MULTIMEDIÁLNÍ A HYPERMEDIÁLNÍ SYSTÉMY

10)

Komprese videa

Petr Lobaz, 5. 5. 2015

NEKOMPRIMOVANÉ VIDEO

DATOVÝ TOK SD

1 snímek 625/50	$720 \times 576 \times 3$	=	1,18 MB
1 snímek 525/60	$720 \times 480 \times 3$	=	0,99 MB
1 vteřina 625/50	$25 \times 1{,}18$	=	29,66 MB } stejné
1 vteřina 525/60	$30 \times 0{,}99$	=	29,66 MB
1 minuta	$1500 \times 1{,}18$	=	1,73 GB
1 hodina	$90000 \times 1{,}18$	=	104,28 GB

⇒ datový tok 249 Mbps (HD video 1080p25: 1,55 Gbps)

- při podvzorkování 4:2:0 datový tok 124 Mbps
- datový tok CD 1,5 Mbps, DVD 11 Mbps, Blu-ray Disc 40 Mbps

⇒ komprese videa je obvykle nezbytná

KOMPRESE VIDEA

POŽADAVKY

- kontrola toku dat (zaručený kompresní poměr)
- rychlost komprese/dekomprese
  - dekomprese musí obvykle být v reálném čase i na relativně slabém hardwaru
  - komprese v reálném čase je nutná pro přímé přenosy (studiová: silný hardware; videokonference: slabý hardware)
- odolnost vůči chybám přenosu – jedna chyba smí znehodnotit jen malou část videa
- musí umožňovat mechanismus pro synchronizaci se zvukem (obvykle řeší systémová vrstva – kontejner)

KOMPRESE VIDEA

TYPY POUŽITÍ

- pro jednosměrný přenos (např. televize)
  - odolnost vůči chybám, škálovatelnost
  - MPEG-2, Windows Media Video, H.264, H.265, WebM
- obousměrný přenos (např. videokonference)
  - realtime komprese/dekomprese
  - H.264, MPEG-4 Video
- archivace (např. DVD, Blu-ray)
  - dobrý kompresní poměr
  - MPEG-2, H.264, VC-1, H.265
- dočasný formát (např. střih)
  - náhodný přístup do souboru, minimální ztráta
  - MJPEG, AVC-Intra, Apple ProRes, DNxHD (VC-3)

KOMPRESE VIDEA

CODEC

- compressor/decompressor
- typicky tři typy
  - video
  - audio
  - systém: kontejnerová struktura, která řeší uložení elementárních mediálních proudů + mechanismus jejich synchronizace
- systémový typ se spíše než pojmem codec označuje jako muxer / splitter (nebo multiplexor / demultiplexor)
- pro kódování / dekódování multimediálních dat je třeba zaručit podporu všech zúčastněných kodeků

TYPY KOMPRESE

BEZZTRÁTOVÁ

- samostatně minimální použití
  - typicky pro záznam obrazovky (screen capture)
  - archivace lékařských a vojenských dat
- RLE, VLC (Huffmanovo kódování apod.), aritmetické kódování – doplňkové metody nebo nepříliš rozšířená kompresní schémata
- LZW – ve formátu GIF
  - jednoduché internetové animace
  - přes svou zastaralost jde o přenositelný a nenáročný způsob prezentace jednoduchého videa bez zvuku
- bezztrátová DWT – MJPEG 2000
- celočíselná obdoba DCT – speciální typ H.264

TYPY KOMPRESE

ZTRÁTOVÁ

- vector quantization (VQ)
  - rozdělení obrazu na „bloky“ pixelů
  - tvorba tabulky reprezentativních „bloků“
  - místo bloku kódujeme číslo nejpodobnějšího bloku v tabulce
  - potenciálně silná metoda, problémy s efektivní tvorbou tabluky a jejím uchováváním
  - při jednoduché implementaci velmi rychlá (nenáročná) dekomprese
  - dnes se prakticky nevyužívá (ale v budoucnu to může být jinak)
  - historie: Intel Indeo 3.2, Cinepak, Sorenson Video 1

TYPY KOMPRESE

- rozdělení obrazu na bloky + diskrétní kosinová nebo obdobná transformace + kvantizace koeficientů
  - princip obdobný JPEG
  - dnes zdaleka nejvyužívanější metoda
  - MJPEG, DV, H.261–5, MPEG*x*, VP*x*, …
- diskrétní waveletová transformace + kvantizace koeficientů
  - princip obdobný JPEG2000
  - MJPEG 2000 – de facto sekvence snímků JPEG2000
  - Redcode (a jiné RAW formáty profesionálních kamer)
  - Digital Cinema System (distribuce filmů do kin)

PRINCIP KOMPRESE

ODSTRANĚNÍ REDUNDANCE (DEKORELACE)

- intra kódování
  - odstranění prostorové (spatial) redundance
  - typický obraz ve videu je rozmazaný pohybem, je místy ostrý a místy neostrý
  - principy stejné jako u komprese statického obrazu
- inter kódování
  - odstranění časové (temporal) redundance
  - typické po sobě jdoucí snímky jsou téměř stejné
  - po odstranění časové redundance se snímek nejčastěji dále podrobuje odstranění prostorové redundance
- při kódování snímků je navíc nutné zařídit, aby kvalita po sobě jdoucích snímků byla srovnatelná

INTRA KÓDOVÁNÍ

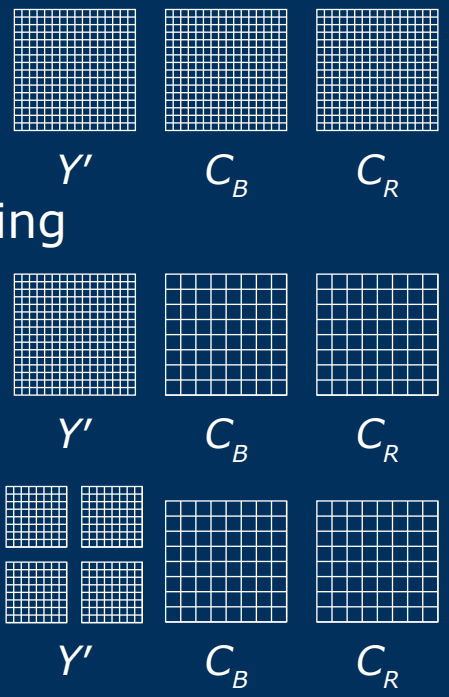
INTRA KÓDOVÁNÍ

- stejné jako komprese statického obrazu
- výhoda: jednoduchost, náhodný přístup do videa
  ⇒ využití pro střih, postprodukci
- nevýhoda: velký datový tok
- parametr „ztrátovosti“ lze dynamicky snímek po snímku měnit ⇒ datový tok konstantní za jednotku času
- čím je datový dok variabilnější, tím větší musí být vyrovnávací paměti pro záznam/přehrávání
- někdy je zapotřebí velmi stabilní datový tok
  - obvykle: konstantní velikost každého snímku
  - ještě přísněji: konstantní velikost části snímku

