VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÉ GRAFIKY A MULTIMÉDIÍ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER GRAPHICS AND MULTIMEDIA

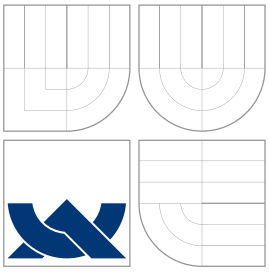
SROVNÁNÍ VIDEOKODEKŮ

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

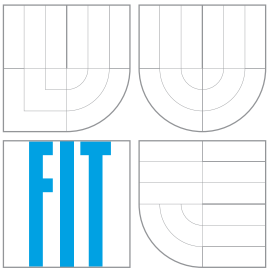
AUTOR PRÁCE
AUTHOR

PAVEL URBÁNEK

BRNO 2011



VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÉ GRAFIKY A MULTIMÉDIÍ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER GRAPHICS AND MULTIMEDIA

SROVNÁNÍ VIDEOKODEKŮ

COMPARISON OF VIDEO CODECS

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE — PAVEL URBÁNEK
AUTHOR

VEDOUCÍ PRÁCE — Ing. DAVID BAŘINA
SUPERVISOR

BRNO 2011

Abstrakt

Tato práce se zabývá analýzou videokodeků a jejich srovnáním. První část tohoto dokumentu poskytuje čtenáři základní informace o problematice kódování a dekódování videa, dále přibližuje jednotlivé kodeky a problematiku standardizace. Jsou představeny transformace jako DCT a DWT, dále intra a inter snímková predikce a entropická kódování. Druhá část je zaměřena na návrh testování a konkrétní srovnání včetně vyhodnocení. K samotném srovnání jsou použity metody PSNR, SSim a BD-PSNR.

Abstract

This thesis is aimed at analysis of video codecs and their comparison. First part of this document provides reader with the basic information on encoding and decoding process including high level description of often used algorithms. It describes codecs and the process of standardization. Second part is focused on test specifications and codecs comparsion it self including conclusion. PSNR, SSim and BD-PSNR are key methods used for comparing codecs.

Klíčová slova

Video, komprese, kodek, kodér, dekodér, DCT, DWT, kvantizace, ztrátový, bezeztrátový, intra, inter, pohybový vektor, MPEG-4 part 2, H.264, AVC, VC-1, VP8, Dirac, Theora, Schrödinger, H.265, PSNR, SSim, BD-PSNR

Keywords

Video, compression, codec, encoder, decoder, DCT, DWT, quantization, lossy, lossless, intra, inter, motion vector, MPEG-4 part 2, H.264, AVC, VC-1, VP8, Dirac, Theora, Schrödinger, H.265, PSNR, SSim, BD-PSNR

Citace

Pavel Urbánek: Srovnání videokodeků, bakalářská práce, Brno, FIT VUT v Brně, 2011

Srovnání videokodeků

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením pana Ing. Davida Bařiny. acknowledgment

.....
Pavel Urbánek
13. května 2011

Poděkování

Děkuji vedoucímu bakalářské práce Ing. Davidu Bařinovi za odbornou pomoc, rady při zpracování této bakalářské práce a doporučení soutěže EEICT.

Obsah

Kapitola 1

Úvod

Použití videokodeků je v současnosti neoddělitelnou součástí mnoha odvětví a stále se rozšiřuje. Sami se s videokodeky setkáváme, i když si to možná v některých případech neuvědomujeme, při různorodých činnostech, například sledování digitální televize, přehrávání DVD a Blu-Ray, přehrávání nejen filmů ale i online streamů na počítači, videokonferencích, v mobilních zařízeních od notebooků, netbooků a tabletů až po mnohé mobilní telefony. V každé z těchto kategorií jsou na kodeky kladeny různé požadavky vyplývající z architektury a použití těchto zařízení. Proto nelze obecně říct, který kodek je nejlepší, přesto existují vlastnosti, podle kterých lze kodeky srovnávat na širší úrovni. Především jde o kvalitu videa, úroveň komprese a rychlost kodeku. Tyto vlastnosti jdou ve většině případů proti sobě, musíme si tedy zvolit vhodný kompromis. Kromě toho je také velice důležité, a to především pro vývojáře produktů využívajících videokodeky, zda jsou použité kódovací algoritmy patentovány a jak je celý kodek lincencován.

Proč je vlastně potřeba video komprimovat? Na takovou otázku existuje hned několik odpovědí, nejpodstatnější je technické omezení přenosu nebo uchování dat. Někomu by se mohlo zdát, že v dnešní době, kdy jsou k dispozici disky s kapacitou TB a internetové připojení s rychlostmi v řádu desítek Megabitů, takže by možná nebylo potřeba video komprimovat. Stačí si však spočítat jednoduchý příklad. Uvažujme 10minutové video streamované na internetu v rozlišení 1080p s 25 snímky za sekundu a 8bitovými barvami. Jeden snímek bude vyžadovat $1920 \times 1080 \times 24$ bitů = 6.2 MB, jedna sekunda videa by vyžadovala 155 MB, 10 minut videa by bylo uloženo na 93GB. Jasným závěrem je, že komprese videa je nutností, bez které by se stěží někdo v dnešní době obešel.

Tato práce je rozdělena do logických celků odpovídajících postupnému seznámení čtenáře s problematikou. Tomuto textu by měl porozumět nebo alespoň získat přehled o problematice čtenář, který se doposud s tématikou videokodeků a jejich srovnání nezabýval. Po úvodní kapitole následuje celek věnovaný základním vlastnostem videa. Ve třetí kapitola jsou rozebírána jednotlivá kódování, transformace, předpovědi atd., čtvrtá kapitola je zaměřena na samotné videokodeky a to především problematice standardů a kodeků jimi definovaných. Pátá kapitola se zabývá návrhem, realizací a vyhodnocením srovnávacích testů, dále popisuje implementaci srovnávacího nástroje a způsoby kódování testovacích sekvencí. V závěrečné kapitole je shrnuto, co bylo předmětem této práce, jaké byly cíle a čeho bylo dosaženo.

Kapitola 2

Video

Pojmem video ve spojitosti s videokodeky lze interpretovat jako sekvenci snímků po sobě jdoucích v čase a vytvářejících tak dojem pohybu. Snímky chápeme jako matice jednotlivých bodů nesoucí informace o barvě daného bodu a jeho jasu (v případě černobílého obrazu je obsažena jen jasová složka). U videa lze identifikovat hned několik stěžejních vlastností jako jsou prokládání, počet snímků, rozlišení, barevné prostory/modely, datový tok a v neposlední řadě kvalita.

2.1 Prokládání

U videa se lze setkat s obrazem prokládaným (interlaced) a neprokládaným (progressive). Přirozeným způsobem zobrazení videa je neprokládaný obraz, jednotlivé snímky se postupně vždy celé zobrazí. Prokládané zobrazení [13] bylo zavedeno na základě problému omezení přenosového pásma, které bylo schopné přenést jen omezené množství snímků (například 25) za jednotku času. Na CRT obrazovkách však 25 snímků působilo blikání obrazu – jednotlivé body zhasínaly mnohem dříve, než byly opět rozsvíceny. Rozdělení obrazu na sudé a liché řádky tento problém vyřešilo. Jedná se o způsob „analogové“ komprese, kdy snížíme šířku pásma na polovinu při zachování plného rozlišení stacionárních scén a při rychlých scénách (například sportovním vysílání) je docíleno dostačující plynulosti na úkor snížení vertikálního rozlišení.

Zobrazování prokládaného obrazu má smysl jen u CRT televizních obrazovek. Počítačové LCD ale i CRT monitory pracují na jiném principu, kdy se zobrazují celé snímky a při vyšších frekvencích (typicky 60 – 120 Hz). Vzniká problém jak prokládaný obraz zobrazit korektně na takových zařízeních.

Existuje několik různě pokročilých metod. Tyto metody nazýváme odstranění prokládání (*deinterlacing*):

- Line doubling – zdvojnásobí každý řádek v půlsnímku, vytvoří tak celý snímek.
- Halfsizing – zobrazí každý půlsnímek tak jak je, vede k deformaci obrazu.
- Blending – složí dva půlsnímky „na sebe“ nevznikají tak zubaté přechody, ale duchové.
- Weaving – složí dva půlsnímky do jednoho, pokud je mezi nimi pohyb, vede k vytvoření zubatých přechodů.

2.2 Snímková frekvence

Snímková frekvence udává počet snímků zobrazených za jednotku času, jednotkou jsou Hz pro zobrazovací zařízení (televize, monitor), případně *fps* (frames per second). Nejčastěji se

snímkové frekvence 2.1 uvádějí ve formátu $fps\,[p \mid i]$, kde p znamená neprokládané zobrazení a i znamená prokládané zobrazení.

24p	typická snímková frekvence používaná ve filmu, lze ji převádět na PAL i NTSC, 24 fps vytváří dojem plynulého pohybu (též 23.976 fps v NTSC)
25p	odvozeno ze standardu PAL, který je zobrazuje 50 půlsnímků za sekundu (50i)
30p	alternativa pro 24p, poskytuje lepší vlastnosti při rychlém pohybu ve videu (například záznam sportu)
50p/60p	použito v moderních HDTV systémech
50i	PAL a SECAM standard, zobrazuje 50 půlsnímků, nebo 25 snímků
60i	respektive 59.94 fps (60×1000/1001) spadá pod NTSC standard

Tabulka 2.1: Běžné snímkové frekvence

2.3 Rozlišení

V rámci videa je chápáno jako počet sloupců × počet řádků, které zařízení zobrazí. Není to úplně korektní, jedná se spíše o rozměr obrazovky v bodech, přesnější představa rozlišení je množství bodů na určité ploše, nejčastěji se používá DPI/PPI (bodů na palec). Další možné pochopení je počet řádků obrazu 2.2, stále se používá, například 1080p představuje FullHD rozlišení 1920×1080 v neprokládaném módu. V tomto případě se vychází ze zavedeného poměru stran.

Standard	Označení	Rozlišení	Popis
SDTV	480i	243 řádků	NTSC, prokládané
	576i	288 řádků	PAL, prokládané
EDTV	480p	720×480	neprokládané
	576p	720×576	neprokládané
HDTV	720p	1280×720	HD ready, neprokládané
	1080i	1920×1080	540 řádků, prokládané
	1080p	1920×1080	FullHD, neprokládané

Tabulka 2.2: Běžná rozlišení

Právě poměr stran je důležitý parametr videa spojený s rozlišením. Jde o poměr šířky a výšky zobrazení. U počítačových monitorů se obvykle vyjadřuje jako poměr x:y (4:3, 16:9 atd.), zatímco v kinematografii jako reálné číslo v poměru k jedné (2.39:1). Spojené s poměrem stran je i *pixel aspect ratio* (PAR). Typickým představitelem nečtvercového PAR je televizní formát PAL, který při rozlišení 720×576 (5:4) je zobrazen na obrazovku s poměrem stran 4:3.

2.4 Kvalita videa

Kvalitu videa lze definovat jako míru podobnosti videa, které bylo zpracováno nějakým systémem a videa originálního. K porovnání dvou videí lze použít subjektivní nebo objektivní metody.

Subjektivní metody měření kvality jsou pochopitelně ne zcela přesné, jsou ovlivňovány mnoha faktory, jsou časově náročné, vyžadují lidské zdroje. Pro získání rozumných výsledků je potřeba provádět měření podle předem definovaných postupů, některé z nich jsou definovány v ITU-R/BT.500. Jednoduchá varianta DSIS může probíhat následovně: pozorovateli se pustí originální video, následně se mu pustí video po průchodu systémem, pak je požádán aby ohodnotil míru odlišnosti od „nepostřehnutelné“ po „velice obtěžující“, toto měření je vhodné opakovat s větším počtem pozorovatelů, doporučený počet je více než 20.

Objektivní metody [31] jsou založeny na kritériích a metrikách, které lze změřit objektivně. Jejich úspěšnost lze zjistit na základě porovnání výsledků objektivních testů s testy subjektivními. Mohou být rozděleny do kategorií podle dostupnosti originálního videa. Rozlišujeme:

- Full Reference Methods (FR)
  k dispozici je celé originální video, předpokládá se vysoká kvalita (typicky bezeztrátová nebo žádná komprese, porovnává se každý bod v originále s odpovídajícím bodem v upraveném videu
- Reduced Reference Methods (RR)
  k dispozici není celé originální video, porovnávají se například jen některé části, využívá se podobný princip jako u FR
- No-Reference Methods (NR)
  k dispozici není žádné originální video, lze použít v případě, že je znám použitý algoritmus komprese (kodek)

Častým způsobem zjištění kvality je výpočet odstupu signál/šum SNR (signal to noise ratio) a PSNR (peak signal to noise ratio), špičkového odstupu signál/šum [17]. Tento způsob ale neodpovídá naprosto přesně subjektivnímu vnímání kvality. Existují složitější, ale přesnější způsoby zjištění kvality, například UQI, VQM nebo SSIM.

