o modeliranju konteksta
- izračun k ja za golomb kodirnik $k = \lceil \log_2(A[Q]/N[Q]) \rceil$

→ posodobitve polji

- ko obdelamo piksel posodobimo polja

$N[Q] = N[Q] + 1$
$A[Q] = A[Q] + |\epsilon|$
$B[Q] = B[Q] + \epsilon$

za C in N:

```
B = B + ε;      /* 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;
}
```

- ϵ zapišemo z golombom

Način ponavljanja

- Pomikanje po pikslih, dokler so piksli 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 ponavljanja a x ne ujema več)
- kodiramo št. ponavljanj m, zapišemo bit 0 dokler se ponavlja, ob prekinitvi ponavljanja zapišeno bit 1.
- Po prekinitvi zapišemo vzorec piksla neujemanja in se vrnemo nazaj na izbiro načina kodiranja, pri zapisu vzorca uporabimo kontekste $Q = 365$ in 366 za zapis vrednosti piksla (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 $m \geq 2^{J[i]}$:
  - na izhod zapiši 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 ostanek m zapišemo z $J[i-1]$ biti

Primer:

$m = 13 - 1 \quad i = 0 \qquad O = 1111111101$
$m = 12 - 1 \quad i = 1$
$m = 11 - 1 \quad i = 2$
$m = 10 - 1 \quad i = 3$
$m = 9 - 2 \quad i = 4$
$m = 7 - 2 \quad i = 5$
$m = 5 - 2 \quad i = 6$
$m = 3 - 2 \quad i = 7$
$i = 8 \Rightarrow 1 < 2^2$

ostanek 1 → zapišemo z biti

JPEG-LS skoraj brezizgubno stiskanje

- Pri vsakem pikslu dovolimo največjo spremembo δ:
  - V način ponavljanja preklopimo, če so piksli znotraj tolerance: $g_i \leq \delta$ (brezizguben način kadar $\delta = 0$) Uporaba gradientov (?)
  - V načinu napovedi, kvantizacija napak napovedi v $2\delta+1$ možnih vrednostih.
- JPEG-LS izvaja ponastavitve zaradi boljšega prilagajanja lokalni okolici in omejitve implementacije
- N, A in B se razpolovijo
- razpolovitev naredimo kadar 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 $w(t) = \exp(0-t^2)$ ← primer
- Okenjsko funkcijo pomnožimo z vhodnim signalom, ter transformiran signal damo v DCT, torej naredimo DCT samo nad tem kratkim časovnim oknom. Nato okno pomaknemo naprej v času in ponovimo postopek.
- Tako dobimo spektrogram $s(f,t)$
- slabost je fiksna velikost okna
  - velika okna - dobra frz ločljivost, slaba časovna ločljivost
  - majhna okna - obratno

Valčna transformacija

- Glavna nadgradnja nad STDCT, širša okna pri nižjih frekvencah, ožja okna pri višjih frekvencah.
- Izbira osnovnega valčka (funkcija $\Psi(t)$), vrednost funkcija $\neq 0$ na omejenem intervalu, namenjena določanju frekvenc na določenem intervalu
- Dela se izračun korelacije med signalom in valčkom pri različnih premikih in širinah (skalah), valček je v vlogi okenjske in bazne funkcije

Funkcije valčkov:

- 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

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

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

• lastnosti, ki jih morajo valjčki imet:

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

Zvezni WT

- uporaba osnovnega $\psi(t)$ za analizo signala
  - analiza signala $x(t)$ pri različnih skalah in intervalih
  - splošen zapis primeren za analizo kratek $\psi \rightarrow$ lokalizirana energija
  - zaradi veliko redundance ni primerno za stiskanje

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