INTRA KÓDOVÁNÍ

ZÁKLADNÍ POSTUP

- převod obrazu do Y′C_BC_R
- rozdělení obrazu na makrobloky (MB)
  - příklad: 16 × 16 pixelů
- aplikace zvoleného typu chroma subsampling
  - příklad: po 4:2:0 vznikne
    16 × 16 hodnot Y′,
    8 × 8 hodnot C_B, 8 × 8 hodnot C_R
- rozdělení makrobloku na bloky
  - příklad: dělení na bloky
    8 × 8 hodnot, vzniknou 4 bloky Y′,
    1 blok C_B, 1 blok C_R
  - aby dělení na bloky vyšlo pěkně, musí být tvar makrobloku kompatibilní s typem chroma subsampling



INTRA KÓDOVÁNÍ

- aplikace DCT 8×8 vzorků na jednotlivé bloky
- kvantizace koeficientů
  - pro Y’ typicky jiná kvantizační tabulka než pro C_B a C_R
  - kvantizační tabulka se typicky mění mezi snímky i v rámci snímku
- převod bloků do 1D proudu dat
- aplikace bezztrátové kompresní metody, typicky VLC (variable length code) – Huffmanovo kódování, aritmetické kódování

INTRA KÓDOVÁNÍ

KONSTANTNÍ DATOVÝ TOK

- nejjednodušší řízení: každý snímek má být průměrně dlouhý *S* bitů
  - je-li delší, pro další snímek se kvantizace „zhorší“
  - je-li kratší, pro další snímek se kvantizace „zlepší“
  - ⇒ datový tok je pak průměrně konstantní, kvůli střihu nelze vyloučit prudkou změnu
  - kvůli zamezení skokové změny kvality se obvykle definuje, jak se smí kvantizace mezi snímky maximálně změnit

INTRA KÓDOVÁNÍ

KONSTANTNÍ DATOVÝ TOK

- lepší řízení: každý makroblok má být průměrně dlouhý *M* bitů ⇒ stejný princip jako výše aplikovaný na makrobloky
  - nevýhoda: typický obraz má složité (členité) i jednoduché (např. jednobarevné) makrobloky ⇒ není moc chytré požadovat konstantní délku makrobloku
  - řešení: je-li makroblok delší než *M* bitů; uloží se přebývající bity k následujícímu makrobloku
  - nevýhoda: makrobloky vedle sebe mají typicky podobnou složitost
  - řešení: průchod makrobloky v pseudonáhodném pořadí + průběžné řízení kvantizace

INTRA KÓDOVÁNÍ

- nevýhody posledního řešení
  - při pseudonáhodném průchodu makrobloky kodér neví, jak složitý makroblok má v dalším kroku očekávat
  - vzniká značně složitá struktura umístění bitů v makroblocích
- finální řešení:
  - dělení obrazu na superbloky (SB, např. 9×3 makrobloky)
  - rozdělení SB na makrobloky (MB): SB_1MB_1, SB_1MB_2, …
  - v prvním průchodu se zpracuje první MB z každého SB, pořadí SB pseudonáhodné
  - v druhém průchodu se zpracuje druhý MB z každého SB
  - …

INTRA KÓDOVÁNÍ

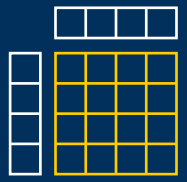
PROKLÁDANÝ OBRAZ

- kódování po půlsnímcích nevýhodné, je-li obraz statický (protože půlsnímky jsou si značně podobné)
- kódování po snímcích nevýhodné, je-li obraz dynamický (protože snímek složený z půlsnímků má „zoubkované" okraje objektů)
- řešení:
  - pro každý MB snímku rozhodnout, je-li statický/ dynamický (resp. jestli obsahuje „zoubkované" okraje)
  - statický MB kódovat normálně
  - dynamický MB rozdělit na sudé/liché řádky, kódovat je odděleně, nebo kódovat jejich součet a rozdíl

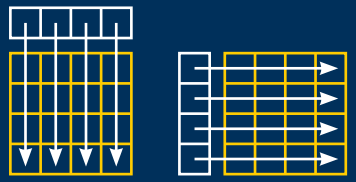
INTRA KÓDOVÁNÍ

PREDIKCE V RÁMCI OBRAZU

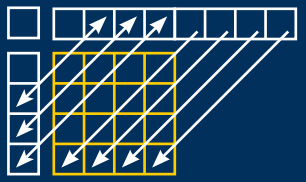
- predikce DC koeficientu bloku – lze jej ukládat diferenciálně oproti předchozímu (jako v JPEG)
- predikce vzhledu bloku na základě okolí (H.264 apod.): v okamžiku dekódování bloku jsou již dekódované bloky vlevo a nad aktuálním; lze ukládat jen rozdíl pixelů oproti pixelům odhadnutým z okolních pixelů



pixely bloku jsou podobné průměru pixelů okolí



pixely bloku jsou podobné situaci, kdy se pixely z okolí „roztáhnou" ve směru indikovaném šipkami; kodér má na výběr několik úhlů šipek (i jiné, než jsou na obrázku)



pro některé úhly a polohy šipek lze navíc hodoty odhadu interpolovat z okolních pixelů

INTER KÓDOVÁNÍ

INTER KÓDOVÁNÍ

- po sobě jdoucí snímky F, G jsou si podobné
  ⇒ využití rozdílového kódování
- typicky se řeší po makroblocích
  – velikost stejná jako u intra kódování
- místo makrobloku G_n kódujeme $F_n - G_n$
  (rozdíly hodnot pixelů), typicky pomocí DCT