Problematika zjištění efektivnosti zkoumaného kodeku v současné době vyžaduje opakované měření kvality na základě zakódovaného videa, což je časově náročné. Řešením je vývoj měření, které by bylo založené na odhadu výsledné kvality bez toho, aby se muselo provést samotné zakódování.

2.5 Datový tok

Datový tok je množství digitálních dat přenesené za jednotku času. Udává se v bps (bits per second – bitech za sekundu) a násobcích kbps, Mbps.

Datový tok lze klasifikovat do dvou kategorií, konstantní (CBR) a proměnný (VBR). Každý z nich má svá pozitiva a negativa. Konstantní datový tok se dobře uplatní při vysílání videa, kde je šířka pásma předem pevně dána. Naopak u videa, které je uloženo na lokálním médiu, a záleží především na poměru kvality a komprese, je výhodné použít proměnný datový tok, což umožňuje rozdělit dostupnou „kapacitu“ podle potřeby jednotlivých scén - složité scény si vyžádají obecně větší datový tok, zatímco statické, jednoduché scény využijí minimální datový tok. V implementacích některých kodeků se vyskytuje i *ABR – average bit rate*, což je varianta VBR, kdy je na začátku stanoven požadovaný průměrný datový tok a kodek se ho snaží dodržet, ale pro různé scény volí odpovídající datový tok.

Kapitola 3

Techniky kódování videa

Kódování videa je proces převodu videosekvence do formátu vhodného pro další zpracování, pro přenos, vysílání nebo ukládání. Na tuto formu jsou kladeny různé požadavky, nejdůležitějším cílem kódování videa je komprese. Té lze dosáhnout odstraněním nadbytečných – redundantních informací. Podle typu nadbytečných informací se ve videosekvencích rozlišuje prostorová redundance a časová redundance. Prostorová redundance je využívána a odstraňována při *intra snímkovém kódování*, časová redundance je eliminována při *inter snímkovém kódování*. Podstatnou součástí problematiky kódování videa jsou také barevné modely.

V této kapitole jsem čerpal ze záznamů lekcí NPTEL [1]. Především se jedná o části DCT a DWT.

Intra snímkové kódování

Videosekvenci lze reprezentovat jako posloupnost jednotlivých snímků. Nalezení a odstranění nadbytečných informací v jednom snímku je nazýváno intra snímkové kódování. Je to technika, která může být bezeztrátová – nedochází ke snížení kvality obrazu (původní snímek je totožný s dekódovaným). Lze rozlišit dva způsoby intra snímkového kódování, transformační kódování a prediktivní kódování.

Transformační kódování je obvyklé pro ztrátové kodeky, vstupní obraz je rozdělen na části – bloky, na ty je aplikována diskrétní kosinová transformace nebo diskrétní vlnková transformace. Výstupní koeficienty bývají kvantizovány (ztrátový element). Dále jsou data kódována do bitového toku pomocí entropických kódování.

Naopak prediktivní kódování je často používané u bezeztrátových kodeků a spočívá v předpovědi dat na základě jejich okolí a následné porovnání předpovědi se skutečnou hodnotou dat. Tento rozdíl je dále převeden do bitového toku entropickým kódováním.

Inter snímkové kódování

Ve videu se kromě prostorové redundance projevuje časová redundance, jejíž míra závisí na dynamičnosti scény (rychlost, množství a velikost pohybujících se objektů) a snímkové frekvenci. Videosekvence zachycující prvky reálného světa (nahrávky z kamery), dosahují značné podobnosti mezi jednotlivými po sobě jdoucími snímky. Odstraněním nadbytečné informace dojde ke značnému snížení celkového objemu dat. Nejjednodušším způsobem je výpočet rozdílu předchozího (referenčního) a současného snímku. Daleko lepšího výsledku však lze dosáhnout použitím techniky kompenzace pohybu.

Barevné modely

Uchování věrohodné barvy patří neodmyslitelně ke kódování videa. Asi nejznámější barevný model je RGB, jedná se o aditivní model, pokud se na počítači pracuje s barvou tak většinou bývá reprezentována pomocí RGB. Vyplývá to z jednoduchých operací nad tímto modelem a poměrně intuitivního pochopení.

Naopak pro video existují poměrně odlišné barevné modely, které vycházejí z historických souvislostí (přechod z černobílého zobrazení na barevné) a samy o sobě provádějí určitou kompresi dat. Využívá se zde poznatků o lidském zraku, z nichž vyplývá, že oko mnohem více reaguje na změnu intenzity světla než na změnu barvy. Informace o barvě bodu tedy nejdříve rozdělíme na složku jasovou a složky barevné. Barevné složky se ale vzorkují typicky s poloviční (nebo nižší) frekvencí. Existuje hned několik modelů, které popisují barvu tímto způsobem, většinou

Prostor	Poměr složek	Pořadí složek
YUV444	poměr složek Y, U, V je 4:4:4	Y1, U1, V1, Y2, U2, V2, ...
YUV422	poměr barevné složky k jasové je 1:2	Y1, U1, Y2, V1, ...
YUV420p a YV12	poměr barevné složky k jasové je 1:4	Y1, Y2, Y3, Y4, U1, V1, ...

Tabulka 3.1: Barevné modely

jsou velice podobné, liší se jen v detailech. Nejznámější jsou YUV 3.1, YPbPr nebo YCbCr.

3.1 Bezeztrátové techniky kódování

Stejně tak jako kodeky rozdělujeme na ztrátové a bezeztrátové, tak i techniky lze rozdělit těchto skupin. Je zde však důležité zmínit, že ztrátový kodek typicky kombinuje několik bezeztrátových technik, ale využívá i techniky ztrátové. Naopak bezeztrátový kodek nesmí obsahovat žádnou ztrátovou techniku.

3.1.1 Diskrétní kosinová transformace

Cílem transformace je dekorelace (odstranění závislostí v signálu) vstupního signálu. Z pohledu komprese videosignálu je žádoucí co nejvyšší stupeň dekorelace. Další důležitou vlastností transformací je koncentrace energie signálu. Obdobně i úroveň koncentrace energie ovlivňuje míru komprese. Diskrétní kosinová transformace [18] – DCT dosahuje vynikajících výsledků u obou zmíněných vlastností což je jeden z důvodů častého využití ve videokodecích.

Z hlediska aplikace si lze DCT 3.1 představit jako součet několika kosinusoid s odlišnými frekvencemi a amplitudami vytvářející původní signál. Od spojité kosinové transformace se liší v tom, že pracuje s konečným množstvím jednotlivých bodů.

$$S(k) = \sum_{n=0}^{N-1} s(n) \cos\left[\frac{\pi\,(2n+1)\,k}{2N}\right] \tag{3.1}$$

Obraz je však představován dvou dimenzionálním signálem. Je tedy nutné použít 2D transformace 3.2. Vstupní signál S transformujeme na výstupní signál s pomocí transformace t.

$$S(k_1, k_2) \xrightarrow{t} s(n_1, n_2) \tag{3.2}$$

Pro správnou rekonstrukci obrazu je v dekodéru použita inverzní transformace i 3.3, kde transformovaný s signál je převeden na původní signál S.

$$s(n_1, n_2) \xrightarrow{i} S(k_1, k_2) \tag{3.3}$$

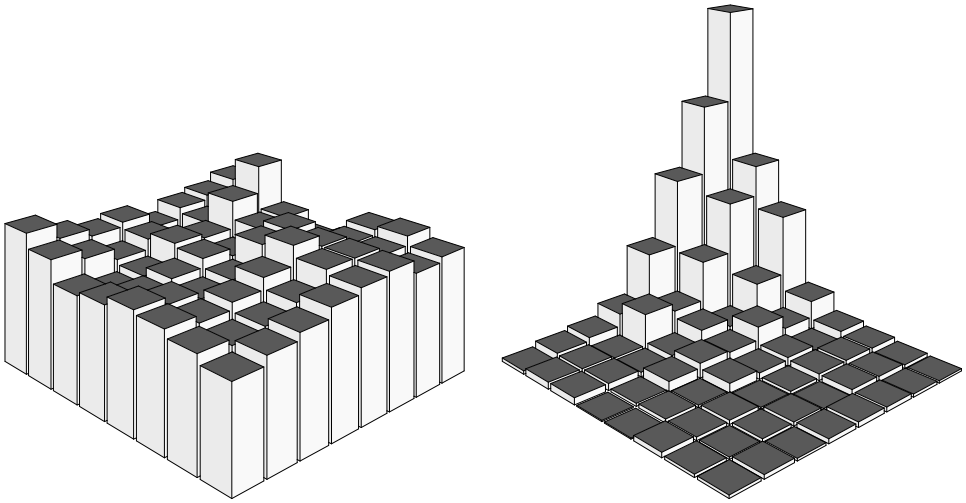
Obecná 2D transformace 3.4 může být zapsána pomocí systému g podobně jako obecná inverzní transformace 3.5 pomocí systému h.

$$S(k_1, k_2) = \sum_{n_1=0}^{N-1} \sum_{n_2=0}^{N-1} s(n_1, n_2) g\left(n_1, n_2, k_1, k_2\right) \tag{3.4}$$

$$s(n_1, n_2) = \sum_{k_1=0}^{N-1} \sum_{k_2=0}^{N-1} S(k_1, k_2) h\left(k_1, k_2, n_1, n_2\right) \tag{3.5}$$

Dosazením vhodného systému za g, h dostaneme odpovídající transformaci respektive její inverzní podobu.

Z uvedených rovnic je patrné, že pro vzorek dat $N \times N$ je počet kombinací N^4. Je to jeden z několika důvodů, proč je vstupní obraz nejprve rozdělen na čtvercové bloky. Například pro $N = 8$ je to 4096 výpočtů, požívají se proto optimalizované varianty DCT označované jako FCT – rychlá kosinová transformace pracujících na podobném principu jako rychlá fourierova transformace – FFT. Složitost je snížena z kvadratické na linearitmickou.



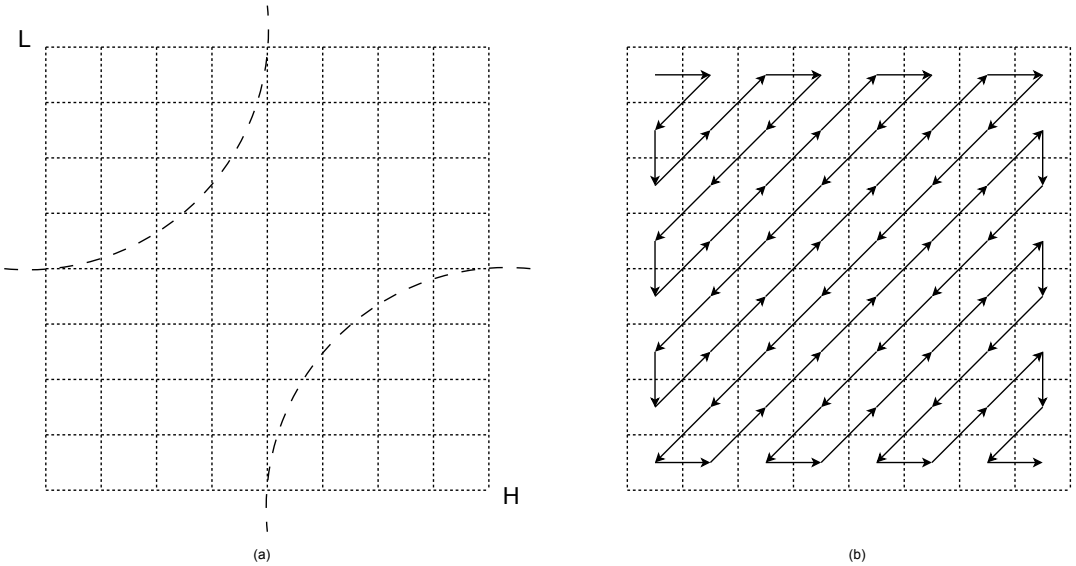
(a) (b)

Obrázek 3.1: (a) Hodnoty jasové složky ve vstupní matici 8×8 a (b) Výstup – koeficienty DCT 8×8.

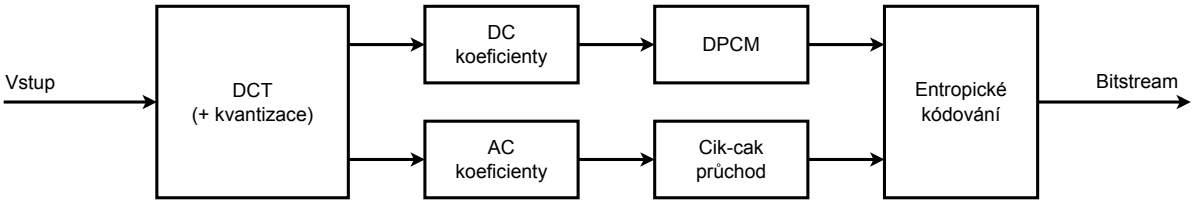
Obdobně jako na vstupu transformace je matice hodnot, tak i na výstupu očekáváme matici hodnot – koeficientů této transformace. Vstupní i výstupní matici lze reprezentovat grafem 3.1a. Z koeficientů DCT je patrný značný stupeň koncentrace energie v blízkosti bodu $[0, 0]$ společně s dekorelací, což vede ke snížení entropie v transformované části obrazu. To je základem pro efektivní entropické kódování takového signálu.