1	1	1	1	1	1	1	1
1	1	9	9	1	1	1	1
1	1	9	9	1	1	1	1
1	1	1	1	1	1	1	1

F_1 F_2

snímek F rozdělený na makrobloky F_1 a F_2

1	1	1	1	1	1	1	1
1	1	9	9	1	0	1	1
1	2	8	9	1	1	1	1
1	1	1	1	1	1	0	2

G_1 G_2

snímek G (kvůli šumu není identický se snímkem F)

0	0	0	0	0	0	0	0
0	0	0	0	0	1	0	0
0	-1	1	0	0	0	0	0
0	0	0	0	0	0	1	-1

$F_1 - G_1$ $F_2 - G_2$

rozdílové makrobloky $F_n - G_n$ obsahují malá čísla
⇒ výhodnější pro kódování

INTER KÓDOVÁNÍ

- předchozí postup výhodný u statické scény
- nevýhodný, došlo-li k pohybu (hodnoty F_n – G_n velké)

1	1	1	1	1	1	1	1
1	1	9	9	1	1	1	1
1	1	9	9	1	1	1	1
1	1	1	1	1	1	1	1

snímek F

1	1	1	1	1	1	1	1
1	9	9	1	1	0	1	1
1	9	8	2	1	1	1	1
1	1	1	1	1	1	0	2

snímek G (kvůli šumu a pohybu není identický s F)

0	0	0	0	0	0	0	0
0	-8	0	8	0	1	0	0
0	-8	1	7	0	0	0	0
0	0	0	0	0	0	1	-1

rozdílový makroblok F_1–G_1 obsahuje velká čísla
⇒ nevhodný pro kódování

INTER KÓDOVÁNÍ

- řešení: pro výpočet rozdílového makrobloku ke kódování G_n nepoužijeme makroblok F_n, ale jinou část snímku F (označíme ji F_n^{MV})
- motion compensation – nalezení motion vektoru MV, který minimalizuje $F_n^{MV} - G_n$

MV →

1	1	1	1	1	1	1	1
1	1	9	9	1	1	1	1
1	1	9	9	1	1	1	1
1	1	1	1	1	1	1	1

snímek F
čárkovaně naznačen F_1^{MV}

1	1	1	1	1	1	1	1
1	9	9	1	1	0	1	1
1	9	8	2	1	1	1	1
1	1	1	1	1	1	0	2

snímek G (kvůli šumu a pohybu není identický s F)

0	0	0	0	0	0	0	0
0	0	0	0	0	1	0	0
0	0	1	0	0	0	0	0
0	0	0	0	0	0	1	-1

rozdílový makroblok $F_1^{MV} - G_1$
obsahuje malá čísla
⇒ vhodný pro kódování

INTER KÓDOVÁNÍ

NÁVAZNOST NA DEKOMPRESOR

1. dekompresor rekonstruuje snímek F
2. po přijetí rozdílových makrobloků ($F_n^{MV} - G_n$) a motion vektorů rekonstruuje makrobloky snímku G:
   $G_n = F_n^{MV} - (F_n^{MV} - G_n)$

- *takto to ale nefunguje!*
- bloky ztrátově uložené (tj. poškozené kompresí)

⇒ dekompresor zná jen poškozené makrobloky ${}^{Z}F_n$ a ${}^{Z}(F_n^{MV} - G_n)$

⇒ rekonstrukce G_n nesmyslná:
$G_n = {}^{Z}F_n^{MV} - {}^{Z}(F_n^{MV} - G_n) = ?$

⇒ dekompresor nemůže rekonstruovat ${}^{Z}G$, protože nezná F

- na stejný problém jsme narazili u kódování zvuku DPCM (viz přednáška 5)

INTER KÓDOVÁNÍ

- řešení: kompresor obsahuje i dekompresor
- postup kompresoru:
  - ztrátové uložení snímku F, odeslání na výstup
  - okamžitá dekomprese ⇒ kompresor zná ^{Z}F (tj. to, co bude znát i dekompresor)
  - kódování snímku G pomocí rozdílových makrobloků $^{Z}F_n^{MV} - G_n$, odeslání na výstup (tj. motion compensation probíhá proti snímku ^{Z}F)
  - okamžitá dekomprese ⇒ kompresor zná ^{Z}G
  - kódování snímku H pomocí rozdílových makrobloků $^{Z}G_n^{MV} - H_n$, odeslání na výstup
  - …

⇒ kompresor i dekompresor musí obsahovat paměť pro „předchozí snímek“

MOTION COMPENSATION

MOTION COMPENSATION

- časově nejnáročnější část komprese
- neslouží k identifikaci pohybu, ale k minimalizaci velikosti zakódovaného rozdílového makrobloku
- na výstup se odesílá rozdílový makroblok + nejlepší nalezený motion vektor
- kvalitnější motion compensation
  ⇒ menší rozdílový snímek ⇒ lepší kompresní poměr
- na druhou stranu:
  - „kvalitnější" motion compensation typicky prohledává více kandidátů na „nejlepší motion vektor"
  ⇒ pro uložení MV bude třeba více bitů
  ⇒ zhoršení kompresního poměru

⇒ vyladění motion compensation je poměrně náročné

MOTION COMPENSATION

- základní algoritmus: prohledání všech MV z intevalu $(-x_{max}, -y_{max})$ až $(+x_{max}, +y_{max})$
- např. pro $x_{max} = y_{max} = 16$ je třeba najít nejlepší MV z $(16 + 16 + 1) \times (16 + 16 + 1) = 1089$ kandidátů
- informace o tom, který MV byl vybrán, zabere 11 bitů
- volba $x_{max} = y_{max} = 16$ je relativně vhodná pro video s minimem pohybu, např. videokonferenci, pro obecné video (např. film) je rozsah prohledávání malý
  ⇒ větší časová náročnost, větší náročnost uložení MV
  (např. H.264 používá $x_{max} = 2048$)

- pro $x_{max} = y_{max} = 1024$ by bylo třeba prohledat 4 198 401 kandidátů, informace o vybraném MV by zabírala 23 bitů
  srovnej: typický blok v JPEG zabere cca 64 bitů

MOTION COMPENSATION

URYCHLENÍ A VYLEPŠENÍ

- hledá se MV, který pravděpodobně povede ke krátkému kódu rozdílového makrobloku *R* (velikosti M × N vzorků)
  - ideální MV vede na nulový rozdílový makroblok (v praktických situacích takový MV neexistuje)
  - dobrý MV vede na rozdílový makroblok s „malými čísly"
  - existují různá kritéria, nedá se říct, že jedno je lepší než druhé:

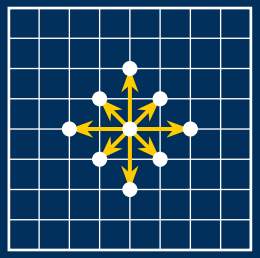
$$MAD = \frac{1}{MN}\sum_i \sum_j \left|R[i, j]\right| \qquad \text{(mean absolute difference)}$$

$$MSE = \frac{1}{MN}\sum_i \sum_j \left(R[i, j]\right)^2 \qquad \text{(mean squared error)}$$

  - snaha o nalezení *R* s minimální hodnotou kritéria

MOTION COMPENSATION

- předpoklad: pro jistý MV je hodnota kritéra minimální, čím je MV odlišnější, tím je hodnota kritéria vyšší
  ⇒ hledání minima funkce přes povolený rozsah MV
- základní urychlení motion compensation:
  1. výpočet kritéria pro MV = (0,0) a několik okolních MV (příklad: „diamond search“)
  2. výběr MV, kde bylo kritérium nejmenší, „posun středu vyhledávání“, pokračování v hledání od nového středu
  3. jakmile je kritérium nejmenší „uprostřed“, prohledání všech zbývajících těsně sousedících MV



makroblok 8×8 vzorků
● = konec MV

MOTION COMPENSATION

- další urychlení: dá se předpokládat, že hodnota kritéria se spojitě mění ⇒ interpolací či extrapolací hodnoty kritéria se dá odhadnout, kde pokračovat v hledání
- další urychlení:
  1. tvorba variant snímků F, G s 1/2 a 1/4 rozlišením
  2. motion compensation na snímcích s 1/4 rozlišením
     ⇒ nejlepší nalezené MV jsou kandidáti na optimum
     (pokud má makroblok v plném rozlišení 16×16 vzorků, má nyní jen 4×4 vzorky, tj. výpočet kritéria je 16× rychlejší, rozsah MV je 16× menší)
  3. motion compensation na snímcích s 1/2 rozlišením, hledá se jen v těsném okolí kandidátů z kroku 2
  4. motion compensation na snímcích s plným rozlišením, hledá se jen v těsném okolí kandidátů z kroku 3

MOTION COMPENSATION

- vylepšení – lze pokračovat krokem 5:
  5a) výpočet, jak by vypadal předchozí snímek (F) posunutý o 1/2 pixelu nahoru, doprava, šikmo doprava
  5b) motion compensation mezi aktuálním snímkem (G) a posunutými verzemi snímku F, hledá se jen v okolí kandidátů z kroku 4
  výsledek: MV s přesností 1/2 pixelu (halfpel)
  příklad: ideální MV je (–5, 12) proti obrázku posunutému o 1/2 pixelu nahoru ⇒ MV je (–5, 12.5)
- další vylepšení: výpočet, jak by vypadal snímek F posunutý o vektory (±1/4, 0), (0, ±1/4), (±1/4, ±1/4)
  ⇒ MV s přesností 1/4 pixelu (qpel)
- některé kodéry pokračují i na 1/8 pixelu

MOTION COMPENSATION

- se zvyšující se přesností MV stále roste počet bitů nutných k jeho uložení, ale:
  - MV je typicky podobný MV okolních makrobloků
  - MV makrobloku je typicky podobný MV téhož makrobloku z předchozího snímku
- důsledky:
  - MV se může odhadnout z okolních MV, které bude mít dokodér k dispozici, a ukládat pouze rozdíl
  - při motion compensation jsou první kandidáti na optimální MV odhadnuti z MV okolních makrobloků

MOTION COMPENSATION

- další urychlení:
  - motion compensation probíhá jen na luma vzorcích (Y'), pro chroma vzorky se použije stejný MV
  - vodorovné pohyby jsou častější ⇒ redukce rozsahu MV ve svislém směru
  - použití stejného MV pro všechny bloky makrobloku
- moderní kodeky je obvykle nevyužívají, tj.:
  - mohou hledat MV pro luma a chroma vzorky zvlášť
  - v případě potřeby má každý blok svůj MV
  - je-li to vhodné, může se makroblok hierarchicky dělit na subbloky (typicky až do velikosti 4×4 vzorky), každý subblok může mít svůj MV

  ⇒ lepší kompresní poměr, ale výpočetně mnohem náročnější

MOTION COMPENSATION

- přes všechnu snahu se může stát, že „optimální MV" vede k příliš velkému rozdílovému makrobloku
  ⇒ potom je výhodnější blok uložit „normálně", tj. bez predikce
  – typicky, pokud se náhle změnil obraz (střih) nebo se do snímku pohybem dostal nový objekt