Entropické kódování zpracovává jednorozměrný signál, proto je nutné převést matici koeficientů do nějaké posloupnosti. Na základě znalosti rozložení energie (a) 3.2 je výhodné procházet jednotlivé koeficienty způsobem cik-cak (b) 3.2. Korelace mezi koeficienty získaná tímto způsobem bude také využita v entropickém kódování.

Dále bod $[0, 0]$ označujeme jako DC-koeficient a ostatní body jako AC-koeficienty. Vzhledem k značné odlišnosti vlastností AC a DC koeficientů může být použito DPCM kódování právě na DC koeficienty, zatímco AC koeficienty jsou procházeny cik-cak 3.3.



Obrázek 3.2: (a) Rozložení energie, kde L označuje oblast nízké frekvence – vysoké úrovně energie a H oblast vysoké frekvence – nízké úrovně energie, (b) cik-cak průchod maticí koeficientů využívající znalosti rozložení energie



Obrázek 3.3: Schéma zpracování koeficientů DCT

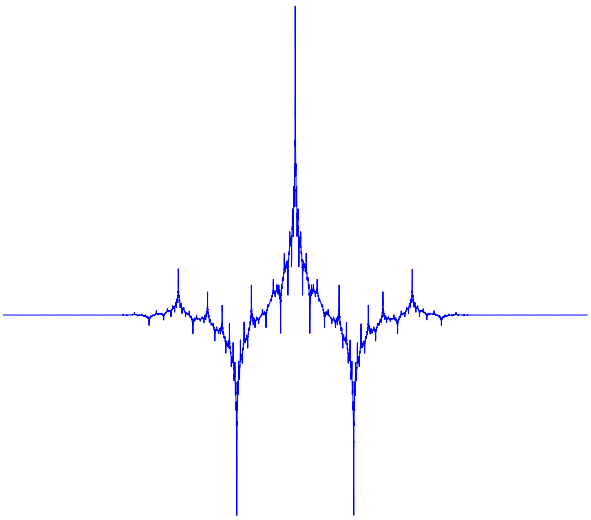
3.1.2 Diskrétní vlnková transformace

Diskrétní vlnková transformace – DWT [10] se zásadně liší od DCT tím, že nevytváří původní signál z kosinusoid ale z tzv. *vlnek* 3.4. Podobně jako u dalších transformací jde v souvislosti s kódováním videa především o koncentraci energie a dekorelaci, čehož dosahuje DWT podstatně lépe než DCT. Opět samotná transformace může být bezeztrátová (pokud jsou filtry vhodně zvoleny). Základním principem DWT je transformace vstupního signálu pomocí dvou filtrů. Tyto filtry se označují jako dolní propust a horní propust. Dolní propust h_φ vrací koeficienty signálu přibližné (vysoká energie, podstatná informace) zatímco horní propust h_ψ vrací koeficienty detailní (nízká energie, méně podstatná informace). Tyto filtry (společně s jejich rekonstrukčnímy opaky g_φ a g_ψ) musí splňovat několik podmínek a to komplementárnost 3.7 a perfektní rekonstrukci 3.6. Pokud splňují obě tyto podmínky tak je možné výstupy podvzorkovat aniž by byl porušen Shannonův teorém (na vstupu je 2^n a na výstupu opět 2^n ale pro oba filtry).

$$h_\varphi(n)\, g_\varphi\left(n^{-1}\right) + h_\psi(n)\, g_\psi\left(n^{-1}\right) = 2 \tag{3.6}$$

$$h_\varphi(n)\, g_\varphi\left(-n^{-1}\right) + h_\psi(n)\, g_\psi\left(-n^{-1}\right) = 0 \tag{3.7}$$

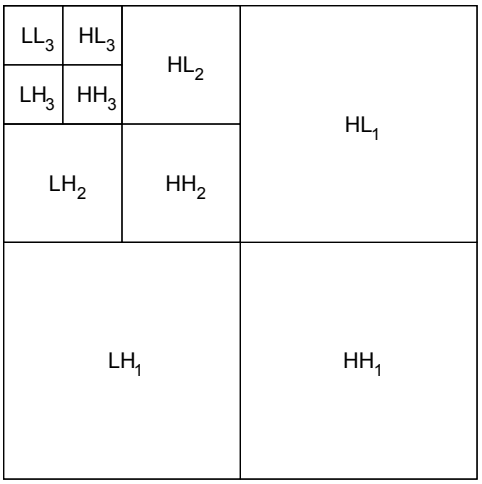
Dolní propust h_φ využívá $\varphi(n)$, které označujeme jako škálovací funkci, horní propust h_ψ využívá $\psi(n)$, které označujeme jako vlkovou funkci. Pro transformaci obrazu se používají



Obrázek 3.4: Příklad vlnky CDF 5/3

nejčastěji CDF 5/3 (reverzibilní, bezeztrátové) a CDF 9/7 (ireverzibilní, ztrátové).

Transformace se opakuje nad výstupem dolní propusti a tím se vytváří banka filtrů. Pro vstupní signál s počtem vzorků 2^n je počet opakování až n.



Obrázek 3.5: Dělení na čtyři subpásma s opakování pro LL

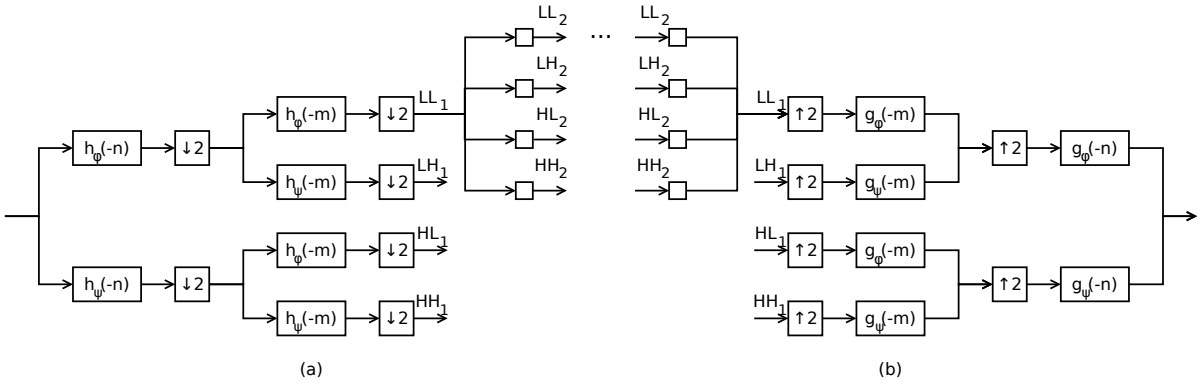
Zpracování obrazu ale vyžaduje dvourozměrnou transformaci. Vstupní 2D signál je dělen do čtyř podpásem 3.5 na rozdíl od transformace jednorozměrného signálu, kde došlo k dělení na dvě podpásma. Jsou proto zavedeny čtyři filtry 3.8, které však nejsou nic jiného než kombinace dvou původních filtrů dolní a horní propusti.

$$\begin{aligned} \varphi(n_1,n_2) &= \varphi(n_1)\,\varphi(n_2) \;\; \text{LL} \\ \psi^H(n_1,n_2) &= \psi(n_1)\,\varphi(n_2) \;\; \text{LH} \\ \psi^V(n_1,n_2) &= \varphi(n_1)\,\psi(n_2) \;\; \text{HL} \\ \psi^D(n_1,n_2) &= \psi(n_1)\,\psi(n_2) \;\; \text{HH} \end{aligned} \tag{3.8}$$

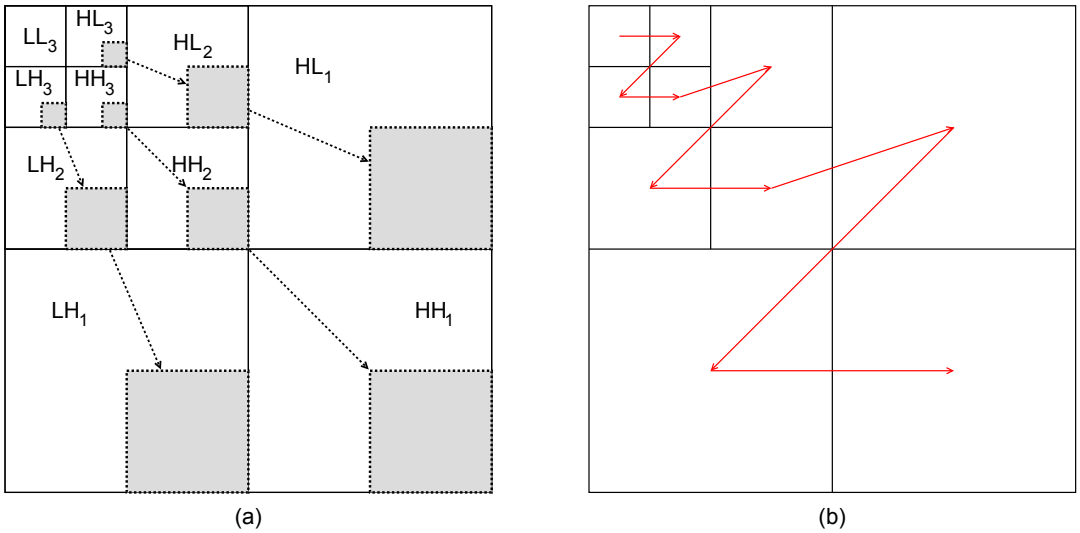
Každý filtr reprezentuje jedno podpásmo: LL, LH, HL, HH. Obdobně jako u transformace 1D signálu i zde dochází k opakování transformace a to na podpásmu LL (nejvyšší energie),

dělení může probíhat až na úroveň jednotlivých bodů obrazu, většinou se provádí pouze několik kroků dělení (a) 3.6.

Jak již bylo zmíněno, správně zvolené filtry umožňují perfektní rekonstrukci původního signálu. Nejde o nic jiného, než o postupné slučování podpásem zpracovaných rekonstrukčními filtry (b) 3.6. Stejně jak bylo provedeno podvzorkování v rámci dělení, tak je zde potřeba provádět operaci „nadvzorkování“. Operace dělení označujeme jako analýzu, operace slučování jako syntézu.



Obrázek 3.6: DWT transformace obrazu znázorňující dělení na podpásma, (a) analýza, (b) syntéza)



Obrázek 3.7: (a) Vztahy mezi oblastmi pásem, (b) průchod zpracování koeficientů u EZW

Jak již bylo zmíněno DWT není sama o sobě ztrátová, podobně jako u DCT je pro dosažení větší komprese zavedena kvantizace, tak i u DWT existují podobné techniky. Vychází se v nich z faktu, že největší část energie je v LL části a snižuje se směrem k HH podpásmu. Dále je také důležitým faktorem to, že existuje určitá forma závislosti 3.7 (a) mezi odpovídajícími oblastmi v různých úrovní analýzy. Technika, která využívá těchto vlastností DWT se nazývá EZW (Embedded Zerotree Wavelet) [30] 3.7 (b). Existují pokročilejší techniky jako je EBCOT (Embedded Block Coding with Optimal Truncation) [29] použitý v JPEG2000.

3.1.3 Kompenzace pohybu

Kompenzace pohybu je technika používaná v kódování videa k dosažení vyšší úrovně komprese díky přítomnosti časové redundance. Jedná se o inter snímkovou techniku, která spočívá v popisování oblastí v jednom snímku pomocí oblastí z okolních snímků v čase. Odkazované snímky mohou být jak předcházející tak následující.

Rozlišejeme několik způsobů kompenzace pohybu:

- Globální kompenzace pohybu (GMC) – definován pohyb celého obrazu, pohyb ve 3D včetně rotací, reflektuje pohyb kamery
- Bloková kompenzace pohybu (BMC) – posunutí jednotlivých bloků definované vektorem pohybu
- Kompenzace pohybu s proměnnou velikostí bloku – kodér má možnost volit mezi několika velikostmi bloku, vylepšení BMC
- Kompenzace pohybu s přesahujícími bloky (OBMC) – velikost bloku je taková, že pokrývá všech 8 sousedních bloků, vede to k lepším výsledkům, ale podstatně vyšší časové složitosti

Nejčastěji implementovaná kompenzace pohybu je BMC, případně její vylepšené verze. Je zaveden prvek nazvaný pohybový vektor, který definuje posunutí bloku. Vypočítán je pomocí techniky odhadu pohybu. Protože se v obraze mohou pohybovat objekty větší než jeden blok a některé sousední bloky budou mít stejné vektory pohybu, je vektor kódován jako rozdíl od předchozích, což vede ke snížení velikosti. Následně je společně s obrazovými daty entropicky zakódován.

Odhad pohybu je proces ve kterém je nalezena odpovídající pozice bloku v daném okolí (prohledávat celý obraz nemá smysl, protože by to bylo příliš výpočetně náročné).

Existuje několik metod prohledávání tohoto okolí:

- MSE (Mean squared error) – hledá se nejmenší rozdíl mezi bloky a záporné hodnoty jsou odstraněny umocněním, což ale není výpočetně výhodné.
- MAD (Mean absolute difference) – upravená verze MSE, umocnění je nahrazeno absolutní hodnotou.
- MPC (maximum pixel count) – hledá se maximální počet stejných hodnot jednotlivých bodů v bloku.
- Jain and Jain, Cross search, Three step search, Diadomd search – metody, označované jako *suboptimální* – nemusí dospět k nejlepšímu řešení, ale jsou mnohonásobně rychlejší, složitost je logaritmická.

Pohyb jako takový lze reprezentovat reálnými čísly, zatímco obraz je mapován celočíselně. Pro dosažení lepších výsledků byla zvýšena přesnost kompenzace pohybu pod úroveň jednoho pixelu, běžně na jednu polovinu až jednu čtvrtinu. Protože obraz obsahuje body pouze na celých souřadnicích je nutné zbývající body vypočítat 3.8, typicky interpolací nebo použitím filtrů. Následně je možný odhad pohybu i na pod-pixelové úrovni.

Snímky

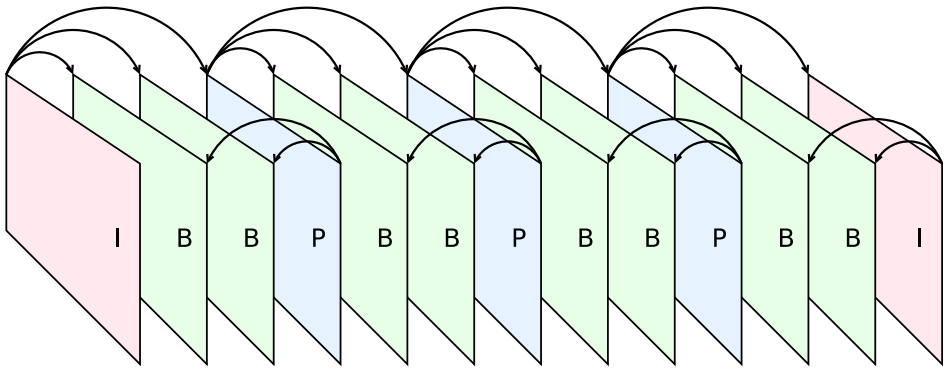
Pro kompenzaci pohybu je vztah mezi snímky rozhodující. Tyto vztahy lze definovat rozdělením snímků do skupin.

Typy snímků:

Obrázek 3.8: Interpolace bloku 4x4 pro přesnost kompenzace pohybu 1/2 pixelu a 1/4 pixelu

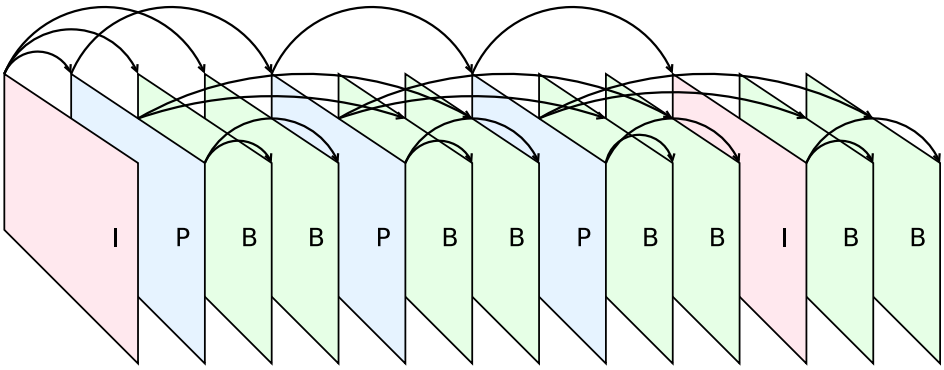
- I snímky – taktéž klíčové snímky, jsou intra kódovány, nejsou tudíž závislé na ostatních snímcích a k jejich dekódování tudíž nejsou potřeba žádné další snímky.
- P snímky (dopředná predikce) – jsou inter kódované snímky, obsahují reference na předchozí P nebo I snímky, značná redukce velikosti proti I snímkům.
- B snímky (oboustranná predikce) – jsou inter kódované snímky, obsahují reference na předchozí a následující snímky, dosaženo nejmenší velikosti.
- Zlaté (golden frame) a alternativní referenční (altref frame) [2] – tyto snímky jsou v samotném videu neviditelné (označeny při kódování), mohou být vytvořeny z několika jiných snímků. Přenáší se jako I nebo P snímky. Obsahují všechny změny od předchozího I snímku, využívá se jich pro dekódování ostatních P snímků. Jedná se částečně o náhradu B snímků.
- SI snímky – umožňují přepínání mezi streamy, přetáčení, synchronizaci mezi více dekodéry. Skládají se z SI makrobloků (intra kódovaný makroblok).
- SP snímky – mají obdobné vlastnosti jako SI snímky, obsahují však P a/nebo I makrobloky.

Skupina snímků



Obrázek 3.9: Skupina snímků – GOP u MPEG

Skupina snímků (GOP – group of pictures) představuje posloupnost jednotlivých snímků, kde všechny snímky jsou dekódovatelné na základě informací z dané skupiny snímků – veškeré reference jsou v rámci jedné GOP. Skupina snímků může obsahovat snímky typu I, P a B, případně další speciální (SI, SP atd.). Je definována vzdálenost dvou referenčních snímků P (a I) M a vzdálenost intra kódovaných snímků I označovaná jako N. Příkladem GOP s $M = 3$ a $N = 12$ 3.9 je MPEG. Snímky typu B využívají oboustranné predikce, což znamená, že pro jejich zakódování je potřeba jak předchozí tak následující referenční snímek. Proto je nutné



Obrázek 3.10: GOP s upraveným pořadím snímků

upravit pořadí snímků tak, aby se snímky odkazovaly jen na předcházející viz obrázek 3.10. Při dekódování je původní pořadí obnoveno.

Makrobloky, řezy a skupiny řezů

Některé kodeky zavádí dělení obrazu na makrobloky, typicky konstantních rozměrů, které obsahují 16×16 vzorků jasové složky a dvě pole 8×8 pro odpovídající složky barvy. Jak složka jasu tak složky barvy jsou buď časově nebo prostorově předpovídány, a rozdíl proti kódovanému snímku je dále zpracováván. Každá složka je rozdělena na bloky, které jsou následně transformovány. Koeficienty transformace jsou následně kvantizovány a výstup je entropicky zakódován.

Makrobloky jsou uspořádány do logických celků nazvaných řezy [33] . Řezy představují oblasti snímku, které mohou být dekódovány nezávisle na ostatních. Každý řez je posloupností makrobloků, jež jsou zpracovány v pořadí rasterizačního průchodu tzn. shora zleva, dolů doprava (neplatí to však vždy, viz FMO). Jeden snímek může obsahovat jeden nebo více řezů, každý řez je úplný, což znamená, že se znalostí aktivní sekvence snímků a parametrů obrazu může být dekódován bez nutnosti čtení dat z ostatních řezů v daném snímku. Řezy jsou používány především pro odolnost vůči chybám a paralelní zpracování.

3.2 Ztrátové techniky kódování

Pakliže není bezeztrátová komprese dostačující, je nutno zavést techniku, která odstraní nejméně potřebnou informaci z dat, s cílem snížení objemu s minimálním dopadem na kvalitu. Tuto techniku označujeme jako *kvantizaci*. Kvantizace se typicky aplikuje na výstup transformace.

Kvantizace

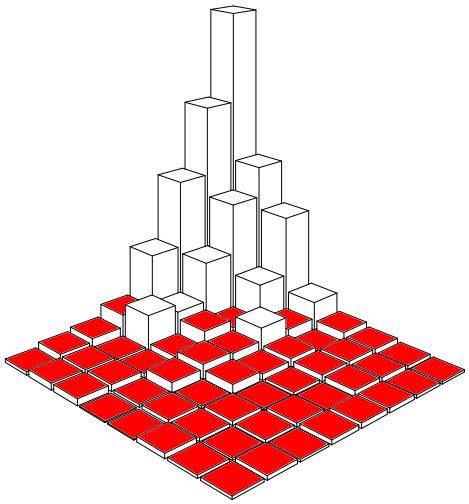
Kvantizace (syn. kvantování) je na rozdíl od od ostatních technik kódování ztrátová operace. Vychází se zde ze znalosti vlastností lidského vidění, kdy je možné určitou část obrazové informace odstranit aniž by došlo k zásadnímu poklesu vnímané kvality obrazu. Lidské oko reaguje především na oblasti s vysokou energií, zatímco oblasti nízké energie jsou méně podstatné.

Existuje více způsobů kvantizace, například zónové kódování nebo prahové kódování. Zónové kódování spočívá ve výpočtu masky 3.11a na základě informace o požadované velikosti výstupního signálu. Jedná se o binární masku, definující které koeficienty budou zachovány a které odstraněny – nahrazeny 0. Příklad této metody je znázorněn na obrázku 3.11b.

Prahové kódování naopak využívá kvantizační matice pro výpočet nových koeficientů. Kvantizační matice obsahuje hodnoty, které byly určeny na základě rozsáhlého studia lidského

1	1	1	1	0	0	0	0
1	1	1	0	0	0	0	0
1	1	0	1	0	0	0	0
1	1	0	0	0	0	0	0
0	1	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

(a)



(b)

Obrázek 3.11: (a) Kvantizační maska a (b) Koeficienty odstraněné kvantizací.

zraku. Protože se nejedná o exaktně určené hodnoty, je ve většině kodeků implementovaná celá sada kvantizačních matic, v některých případech je dovoleno definovat vlastní hodnoty. Příklad matice koeficientů DCT 3.9, kvantizační matice 3.10 a kvantizovaného výstupu 3.12.

$$S\left(k_1,k_2\right)=\begin{bmatrix} -415 & -33 & -58 & 35 & 58 & -51 & -15 & -12 \\ 5 & -34 & 49 & 18 & 27 & 1 & -5 & 3 \\ -46 & 14 & 80 & -35 & -50 & 19 & 7 & -18 \\ -53 & 21 & 34 & -20 & 2 & 34 & 36 & 12 \\ 9 & -2 & 9 & -5 & -32 & -15 & 45 & 37 \\ -8 & 15 & -16 & -7 & -8 & 11 & 4 & 7 \\ 19 & -28 & -2 & -26 & -2 & 7 & 44 & -21 \\ 18 & 25 & -12 & -44 & 35 & 48 & -37 & -3 \end{bmatrix} \tag{3.9}$$

$$T\left(k_1,k_2\right)=\begin{bmatrix} 16 & 11 & 10 & 16 & 24 & 40 & 51 & 61 \\ 12 & 12 & 14 & 19 & 26 & 58 & 60 & 55 \\ 14 & 13 & 16 & 24 & 40 & 57 & 69 & 56 \\ 14 & 17 & 22 & 29 & 51 & 87 & 80 & 62 \\ 18 & 22 & 37 & 56 & 68 & 109 & 103 & 77 \\ 24 & 35 & 55 & 64 & 81 & 104 & 113 & 92 \\ 49 & 64 & 78 & 87 & 103 & 121 & 120 & 101 \\ 72 & 92 & 95 & 98 & 112 & 100 & 103 & 99 \end{bmatrix} \tag{3.10}$$

DCT koeficienty S jsou vyděleny kvantizační maticí T a zaokrouhleny na celé číslo 3.11. Změnou faktoru M lze dosáhnout různé úrovně kvantizace.

$$\dot{S}\left(k_1,k_2\right)\doteq\left[\frac{S\left(k_1,k_2\right)}{T\left(k_1,k_2\right)M}\right] \tag{3.11}$$

$$\dot{S}(k_1, k_2) = \begin{bmatrix} -26 & -3 & -6 & 2 & 2 & -1 & 0 & 0 \\ 0 & -3 & 4 & 1 & 1 & 0 & 0 & 0 \\ -3 & 1 & 5 & -1 & -1 & 0 & 0 & 0 \\ -4 & 1 & 2 & -1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \tag{3.12}$$

3.3 Entropická redundance

Po transformaci (s případnou kvantizací) a cik-cak čtení koeficientů je vstupní obraz převeden do jednorozměrné posloupnosti číselných hodnot. Tyto hodnoty dosahují značné úrovně entropické redundance, proto jsou dále zpracovávány. Proces odstranění redundance se označuje jako *entropické kódování*. To lze rozdělit na dvě hlavní činnosti a to modelování a kódování. Cílem prvního je efektivně přiřadit vstupním znakům pravděpodobnosti, zatímco samotné kódování provádí převod vstupních znaků na výstupní sekvence bitů odpovídající délky. Tato délka jde určit pomocí Shannonova teorému, který popisuje vztah mezi pravděpodobností výskytu znaku a výstupní posloupností bitů. Pro vstupní znak s pravděpodobností P a kde b (v případě binárního výstupu je $b = 2$) je počet znaků, kterými bude zakódován výstup, je optimální délka posloupnosti dána $-\log_b P$. Pochopitelně efektivita výsledného kódování se odvíjí od správně určených pravděpodobností, proto modelování je nejpodstatnější částí celého procesu. Nejčastěji používané entropické kódování je Huffmanovo a aritmetické kódování.