1	1	1	1	1	1	1	1
9	9	1	1	1	1	1	1
9	9	1	1	1	1	1	1
1	1	1	1	0	2	1	1

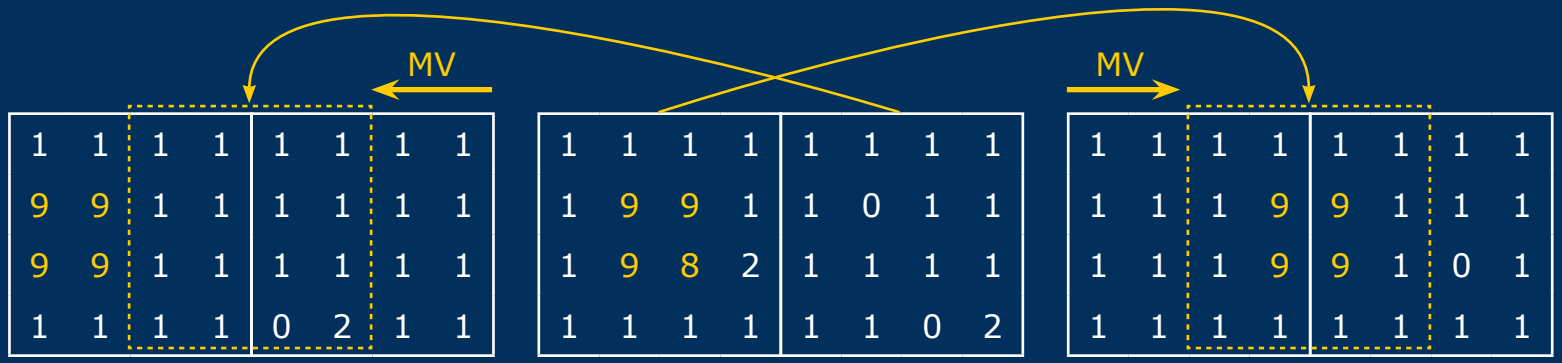
snímek F

1	1	1	1	1	1	1	1
1	9	9	1	1	0	1	1
1	9	8	2	1	1	1	1
1	1	1	1	1	1	0	2

k makrobloku G_1 nelze najít vhodný MV do snímku F

OBOUSMĚRNÉ KÓDOVÁNÍ

- řešení: makroblok snímku G může být kódovaný rozdílově oproti snímku H (následujícímu po G)



snímek F
čárkovaně naznačen F_2^{MV}

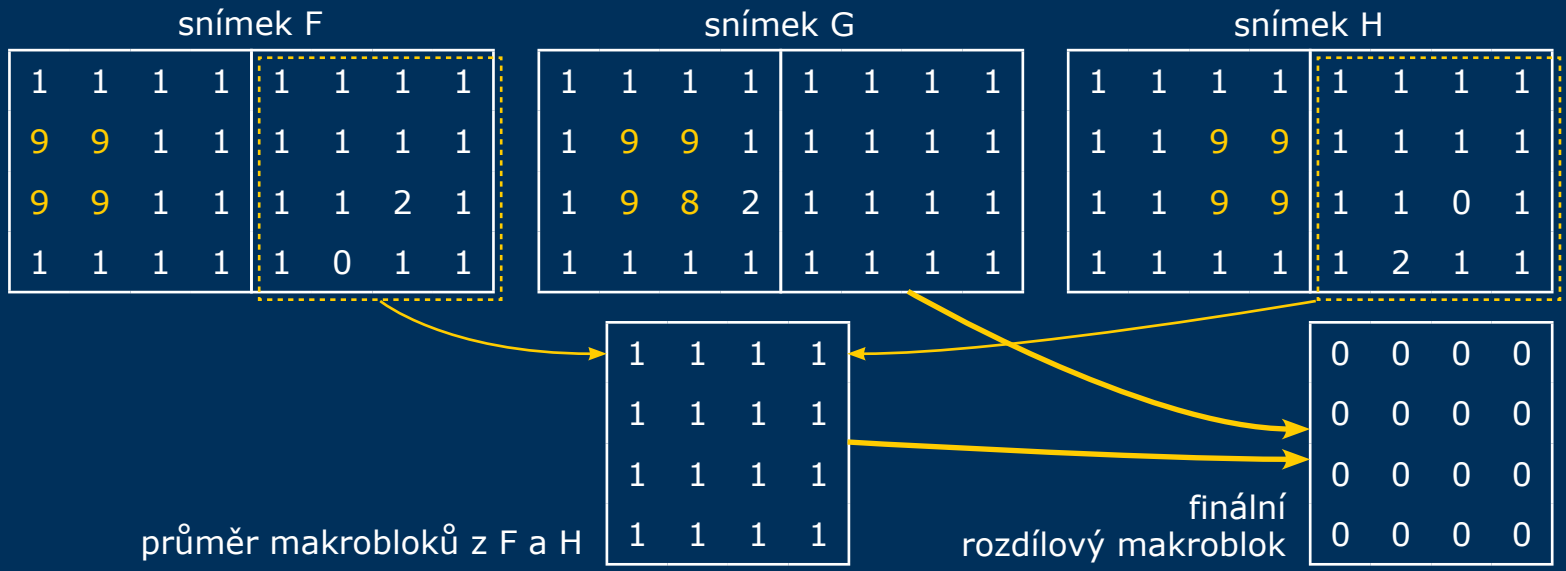
makroblok G_1 se bude kódovat s MV do snímku H, G_2 s MV do snímku F

snímek H
čárkovaně naznačen H_1^{MV}

- jak je vidět na příkladu makrobloku G_2, motion vektor vůbec nemusí souviset s pohybem!

OBOUSMĚRNÉ KÓDOVÁNÍ

- makroblok může být kódovaný
  - „normálně“, rozdílově proti makrobloku z předchozího nebo z následujícího snímku
  - rozdílově proti průměru makrobloků z následujícího a předchozího snímku (průměrem se redukuje šum!)



KÓDOVÁNÍ SNÍMKŮ

- standardní názvosloví:
  - I snímek: všechny makrobloky jsou uloženy „normálně“
  - P snímek: makrobloky uloženy buď „normálně“, nebo rozdílově oproti předchozímu snímku;
    typicky 3× menší než I snímek
  - B snímek: makrobloky uloženy buď „normálně“, nebo rozdílově oproti předchozímu/následujícímu snímku nebo průměru makrobloků;
    typicky 3× menší než P snímek
- značení X_Y – snímek č. Y je typu X
- příklad sekvence snímků: $I_0 B_1 B_2 P_3 B_4 B_5 P_6 B_7 B_8 I_9 B_{10} B_{11} P_{12} \ldots$

- problém: jak zařídit, aby po sobě jdoucí snímky P/B nezávisely cyklicky jeden na druhém?