Koncept entropického kódování je často doprovázen RLE – run-lenght encoding, což je technika redukce dlouhých běhů stejné hodnoty. Jedno výstupní kódové slovo reprezentuje dvě informace ze vstupu a to délku běhu a daný znak.

Huffmanovo kódování

Bezeztrátová komprese dat založena na analýze dat a mapování nejčastějších vzorů (znaků) na krátké kódy, zatímco málo časté vzory jsou kódovány na delší bitové řetězce.

Vyžaduje dva průchody, v prvním se provede analýza, v druhém samotné zakódování na základě binárního stromu vytvořeném po prvním průchodu. Neznámější použití Huffmanova kódování je kodek HuffYUV.

Aritmetické kódování

Jedná se o algoritmus pro bezeztrátovou komprimaci dat [24]. Na rozdíl od Huffmanova kódování, kde dochází k nahrazování vzorů ve vstupních datech kódy různé délky na základě četnosti, aritmetické kódování pracuje s celým vstupním řetězcem, který převede na zlomek v rozsahu (0.0, 1.0). Je dosaženo lepší komprese než v případě Huffmanova kódování.

Adaptivní kódování

Protože vstupní data entropického kódování bývají často poměrně složitá, je důležité nalézt odpovídající modely těchto dat. Adaptivní kódování spočívá v odhadu pravděpodobnosti vstupních znaků, které jsou následně zakódovány například aritmetickým kódováním.

CABAC a CAVLC

CABAC (Context-adaptive binary arithmetic coding) a CAVLC (Context-adaptive variable length coding) jsou entropická kódování používaná v H.264/AVC [23]. Obě tyto techniky jsou bezeztrátové. CABAC patří k nejlepším entropickým kódováním, nicméně je velmi náročný. Součástí je definovaná množina modelů pro různé kontexty. Vstupní data jsou převedena do binární podoby, každý bit je kódován pomocí odpovídajícího modelu, přičemž využívá hodnot ze svého okolí pro zlepšení přesnosti modelu. Na výsledek je aplikováno aritmetické kódování. CAVLC je alternativou k CABAC, nedosahuje takové úrovně komprese, ale zase není zdaleka tak náročný.

3.4 Post-processing

Post-processing patří mezi důležité techniky zpracování obrazu [25]. Hlavním cílem je zlepšit vizuální kvalitu videa po jeho dekódování (používá se ale i v jiných oblastech). Ve většině případů je součástí přehrávače, jak HW tak SW. Post-processing nastupuje po fázi dekódování videa, je nezávislý na použitém kodeku. Vyžaduje nezanedbatelný výkon ze strany přehrávacího zařízení. Zlepšuje obraz především při změnách velikosti, kdy je potřeba provést buď podvzorkování nebo vhodnou interpolaci, nedochází tak ke vzniku nežádoucích artefaktů v obraze. Další funkcí je také odstraňování efektů „kostkatění“ vzniklých kompresí.

Existuje také techniky *pre-processing*, které se provádějí před kódováním, jejich cílem je např. odstranění šumu.

Kapitola 4

Videokodeky

4.1 Standardizace

Jedná se o proces, jehož cílem je formulace vlastností, chování, architektury atd. určitého systému. Výstupem může být text, grafy, schémata, tabulky atd., z nichž lze jednoznačně vyvodit funkčnost systému a následně jej nebo jeho část realizovat ať již implementací v software nebo hardware. Cílem standardizace [26] je především kompatibilita mezi jednotlivými implementacemi, dále dostupnost tohoto řešení pro kohokoli, kdo se rozhodne implementovat daný standard nebo jeho část. Standardizace je nejdůležitější pro implementaci v HW, kde pozdější úpravy nebo opravy výrobku jsou téměř nemožné a velice nákladné.

Autority na poli standardizace kodeků:

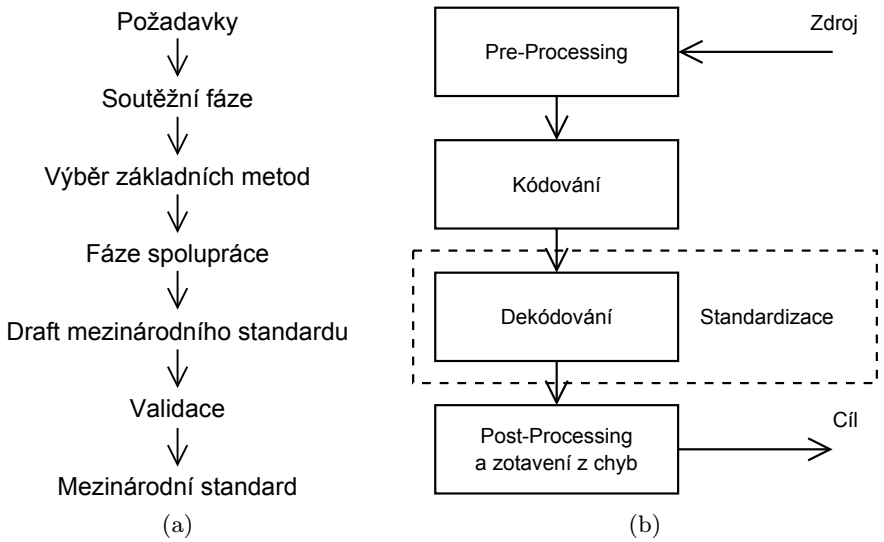
- ITU-T – VCEG video coding experts group
- ISO/IEC – MPEG motion picture experts group, JPEG joint photographics experts group

Samotný proces standardizace 4.1a se může lišit v rámci jednotlivých organizací, základní kroky jsou přibližně stejné. První fáze spočívá v ustanovení požadavků vyplývajících z konkrétního nasazení nebo oblasti nasazení, druhá fáze probíhá na jednotlivých pracovištích účastnících se vývoje, kde každé pracoviště vyvíjí vlastní řešení – algoritmus, tato řešení jsou následně porovnána a je vybráno jedno konkrétní řešení, což odpovídá třetí fázi. Toto řešení je dále rozvíjeno v rámci společného úsilí zúčastněných pracovišť. V páté fázi je vydán *draft* – návrh standardu, který je následně validován na základě testování, výsledků z reálného nasazení a porovnávání s požadavky. Výsledkem úspěšné validace je vydání mezinárodního standardu.

Cíl standardizace je naznačen na obrázku 4.1b.Standardy ISO/IEC a ITU-T jsou realizovány ve všech případech obdobným způsobem, kdy standardizaci podléhá pouze hlavní dekodér a jsou kladena určitá omezení na vstupní signál (bitstream) a syntaxi dekodéru. Vede to k tomu, že každý dekodér odpovídající standardu by měl pro jeden konkrétní vstupní signál poskytnout stejný nebo velice podobný výstup. Tento styl standardizace směřuje k maximální volnosti při optimalizaci implementací kodéru pro konkrétní cílové použití při zachování široké kompatibility v rámci dekódování.

4.1.1 Problematika analýzy kodeku na základě standardů

Jak již vyplývá z předchozího seznámení se standardizací, součástí standardu je typicky pouze popis dekodéru a výstupního bitového toku. V tomto ohledu může být poměrně složité zjistit jaké algoritmy jsou v daném kodeku použity, pokud to není explicitně uvedeno ve standardu. Dále jsou některé standardy proprietární, tudíž přístup k nim je omezený. V neposlední řadě se lze setkat s kodeky, které standard ani nemají. Z těchto poznatků vyplývá nutnost použití dalších zdrojů informací.



Obrázek 4.1: (a) Proces standardizace a (b) cíl standardizace.

4.1.2 Profily a úrovně

Profily a úrovně představují „záchytné body“, jsou zaváděny z důvodu zachování vzájemné kompatibility aplikací, které využívají obdobných funkcí kodeku. Profil definuje množinu nástrojů a algoritmů, které slouží k vytvoření výstupního bitového toku. Úroveň určuje podmínky a omezení klíčových parametrů videa a bitového toku.

Všechny dekodéry vztahující k danému profilu musí podporovat všechny funkce toho profilu na rozdíl od kodérů, které nemusí implementovat žádné z daných funkcí profilu, ale musí generovat odpovídající bitový tok.

4.2 Standardy

4.2.1 MPEG-4 Part 2

MPEG-4 představuje standard ISO/IEC 14496 vyvíjený skupinou MPEG. Jedná se o množinu metod zpracování multimediálního obsahu, především videa a audia. Vychází z předchozích standardů MPEG-1 a MPEG-2 nebo VRML. Sestává z mnoha částí, které se zaměřují na jednotlivé podproblémy zpracování/kódování multimediálního obsahu [5].

Významné části standardu:

- Part 2: Visual
- Part 3: Audio
- Part 10: Advanced video coding
- Part 14: MP4 file format
- Part 15: Advanced Video Coding (AVC) file format

MPEG-4 Part 2 je standardem ISO/IEC 14496-2 [3], popisujícím částečně kódování, ale především dekódování videa. Standard definuje nástroje pro zpracování, uložení a přenos dat reprezentujících textury, obrazy a videa pro různá multimediální prostředí. Tyto nástroje umožňují dekódování atomických prvků označovaných jako Video Object (VO). Aby bylo dosaženo vyšší použitelnosti, jsou jednotlivé nástroje sloučeny do skupin podle společného nasazení. Jedná se

především o skupiny nástrojů pro kompresi obrazu a videa, pro kompresi a mapování textur na 2D a 3D objekty, kódování obrazu a videa na základě jeho obsahu, škálovatelnost prostorovou, časovou i kvalitativní a odolnost vůči chybám či ztrátám.

Základní schéma kódování zahrnuje jak klasický přístup (DCT kódování bloků 8×8), tak tvarové kódování (shape coding) – pro libovolně tvarované VO (transformace pomocí shape-adaptive DCT). Výhodou je zvýšení efektivity komprese v případech, kdy lze některé objekty ve scéně detekovat a použít odpovídající techniku objektové kompenzace pohybu. Mezi další techniky inter snímkového kódování patří u MPEG-4 Part 2 standardní 8×8 bloková kompenzace pohybu s přesností až na 1/4 bodu ataké *sprite coding* – efektivní kódování statických objektů nebo pozadí.

Jedním z hlavních cílů MPEG-4 Part 2 bylo umožnit kódování videa na úrovni objektů ve scéně. Výhody tohoto přístupu jsou nesporné, kromě lepších možností uložení takovýchto objektů (formou stromového grafu a s tím spojené snížení velikosti dat), přináší také jistou míru interakce, například možnost změny barvy auta ve filmu nebo označení hráče ve sportovním přenosu. Přestože je objektový přístup k videu revoluční a odemyká široké spektrum možností, nebylo dosaženo úspěšného praktického nasazení této technologie. Vyplývá to z několika komplikací spojených s kódováním a dekódováním. Prvním a nejzásadnějším problémem je analýza scény, identifikace a rozdělení objektů v ní. Tato činnost je algoritmicky extrémně složitá, v některých případech až nemožná (v porovnání s dosavadními technikami kódování videa pomocí transformací atd.). Podobný problém vzniká na straně dekodéru, kde k manipulaci s objekty ve scéně nebo obecně jakoukoli interakcí dochází k dramatickému nárůstu požadavků na vybavení jak hardwarové tak softwarové. Jako důsledek zmíněných a mnoha dalších komplikací spojených s objektově orientovaným přístupem k videu, nebyly tyto funkce nasezeny v širším měřítku.Část funkcí MPEG-4 Part 2 nicméně našla uplatnění, jedná se především o množinu funkcí Simple Profile (SP) a Advanced Simple Profile (ASP), jejichž rozšířenost ve formě kodeků DivX, Xvid, libavcodec nebo QuickTime je nepopiratelná.

4.2.2 H.264/MPEG-4 AVC

Za vývojem tohoto standardu [14], [4] stojí společné úsilí expertů ze skupin MPEG a VCEG formující JVT - joint video team. Jako základní požadavky bylo stanoveno následující:

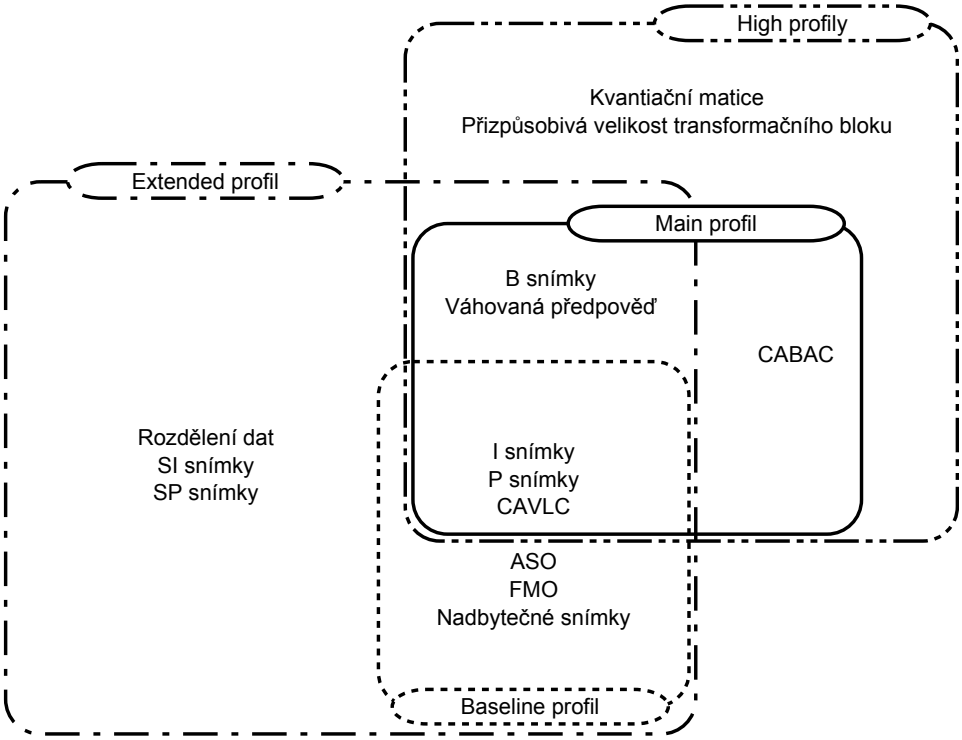
- Satelitní, pozemní a kabelové vysílání, přenos přes DSL atd.
- Ukládání na magnetická a optická média, např. DVD, Blu-Ray atd.
- Videokomunikační služby dostupné přes DSL, LAN, Ethernet, mobilní sítě atd.
- Ukládání záznamů a streamování videa

Pro splnění těchto požadavků bylo nutné zdokonalit samotné kódování, zobecnit a zjednodušit specifikace formátu dat a docílit perfektní integrace a kompatibility. Vycházelo se proto nejen ze stávajících standardů MPEG-1, MPEG-2, MPEG-4 Part 2, H.261 a H.263, ale byly představeny vylepšení jako je intra kódování, celočíselná transformace 4x4, vícenásobné referenční snímky, proměnná velikost bloku, čtvrt-pixelová přesnost pro kompenzaci pohybu, in-loop deblocking filtr a zdokonalené entropické kódování.

Dále se u H.264/AVC se počítá s nasazením nad širokým spektrem jak současných tak i budoucích sítí. Aby to mohlo být efektivně splněno, byly zavedeny dvě hlavní části a to VCL – video coding layer a NAL – network abstraction layer [28]. VLC reprezentuje video obsah a NAL formátuje tento obsah do odpovídající podoby pro konkrétní síť. Zvýšení efektivity s sebou přineslo i zvýšení časové složitosti kódování i dekódování, proto bylo v H.264/AVC zavedeno několik vylepšení. Jedná se především o transformace, ze kterých bylo odstraněno násobení, v případě přesných transformací se provádí jako součást kvantizace.

Součástí požadavků kladených na tento kodek byl přenos kódovaného videa přes bezdrátové sítě (např. WiFi), což je spojeno s podstatně vyšší ztrátovostí a rušením než na metalických spojích. V případě vzniku chyby nebo ztráty informace docházelo v předchozích kodecích ke značné degradaci kvality obrazu.Byly proto zavedeny techniky zvyšující odolnost signálu vůči chybám a ztrátám. Patří mezi ně například strukturovaná množina parametrů, FMO, záměna řezů nebo nadbytečné řezy.

H.264 zavádí tři profily [20] (v první verzi): Baseline, Main a Extended Profile, a čtyři High profily (ve třetí verzi): High, High 10, High 4:2:2 a High 4:4:4. Později byly přidány aplikačně specifické profily: Stereo High Profile, Multiview High Profile.



Obrázek 4.2: Profily H.264/AVC

Algoritmus kódování H.264/AVC

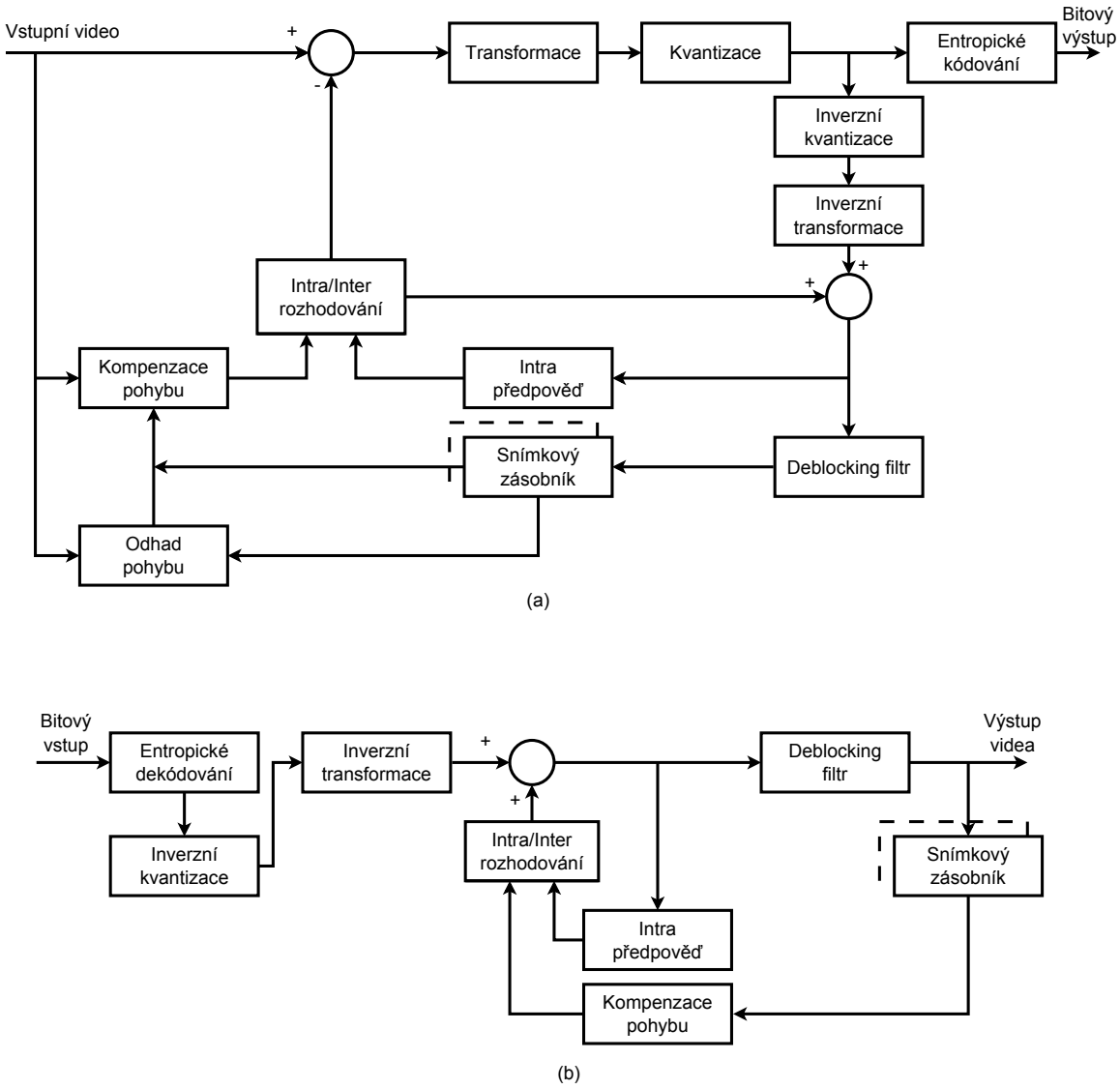
Schéma blokového zapojení jak kodéru tak dekodéru je naznačeno s jistou mírou abstrakce (absence řídících prvků) 4.3. Jak je ze schématu patrné, základní struktura je velice podobná dalším kodekům z kategorie MPEG a H.26x [21].

Vybrané techniky kódování H.264 [33]

Proměnná velikost bloků kompenzace pohybu: Na rozdíl od předchozích standardů je velikost bloků kompenzace pohybu nižší a flexibilnější, nejmenší blok je pro jasovou složku o velikosti 4×4.

Přesnost kompenzace pohybu až 1/4: Zavedeno už ve standardu MPEG-4 part 2, změna v algoritmu interpolace zaměřená na snížení složitosti výpočtů.

Vektory pohybu až za hranice snímku: Na rozdíl od MPEG-2 a jeho předchůdců, kdy vektor pohybu musel ukazovat do oblasti již dekódovaných referenčních snímků, H.264/AVC vychází z H.263, kde byla zavedena extrapolace za hranice snímku jako volitelná funkce.



Obrázek 4.3: Schéma kodéru(a) a dekodéru(b) H.264/AVC

Kompenzace pohybu využívající více snímků: U MPEG-2 se snímky P odkazovaly pouze na jeden z předchozích snímků a snímky B se mohly odkazovat pouze na dva konkrétní snímky. Standard H.264/AVC přináší rozšíření pro přesnější kompenzaci pohybu spočívající v umožnění kodéru vybrat si vhodné referenční snímky z velkého množství již zpracovaných snímků.

Zrušení závislosti pořadí kódovaných a zobrazovaných snímků: V předchozích standardech byla zavedena absolutní závislost mezi pořadím kódovaných snímků a pořadím snímků zobrazovaných. Ve standardu H.264/AVC byla tato závislost z velké části odstraněna, což ponechává volbu pořadí snímků na kodéru, omezení spočívá jen v paměťových možnostech dekodéru.

Možnost použití odkazujících se snímků jako referenčních: Původní přístup neumožňoval použití snímků B jako referenčních snímků. Odstraněním tohoto omezení bylo docíleno zpřesnění odkazování, protože v některých případech je snímek B bližší kódovanému snímku než snímek I nebo P.

Váhovaná predikce: H.264/AVC dovoluje kodéru definovat váhu a posunutí signálu kompenzace pohybu. Vede to k podstatně lepším výsledkům co se týče sekvencí s globální změnou jasu (tmavnutí/světlání).

In-loop deblocking filtr: Kodeky využívající blokovou transformaci obrazu mohou způsobovat artefakty v obraze spočívající v rozdílech na hranách jednotlivých bloků. Standard H.264/AVC rozšiřuje adaptivní deblocking filtr a používá jej v rámci předpovědi kompenzace pohybu pro dosažení lepších výsledků předpovědí dalších snímků.

Snížení velikosti transformačních bloků: Od předchozích standardů využívajících transformačních bloků velikosti 8×8 je zavedena transformace 4×4. Vede to k snížení vlivu okolí v rámci jednoho bloku a tím pádem snižuje artefakty obrazu zvané ringing.

16bitová transformace: Obdobně jako u VC-1 je zpracování prováděno na 16 bitech na místo předchozích 32.

Přesná shoda inverzní transformace: Většina předchozích standardů definovala pouze maximální chybovou odchylku při porovnání obrazu transformovaného a inverzně transformovaného zpět s původním obrazem. Jednotlivé implementace dekodérů tak poskytovaly rozdílné výstupy. H.264/AVC však dosahuje přesné shody.

Obsahově přizpůsobivé kódování: Kódování CABAC a CAVLC.

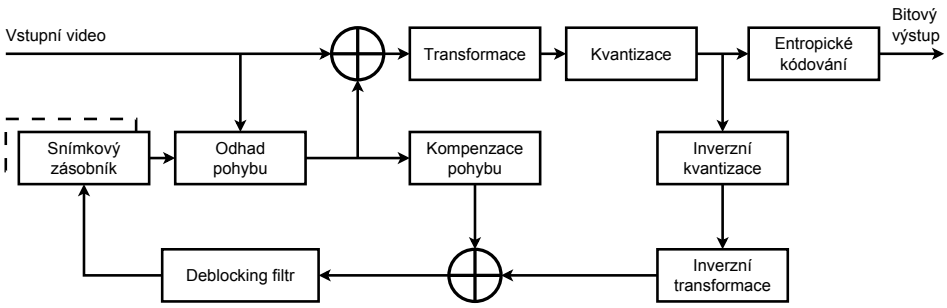
Strukturovaná množina parametrů: Díky vhodně zvolené architektuře přenosové hlavičky je docíleno podstatného zvýšení odolnosti vůči ztrátě dat.