KÓDOVÁNÍ SNÍMKŮ

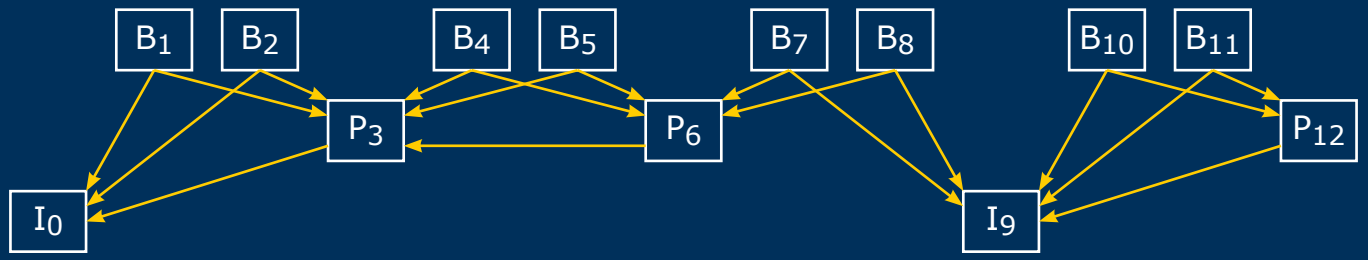
(ZPĚTNÉ) PREDIKTIVNÍ KÓDOVÁNÍ

- nevyužívá B snímky ⇒ nemohou vzniknout cykly, kodér a dekodér jsou jednodušší
- sekvence $I_0 P_1 P_2 P_3 P_4 P_5 P_6 \ldots$
- výhoda: neustálé vylepšování kvality (statického) obrazu – snímek I_0 je uložen ztrátově; P_1 ztrátu trochu opraví atd.
- v případě chyby poškozen zbytek videa
- nemožnost náhodného přístupu do videa
- nemožnost rozdělit video (není zaručeno, že snímek *n* půjde zakódovat jako I_n v přesně stejné kvalitě)
- částečené řešení: vynucené vkládání snímku bez predikce – např. sekvence $I_0 P_1 P_2 P_3 \ldots P_{24} I_{25} P_{26} P_{27} \ldots$

KÓDOVÁNÍ SNÍMKŮ

OBOUSMĚRNÉ PREDIKTIVNÍ KÓDOVÁNÍ

- využívání B snímků, ale
  - P snímek může záviset jen na předchozím I/P snímku
  - B snímek může záviset jen na okolním I/P snímku
- sekvence I_0 B_1 B_2 P_3 B_4 B_5 P_6 B_7 B_8 I_9 B_{10} B_{11} P_{12} …



příklad diagramu závislosti snímků

- nevýhoda: dekodér musí v paměti držet snímky B_1, B_2, dokud nepřijme snímek P_3 ⇒ až pak může dekódovat a zobrazovat ⇒ dekodér musí mít velkou vyrovnávací paměť

KÓDOVÁNÍ SNÍMKŮ

- řešení: kodér ukládá snímky v takovém pořadí, aby je dekodér mohl okamžitě dekódovat
- dekodér musí obsahovat pouze vyrovnávací paměti pro předchozí (F) a následující (H) snímek
- ke každému snímku musí být přiložena časová značka *t*, kdy se má snímek zobrazit
- příklad vstupu: I_0 P_3 B_1 B_2 P_6 B_4 B_5 I_9 B_7 B_8 P_{12} B_{10} B_{11} …
  činnost dekodéru:

I_0	dekódovat, uložit do F, zobrazit v čase $t = 0$
P_3	dekódovat, uložit do H, naplánovat zobrazení na $t = 3$
B_1	dekódovat, neukládat, zobrazit v čase $t = 1$
B_2	dekódovat, neukládat, zobrazit v čase $t = 2$
–	zobrazit H podle plánu v čase $t = 3$, přesunout H do F
P_6	dekódovat, uložit do H, naplánovat zobrazení na $t = 6$

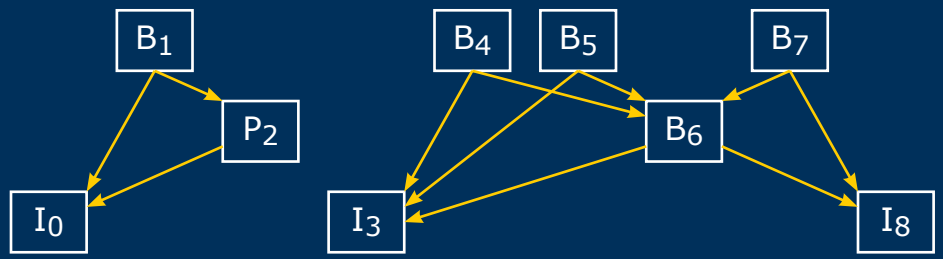
KÓDOVÁNÍ SNÍMKŮ

- nevýhody sekvence I_0 B_1 B_2 B_3 B_4 B_5 … B_{N-1} I_N …
  - kodér nemůže zakódovat B_1, dokud nezakóduje I_N
    ⇒ velké zpoždění, problém u přímého přenosu
  - sekvence BBBB… nezvyšuje kvalitu obrazu
    (na rozdíl od sekvence PPPP…)
  - pokud I_0 a I_N zobrazují stejnou scénu s různým jasem (např. plynule se změnilo osvětlení scény), potom B snímky z prostředka sekvence nenajdou vhodný MV ani do I_0, ani do I_N
- B snímky jsou ale typicky krátké ⇒ je vhodné hledat řešení

KÓDOVÁNÍ SNÍMKŮ

VYLEPŠENÉ OBOUSMĚRNÉ PREDIKTIVNÍ KÓDOVÁNÍ

- u klasického obousměrného kódování nesmí B snímek záviset na jiném B snímku – zbytečné omezení
- řešení (H.264 aj.): B snímek smí záviset na jiném B snímku, ale závislosti se nesmí zacyklit



příklad diagramu závislosti snímků

- uložení: I_0 P_2 B_1 I_3 I_8 B_6 B_4 B_5 B_7 …

KÓDOVÁNÍ SNÍMKŮ

- další vylepšení:
  - standardní B snímek mohl záviset na aritmetickém průměru z makrobloků ${}^{Z}F_{n}^{MV}$ a ${}^{Z}H_{n}^{MV}$, tj. na $0{,}5\ {}^{Z}F_{n}^{MV} + 0{,}5\ {}^{Z}H_{n}^{MV}$
  - uprostřed sekvence B snímků je lepší vážený průměr: $\omega\ {}^{Z}F_{n}^{MV} + (1 - \omega)\ {}^{Z}H_{n}^{MV}$ pro $0 < \omega < 1$

KÓDOVÁNÍ SNÍMKŮ

GOP (GROUP OF PICTURES)

- samostatně dekódovatelná sekvence na sobě závislých snímků (např. od I do I snímku)
- problém: sekvence I_0 B_1 B_2 P_3 B_4 B_5 P_6 B_7 B_8 I_9 B_{10} P_{12} … nelze jednoduše rozdělit (GOP vyznačen žlutě)
- řešení 1: closed GOP
  - GOP ukončen P snímkem:
    I_0 B_1 B_2 P_3 B_4 B_5 P_6 I_7 B_8 B_9 P_{10}…
  - lze bez problémů dělit na hranici GOP
  - ⇒ snadný hrubý střih (např. vložení reklamy do pořadu)
- řešení 2: krátký GOP
  - sekvence I_0 B_1 I_2 B_3 I_4 B_5…
  - při rozdělení snadný přepočet B snímku
  - ⇒ relativně snadný přesný střih (studiová práce)

KÓDOVÁNÍ SNÍMKŮ

STRUKTURA GOP

- stálá
  - implementačně jednoduché, relativně výhodné pro samostatné záběry
  - typicky zadána délkou GOP (typicky 1–30) a délkou sekvence B snímků (typicky 1–3)
  - algoritmus se i tak musí vypořádat se situací typu „záblesk světla“ (tj. snímek je najednou přesvětlený)
- proměnná
  - nutná pro pořad se střihy
  - obtížnější implementace
  - komprese často víceprůchodová (1. průchod: analýza, 2. průchod: samotná komprese)

KÓDOVÁNÍ SNÍMKŮ

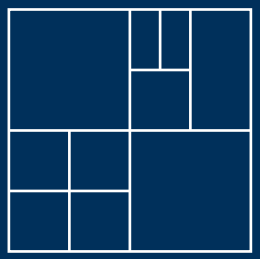
- krátký GOP
  - horší kompresní poměr
  - rychlejší náhodný přístup k jednotlivým snímkům

  ⇒ výhodnější pro studiovou práci, extrém: GOP délky 1 (tj. všechny snímky jsou typu I)
- dlouhý GOP (typicky desítky až stovky snímků)
  - opačné vlastnosti než krátký GOP
  - typické pro koncovou distribuci
  - délka GOP se nesmí přehnat:
    poškození jednoho snímku může ovlivnit celý GOP
    přepínání mezi TV stanicemi trvá moc dlouho

  ⇒ jednou za čas je vhodné vynutit I snímek i na úkor sníženého kompresního poměru

DALŠÍ POSTUPY

- dynamická volba velikosti bloku: typicky 4×4 až 32×32 px
  - malý blok: přesnější motion compensation, lepší pro členitou část obrazu
  - velký blok: lepší komprese ploché části obrazu
  - bloky (jednotky DCT) vznikají dělením MB na poloviny nebo čtvrtiny
  - bloky mohou sdílet společný MV, nebo má každý blok svůj



- interpolované motion vektory (MV) – místo jednoho MV pro makroblok se uloží pro každý roh jeden
  $\Rightarrow$ každý pixel má vlastní interpolovaný MV
- motion compensation se může odkazovat na libovolný snímek v GOP

DALŠÍ POSTUPY

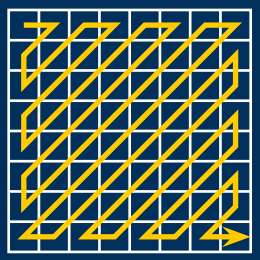
- in-loop deblocking
  - artefaktem DCT je viditelná struktura bloků v obrazu
  - při motion compensation se kodér odkazuje na ztrátově uloženou podobu snímku ⇒ DCT artefakty zvyšují koeficienty rozdílových makrobloků
  - DCT artefakty lze vizuálně „zahladit" – deblocking technika se stala velmi populární ⇒ proč ji nevyužít?

  ⇒ kodér předpokládá, že dekodér bude deblocking provádět (proto název „in-loop")
  - při motion compensation se kodér odkazuje na ztrátově uložený a při dekompresi vyhlazený obraz

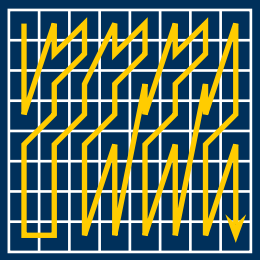
    ⇒ rozdílové makrobloky vycházejí lepší
- kompresní schéma nesmí klást velký odpor při hardwarové nebo paralelní implementaci

DALŠÍ POSTUPY

- alternativy k DCT s podobnými vlastnostmi, ale založenými na celočíselné aritmetice
- alternativní průchody 2D polem pro převod do 1D proudu – výběr podle toho, kde se očekává koncový shluk nul
- do výstupního proudu dat se ukládají kromě kvantizovaných DCT koeficientů také motion vektory, parametry kvantizace, typ snímku / makrobloku / bloku, typ dělení makrobloku, …
  ⇒ vše se pokud možno kóduje prediktivně (rozdílově)
  ⇒ vše se kóduje Huffmanovým nebo aritmetickým kodérem



průchod pro standardní obraz



průchod pro prokládaný obraz

DALŠÍ POSTUPY

ŠKÁLOVATELNOST

- požadavek: mít několik verzí videa různé kvality (rozlišení, ztrátovost, snímková frekvence, …)
  - pro internetovou distribuci prakticky nezbytnost: podle rychlosti stahování se automaticky mění kvalita
- naivní řešení: pro každou kvalitu jedna verze
  - nevýhodné: kvalitnější video musí obsahovat stejné informace jako nekvalitní plus nějaké navíc
- dobré řešení: základní video uloženo v nejhorší kvalitě, pro lepší kvalitu se do proudu dat doplní chybějící informace
- škálovatelnost je přirozená u kodeků založených na diskrétní waveletové transformaci

DALŠÍ POSTUPY

PROFILY / ÚROVNĚ

- požadavek: zařízení by mělo jednoznačně specifikovat, jaké video umí kódovat/dekódovat (zejména v reálném čase)
- požadavek: kompresní schéma by mělo být velmi obecné (nízké i vysoké rozlišení, datové toky, 2D/3D obraz, …)

⇒ nelze pouze tvrdit „zařízení X je umí zpracovat kompresní schéma Y“ – všechna zařízení by musela být výkonná nebo kompresní schéma jen jednoúčelové

- řešení u současných kompresních schémat:
  - profil: specifikuje, které vlastnosti schématu musí zařízení podporovat (např. 10bitová kvantizace, 3D obraz apod.)
  - úroveň: specifikuje parametry datového toku, které musí zařízení zvládnout (např. rozlišení 1920×1080 apod.)