NAL jednotka: Veškerý výstup H.264/AVC je vložen do datového paketu, který je označen jako NAL jednotka. Dojde tak k odstínění formy bitového toku při přenosu přes různé typy sítí.

Snímky SP/SI: Kromě běžných snímků I, P a B byly zavedeny snímky SI a SP.

4.2.3 VC-1

VC-1 je standard [27] videokodeku pocházející od SMPTE (Society of Motion Picture and Television Engineers), samotný proces standardizace byl proveden skupinou C24-Video Compression Technology Committee. Existují celkem tři dokumenty vztahující se k VC-1, SMPTE 421M popisuje samotný kodek, SMPTE RP228 se věnuje implementaci, a SMPTE doplňuje specifikace přenosu dat.



Obrázek 4.4: Schéma kodéru VC-1

Blokové schéma 4.4 řadí VC-1 do skupiny MPEG kodeků. Stejně je tomu i u implementovaných algoritmů, které jsou příbuzné MPEG-4 a H26x, přidává však vlastní optimalizace a vylepšení:

Přizpůsobivá velikost transformačního bloku: Běžné transformace 8×8 mohou vést k artefaktům v obraze, především na okrajích a hranách. VC-1 umožňuje kódovat jednotlivé 8×8 bloky kromě klasického způsobu pomocí dvou 4×8 bloků nebo dvou 8×4 bloků nebo čtyř 4×4 bloků. Vede to k lepším výsledkům co se týká kvality obrazu.

16bitové transformace: Vede ke snížení výpočetní složitosti a možnosti implementace v HW, kde jsou velmi rozšířené digitální signálové systémy (DSP) postavené na 16bitových procesorech.

Kompenzace pohybu: VC-1 umožňuje kombinaci 16×16 a 8×8 bloků v jednom snímku. Existují dva typy filtrů, bikubický a bilineární. Kombinace velikosti bloku, přesnosti a typu filtru jsou uloženy ve čtyřech předdefinovaných módech, které reflektují možné scénáře pohybu v různých snímcích.

In-loop deblocking filter: Zlepšení výsledků metody kompenzace pohybu

Kompenzace změny intenzity jasu: Běžná kompenzace pohybu při scénách se změnou světelnosti neposkytuje dobré výsledky, ve VC-1 je tento problém detekován a změna jasu je kompenzována.

Rozdílná kvantizace: V rámci jednoho snímku nejsou všechny bloky kvantizovány na stejné úrovni, pokud jsou nalezeny významné bloky, ve kterých je žádoucí zachovat více koeficientů, pak tyto bloky jsou na rozdíl od ostatních kvantizovány na nižší úrovni.

4.2.4 VP8

Standard [32] a kodek nezatížený patenty a distribuovaný jako open-source (licence typu BSD [6]). Jako další standardy popisuje především výstupní bitový tok a jeho dekódovování.

VP8 je obdobný kodekům typu MPEG, má velice blízko k H.264/AVC. Je založen na rozdělení snímků do bloků, jejich předpovědi jak intra tak inter snímkové, následné transformaci DCT a WHT (Walsh-Hadamard Transform). Tímto způsobem je využito závislosti obrazových informací jak v prostoru tak čase.

Na rozdíl od předchozích kodeků typu MPEG-2 využívá transformace a inverzní transformace s celočíselnými hodnotami a předem definovanou přesností. Vede to k odstranění problému rozdílných výstupů různých implementací dekodérů (způsobené typicky zaokrouhlovacími chybami nebo posuny), u jiných kodeků se lze setkat s pojmem *Přesná shoda inverzní transformace*. Je důležité podotknout, že konečný výstup videa se může lišit od skutečných dekódovaných snímků, protože mnoho systémů poskytuje funkce vylepšení nebo úpravy obrazu – *postprocessing*.

Barevný model se kterým VP8 pracuje je 8bitový YUV 4:2:0, který vychází z obdobného YCrCb [16].Obraz je postupně zpracováván formou rasterizačního průchodu (shora zleva, dolů doprava).

V některých případech (především pro velmi nízký datový tok) může být vstupní obraz redukován co se rozlišení týče, aby bylo docíleno odpovídající komprese, v dekodéru pak dojde k rekonstrukci na původní rozlišení. Z hlediska uživatele nedojde k žádné změně, tento přístup je označován jako *black box*.

VP8 pracuje nad Y makrobloky 16×16, U,V makrobloky 8×8. DCT i WHT jsou vždy prováděny nad vzorky o velikosti 4×4, konkrétně v rámci jednoho makrobloku proběhne celkem 16 DCT nad jasovou složkou a 8 DCT nad složkami barvy (4×U a 4×V), WHT probíhá nad průměrnými jasovými hodnotami 16 bloků 4×4 v rámci jednoho makrobloku. Výstup je nazýván Y2 a představuje „virtuální“ 25. blok (16Y+4U+4V) makrobloku.

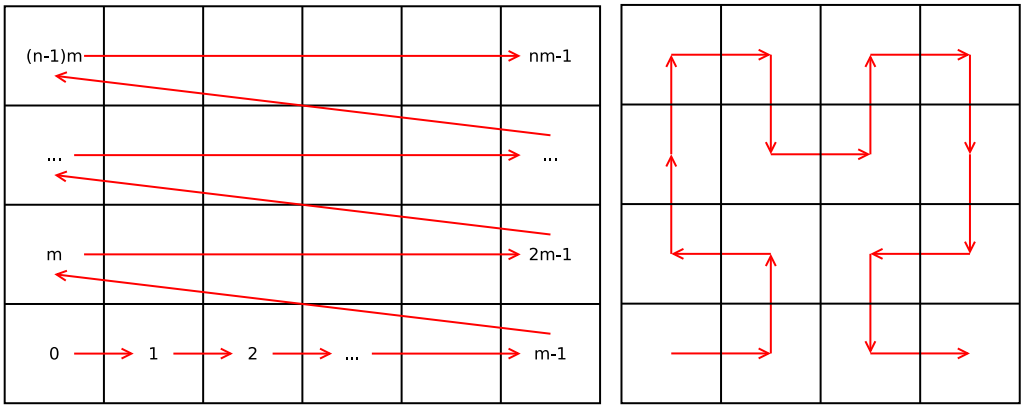
VP8 využívá snímky I a P podobně jako MPEG kodeky, ale nepodporuje oboustrannou předpověď ve formě B snímků. Naopak zavádí zlaté snímky a alternativní referenční snímky (altref). Každý blok může být v inter režimu předpovězen na základě bloků v předchozím snímku nebo posledního předchozího zlatého/altref snímku. VP8 neumožňuje pokračovat v dekódování obrazu pokud dojde ke ztrátě nebo poškození P snímku, proces pokračuje až v momentě příchodu snímku I. Zlaté a altref snímky poskytují částečné řešení tohoto problému. Dochází zde k vytvoření snímku, který obsahuje všechny obsahové změny od posledního klíčového (I, zlatého) snímku. Této techniky se využívá především ve videokonferencích, kdy se při ztrátě znovu zasílá zlatý snímek (pokud je to výhodnější, než přeposílat všechna potřebná data).

4.2.5 Theora

Theora [9] patří mezi univerzální ztrátové kodeky, je uvolněná pod BSD licencí. Vychází z VP3 kodeku, vyvíjeným pod On2. Theora rozšiřuje možnosti VP3, existuje částečná kompatibilita – VP3 obsah lze bezeztrátově převést do theory, zpětně to nemusí být možné. Knihovna libtheora je referenční implementací Theory.

Specifikace kodeku je obdobná standardům MPEG, využívá blokové DCT transformace 8×8 a blokovou kompenzaci pohybu. Odlišuje se však uspořádáním těchto bloků a zpracováním koeficientů. Dalším podstatným rozdílem je absence oboustranné predikce – B snímků. Podporuje barevný prostor YCbCr a podvzorkování barvy 4:2:0, 4:2:2 a 4:4:4.

Výstupem kodéru jsou *raw* pakety, dekodér očekává tyto pakety na vstupu. Není definována žádná konkrétní velikost paketů ani jiná omezení. Očekává se přenos takovým protokolem, který zajistí bezchybný přenos a správné pořadí doručení – například pro přenos souborů je to Ogg formát a pro přenos po síti RTP.



(a) (b)

Obrázek 4.5: (a) Rasterizační průchod polem o velikosti m×n a (b) průchod pomocí Hilbertových křivek.

Vstupní snímek je rozdělen do bloků o velikosti 8×8, kde oblast 4×4 bloků je označována jako superblok. Pokud je barevná složka podvzorkována, jsou její bloky opět tvořeny z 8×8 bodů, ale v superbloku bude těchto bloků méně. Jednotlivé superbloky v jasové a barevné složce se nemusí vzájemně překrývat. Způsob zpracování bloků je vyjádřen rasterizačním průchodem 4.5a. Druhá varianta je rasterizační průchod přes superbloky, zatímco jednotlivé bloky uvnitř každého superbloku jsou procházeny pomocí Hilbertových křivek 4.5b. Dále jsou zavedeny makrobloky, které obsahují 2×2 bloky jasové složky a počet bloků barvy odpovídající úrovni podvzorkování.

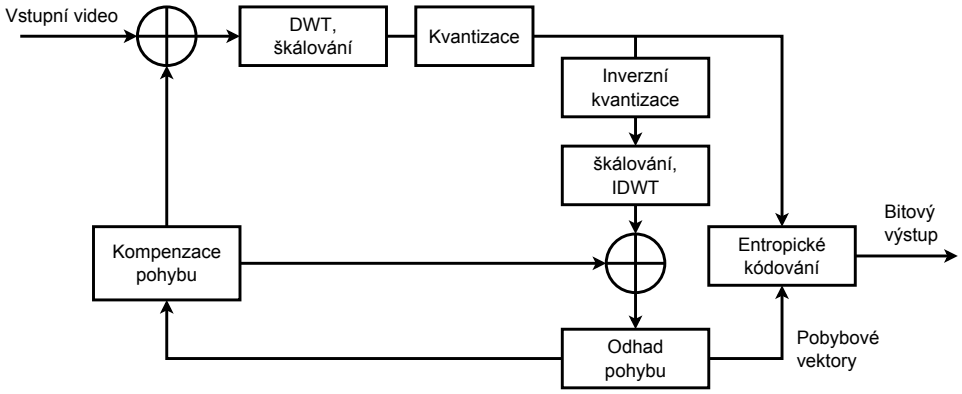
Způsoby kódování snímků vycházejí ze stejných základů jako MPEG, zásadní odlišností (která přibližuje theoru k VP8) je absence B snímků.

Jak již bylo zmíněno, k transformaci je použito DCT nad maticí 8×8 a následně jsou koeficienty kvantizovány. Theora dovoluje definovat až 384 různých kvantizačních matic s faktorem kvantizace v rozsahu 0–63. Koeficienty jsou převedeny do lineárního formátu klasickým cik-cak průchodem. Na výsledek je aplikováno Huffmanovo kódování s RLE, theora disponuje 80 variantami kódových tabulek.

4.2.6 Dirac

Standard [7] a kodek vyvinutý v BBC využívající diskrétní vlnkové transformace a kompenzace pohybu. Cílové nasazení je široké, od streamování přes internet, přes SD i HD televizní vysí-

lání a digitální kinematografickou produkci a distribuci videa až po *embedded* zařízení. Tento formát je uvolněn pod licencí RF (royalty-free). Dirac podporuje jak ztrátové, tak bezeztrátové komprese, mód intra snímkového kódování pro nasazení v profesionálním sektoru zpracování videa, speciální režim nízkého zpoždění pro vysílání v HD formátu, módy s kompenzací pohybu a dlouhým GOP pro distribuci především na optických médiích a postupný přechod poměru kvality a komprese.



Obrázek 4.6: Schéma kodéru Dirac

Základem diracu je diskrétní vlnková transformace do více rozlišení. Zvyšování úrovně komprese vede k postupnému snižování rozlišení. Pokud je požadována nízká výpočetní složitost dekódování, lze využít tu část dat, která reprezentuje nízké rozlišení. DWT se provádí nad celým snímkem.K dispozici je celá sada filtrů počínaje CDF(9,7).

Kódování může probíhat ve dvou variantách a to standardní, která využívá kompenzaci pohybu a pokročilé aritmetické kódování, což vede k vysokému kompresnímu poměru při zachování rozumné kvality, nebo v rychlém módu, kdy je použito jen intra kódování a jednoduché VLC (variable length coding) jako forma entropického kódování. Kompenzace pohybu v inter snímkovém módu je založena na OBMC – obraz je rozdělen do navzájem přesahujících se bloků.