DALŠÍ POSTUPY

PŘÍKLAD: MPEG-2 PROFILY (PROFILES)

- *Simple* – jen I, P snímky, chroma subsampling 4:2:0
- *Main* – simple + navíc B snímky
- *SNR* – main + navíc SNR škálovatelnost
  (tj. umí uložit video ve špatné kvalitě + data, která kvalitu zlepší)
- *Spatial* – SNR + navíc škálovatelnost v rozlišení
  (tj. umí uložit video v nízkém rozlišení + data, která rozlišení zlepší)
- *High* – spatial + navíc 4:2:2 chroma subsampling, 11bitové rozlišení DC koeficientu
- *Multiview* – main + více pohledů kamer (např. pro stereoskopii) + škálovatelnost ve snímkové frekvenci

DALŠÍ POSTUPY

PŘÍKLAD: MPEG-2 ÚROVNĚ (LEVELS)

Level	**High**	**High1440**	**Main**	**Low**
max. rozlišení	1920 × 1152	1440 × 1152	720 × 576	352 × 288
max. fps	60	60	30	30
max. px/s	$62{,}7 \cdot 10^6$	$47{,}0 \cdot 10^6$	$10{,}4 \cdot 10^6$	$3{,}0 \cdot 10^6$
max. bitrate	80 Mbps	60 Mbps	15 Mbps	4 Mbps

- zařízení pak např. deklaruje kompatibilitu s MPEG-2 Main profile @ High level

JEDNODUCHÝ KODÉR

ZÁKLADNÍ POSTUP

1. vstup: snímky X_0 X_1 X_2 X_3 …
2. rozhodnutí, které snímky budou I/P/B, změna pořadí snímků na Y_0 Y_1 Y_2 Y_3 …

příklad:

$Y_0 = X_0/I$ (prvním vstupem do kodéru bude snímek X_0, kódovat se bude jako I)

$Y_1 = X_3/P$ (pak zpracuje kodér snímek X_3 jako typ P)

$Y_2 = X_1/B$

$Y_3 = X_2/B$

…

JEDNODUCHÝ KODÉR

3a) má-li být snímek Y_i typu I:

- rozděl snímek na makrobloky
- proveď chroma subsampling
- rozděl výsledek na bloky
- na každý blok aplikuj DCT
- koeficienty DCT kvantizuj
- kvantizované koeficienty + parametry kvantizace odešli na výstup
- simuluj dekodér, dekódovaný snímek ulož do paměti M (= minulý)

JEDNODUCHÝ KODÉR

3b) má-li být snímek Y_i typu P:

- rozděl snímek na makrobloky + chroma subsampling
- pro každý makroblok se pokus najít ve snímku z paměti M nejpodobnější makroblok (motion compensation)
- podobný makroblok se nepovedlo najít: ulož makroblok stejně jako v I snímku
- podobný makroblok se povedlo najít: vypočítej rozdílový makroblok a ulož jej stejně jako v I snímku (a na výstup půjdou navíc motion vektory)

JEDNODUCHÝ KODÉR

- simuluj dekodér
- bude-li Y_{i+1} časově následovat po snímku Y_i, dekódovaný snímek ulož do paměti M
- bude-li snímek Y_{i+1} časově předcházet snímek Y_i, dekódovaný snímek ulož do paměti N (= následující; pokud není prázdná, přesuň napřed její obsah do M)

3c) má-li být snímek Y_i typu B:

- rozděl snímek na makrobloky + chroma subsampling
- pro každý makroblok se pokus najít ve snímku z pamětí M a N nejpodobnější makroblok (motion compensation)
- ulož makroblok podobně jako v P snímku (drobná změna: motion compensation je složitější)
- pokud na B snímcích jiný snímek nezávisí, máme hotovo

JEDNODUCHÝ KODÉR

4. zjisti aktuální datový tok (tj. počet bitů posledního snímku)
5. datový tok dobrý ⇒ nedělej nic
   datový tok špatný ⇒ uprav do dalšího snímku parametry kvantizace, aby se datový tok přiblížil požadavkům
6. pokračuj dalším snímkem

BĚŽNÁ KOMPRESNÍ SCHÉMATA

- MPEG-2 Video
  - typicky pro televizní vysílání, DVD, Blu-ray
  - relativně velké datové toky (SDTV kolem 3 Mbit/s, DVD kolem 5 Mbit/s ⇒ kompresní poměr cca 1 : 50)
  - relativně jednoduchý kodér/dekodér
- H.264 (= MPEG-4 Advanced Video Coding = AVC)
  - dá se používat prakticky na všechno: distribuční formát, videokonference, interní formát videokamer, …
  - Blu-ray, satelitní TV, internetové video, …
  - cca 2× menší datový tok než MPEG-2 při stejné kvalitě
  - výrazně komplikovanější kodér
- VC-1 (= SMPTE 421M, část Windows Media Video = WMV)
  - srovnatelné s H.264

BĚŽNÁ KOMPRESNÍ SCHÉMATA

- WebM
  - prosazující se formát fy Google založený na kompresním schématu VP-x (v současnotsi VP-9)
  - základní myšlenka: vyvinout algoritmy nezatížené patentovými problémy
  - srovnatelné vlastnosti jako H.264
- Apple ProRes, DNxHD (= SMPTE VC-3)
  - jednoduchá kompresní schémata, vysoký datový tok, vysoká kvalita obrazu, nízká výpočetní náročnost
  - pracovní formáty pro studiové zpracování
- H.265 (= MPEG-H = High Efficiency Video Coding = HEVC)
  - nastupující formát pro 4K rozlišení
  - očekávaný datový tok 2× menší než H.264

BĚŽNÁ KOMPRESNÍ SCHÉMATA

ROZLIŠUJEME

- kompresní schéma (standard): typicky popisuje jen syntaxi datového toku, neřeší implementaci
- kodek: implementace kodéru/dekodéru kompatibilního se standardem
  - kvalitní kodek = dobrá motion compensation, dobré řízení datového toku, dobrá rozhodnutí o typu kódování snímku apod. + rychlý běh kódování / dekódování
  - kvalitní kodek bývá podstatně efektivnější než jeho první (vývojová) verze

  ⇒ nově navrhovaná kompresní schémata musí nabídnout takové myšlenky, aby i první verze kodéru mohla soupeřit s vyladěnými kodéry starších standardů