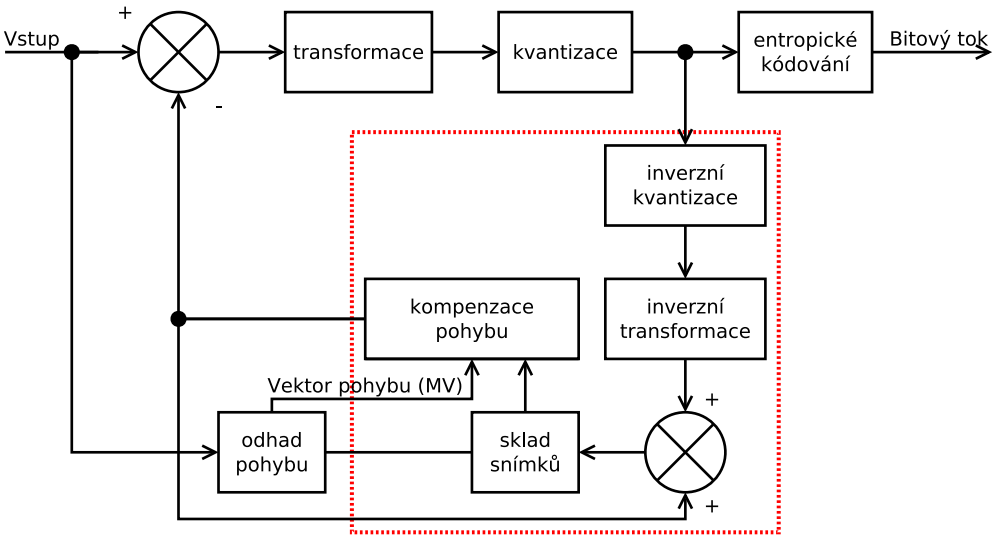
Podvzorkování barevných složek je ve variantách 4:4:4, 4:2:2 a 4:2:0 s bitovou hloubkou 8, 10, 12 a 16. K dispozici je i bezeztrátové RGB kódování.Výstupní bitový tok obsahuje jednotlivé snímky očíslované a uspořádané do obousměrného seznamu, pro efektivní navigaci v toku.

Kromě referenčního kodeku dirac je k dispozici i kodek Schrödinger – libschroedinger, který je více optimalizován a dosahuje tak lepších výsledků co se týče rychlosti.

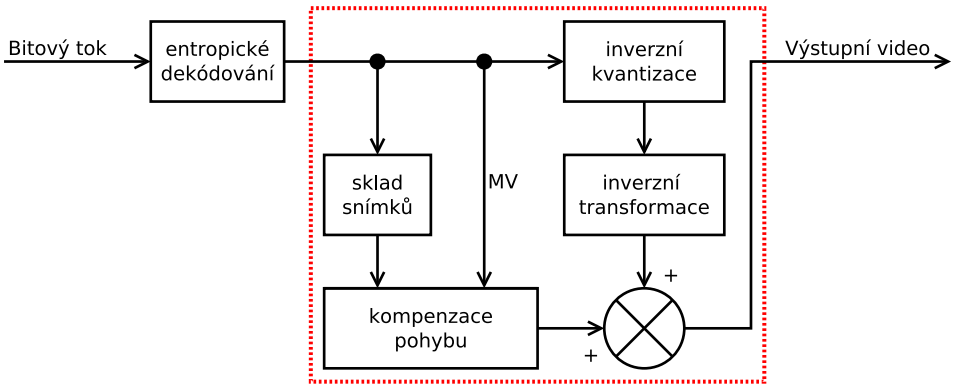
Dirac Pro (označovaný také jako VC-2) je navrhovaná speciální verze diracu (obsahující podmnožinu funkcí) určená pro aplikace zpracování videa – využívá pouze I snímky.

4.3 Kodeky

Kodekem se v této části rozumí softwarová implementace algoritmů kódování videa, ve většině případů popsaných standardy. Kodek se skládá, jak již jeho název napovídá, ze dvou hlavních částí a to kodéru a dekodéru. Kodér je implementací kódovacích algoritmů jako je DCT, DWT, kvantizace nebo entropického kódování a slouží především pro převod vstupního nekomprimovaného videa do bitového formátu, který dosahuje značné komprese vůči originálu. Naopak dekodér je část kodeku provádějící zpětné převedení bitového toku na video, které (do určité míry – kvality) koresponduje se vstupem kodéru. Využívá algoritmů jako je entropické dekódování, inverzní kvantizace, IDCT atd.



Obrázek 4.7: Schéma kodéru



Obrázek 4.8: Schéma dekodéru

4.3.1 x264

x264 je kodek implementující standard H.264/AVC, jedná se o volně dostupnou knihovnu (a program) šířenou pod GNU GPL. Umožňuje kódování několika FullHD streamů najednou na běžném počítači, dosahuje vynikajících vizuálních kvalit, podporuje technologie jako je pozemní i satelitní televizní vysílání, záznam na optická média včetně Blu-ray, umožňuje nasazení ve videokonferenčním prostředí. Nejpodstatnějšími fakty podporující použití x264/H.264 je však špičková komprese a masivní rozšíření a podpora jak v SW a webu (Youtube, Facebook, Vimeo…), tak HW – videopřehrávače, rekordéry, set-top-boxy, mobilní zařízení atd.

Podporované funkce: adaptivní transformace 8×8 a 4×4, přizpůsobivé umístění B snímků, AFO, B snímky mohou být použity jako referenční, entropické kódování CAVLC/CABAC, vlastní kvantizační matice, intra kódované I snímky a inter kódované P snímky: všechny velikosti makrobloků (16×16, 8×8, 4×4), inter kódované P snímky: 16×16 a 8×8, vícenásobné referrnční snímky, nastavení pomocí kvatizačního faktoru, kvality, jednoprůchodové nebo vícepruchodové zpracování, detekce střihu ve scéně, prostorový i časový přímý režim pro B snímky, paralelní zpracování, bezeztrátové kódování s využitím inter snímkové predikce, přizpůsobivé kvantizační matice,

Nedostatky: Chybějící adaptivní MBAFF pro prokládaná videa

4.3.2 HM 1.0

HM 1.0 je implementací TMuC (Test Model under Consideration), testovacího modelu [15] vyvíjeného standardu HEVC (High Efficiency Video Coding) označovaného též jako *H.265* [34]. Kodek je ve ve stádiu testování jednotlivých návrhů. Plánované vydání standardu spadá do roku 2012. Základním cílem je dosažení v Baseline profilu o 50% nižší velikosti komprimovaného videa než u H.264 High profilu, pochopitelně se zachováním odpovídající kvality. HM 1.0 je dostupný ve formě zdrojových kódů.

4.3.3 Xvid

Kodek se řadí mezi open source, všechny zdrojové kódy jsou uvolněny pod licencí GNU GPL. Jedná se o přímou konkurenci komerčního DivX. Mezi jeho hlavní výhody patří multiplatformnost.

Kodek Xvid implementuje MPEG-4 SP a MPEG-4 ASP. Mezi hlavní vlastnosti patří použití B snímků, přesnosti odhadu pohybu na 1/4 bodu, globální odhad pohybu, vlastní kvantizační matice. Algoritmy kodéru a dekodéru jsou optimalizovány jak pro více vláknové/procesorové systémy, tak i pro rozšiřující instrukční sady jako je SSE2/SSE3.

4.3.4 DivX

Jméno DivX označuje firmu a její produkty, mezi něž patří kodek DivX, který se čtenáři pod tímto názvem zřejmě nejčastěji vybaví. Tento kodek byl v počátcích vyvíjen jako open source software, od verze 5 se stal uzavřeným, což vedlo k oddělení skupiny vývojářů, kteří založili projekt Xvid. Kodek Divx pro Windows prošel několika verzemi:

DivX 4 – prvotní verze, podporuje všechny typy snímků, vychází z MPEG-4 part 2 a je kompatibilní s MPEG-4 part 3

DivX 5 – kódování typu MPEG-4 part 2 ASP/SP, QPEL, použití adaptivních metod obousměrné predikce B snímků, výběr kvantizace H.263 nebo MPEG-2

DivX 6 – vnitřní optimalizace vedoucí ke zvýšení kompresního poměru, zvýšení obrazové kvality, podpora pro více-vláknové (HT) a více-jádrové procesory společně s optimalizací pro SSE2 a SSE4, kódování FullHD (1080p) obsahu, několikanásobné zrychlení procesu dekódování, MMX optimalizace pro MPEG kvantizace atd.

DivX 7 – podpora dekódování H.264 obsahu v kontejnerech mkv

DivX Plus HD – implementace standardu H.264/AVC a použití kontejneru mkv, rozlišení až 1920×1080 a 30 fps nebo 1280×720 a 60fps, profily Baseline, Main a High, maximální datový tok 20 Mbit/s

4.3.5 Windows Media Video 9

Windows Media Video 9 nebo také WMV9 je implementace standardu SMPTE 421M (VC-1) vytvořená společností Microsoft. Tento kodek je určen především pro vysílání videa přes různá média, ať už se jedná o internet nebo digitální vysílání (satelitní i pozemní), dále pro ukládání obsahu v HD kvalitě na DVD i Blu-ray. Microsoft podporuje tento kodek ve všech fázích – pro kódování lze použít Windows Media Encoder 9, pro distribuci po internetu Windows Media Services server a pro přehrávání Windows Media Player. WMV9 poskytuje ochranu DRM – Digital rights management. Spočívá v licencování obsahu, který je zpřístupněn (lze jej přehrát) jen v případě, že uživatel disponuje danými právy.

WMV9 využívá také vlastního kontejneru. Jedná se o ASF – Advanced Systems Format, fakt, že obsah je kódován pomocí WMV9 a uložen v ASF je často reprezentovaný soubory s příponou wmv. ASF podporuje DRM, a šifrování DES, SHA-1.

4.3.6 FFmpeg

FFmpeg je open source projekt rychlého video i audio konvertoru. Je dostupný pod dvěmi licencemi, a to GNU LGPL a GNU GPL. Jeho hlavní součástí jsou knihovny libavcodec a libavformat. První představuje kodek, druhá zpracovává audio/video streamy pro jednotlivé kontejnery. FFmpeg lze považovat za úplné řešení pro nahrávání, konvertování a streamování audia i videa. Vývoj probíhá především pro platformu Linux, ale je přenositelný i na jiné platformy.

Vývoj FFmpeg je neustálý proces, proto žádná z jeho částí nemá standardní verze, tak jak je tomu u většiny programů. Naopak jsou k dispozici pravidelné updaty. To může vést do jisté míry k omezení funkčnosti aplikací, které využívají FFmpeg, ale na druhou stranu jsou takto rychle implementovány nejnovější technologie a podpora pro nové formáty.

libavcodec

Libavcodec je open source knihovna distribuovaná pod licencí GNU LGPL složící ke kódování a dekódování video a audio obsahu. Je vyvíjena jako součást projektu FFmpeg. Využívá ji mnoho open source přehrávačů, kodérů nebo editačních programů (VLC media player, xine, MPlayer, MEncoder, Avidemux atd.). Knihovna libavcodec je dostupná pro všechny hlavní operační systémy – Windows, Linux a OS X.

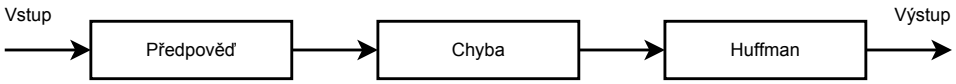
Specifickou vlastností knihovny libavcodec je implementace některých proprietárních standardů (jejichž specifikace nebyly uveřejněny) jako dekodérů a někdy i kodérů. Reimplementace těchto komerčních kodeků je možná díky reverznímu inženýrství. Hlavní výhodou je lepší přenositelnost a v některých případech i vyšší výkon díky požití vysoce optimalizovaných komponent.

Několik vybraných kodeků implementovaných v libavcodec: MPEG-1, MPEG-2, H.263, MPEG-4 ASP and SP, WMV 7, WMV 8, FLV1, HuffYUV, Lagarith, FFV1, Snow, RealVideo. Některé standardy jsou implementovány pouze jako dekodér: MPEG-4 AVC/H.264, VC-1, Theora, Dirac, Schrödinger.

4.3.7 HuffYUV

Videokodek umožňující bezeztrátovou kompresi videa. Primární určení je pro editaci, respektive pro kompresi zachycených dat, nebo kompresi dat v průběhu editace. HuffYUV pracuje nad barevným prostorem YCbCr (přestože název napovídá YUV).

Algoritmus komprese snímků vychází z předpovědi hodnoty bodu na základě sousedních bodů, kde se následně vypočítá rozdíl mezi originální hodnotou a předpovědí a označí se jako chyba. Na tento výsledek je aplikováno Huffmanovo kódování. Pro každý barevný kanál je použita jiná tabulka, tyto tabulky jsou již předdefinovány v kodeku pro lepší zpětnou kompatibilitu, lze použít ale i jiné. Pro RGB kompresi je z důvodu lepších výsledků komprimováno ne R, G, B, ale R-G, G, B-G.



Obrázek 4.9: Schéma HuffYUV

Dekomprese vychází z opačného procesu, kdy se získá chybová hodnota pro daný pixel z Huffmanem zakódovaných dat, spočítá a přičte se předpověď, výsledkem je původní pixel. Tento proces je založen na faktu, že předpověď se počítá z již známých (dekódovaných) pixelů a je dán počáteční pixel.

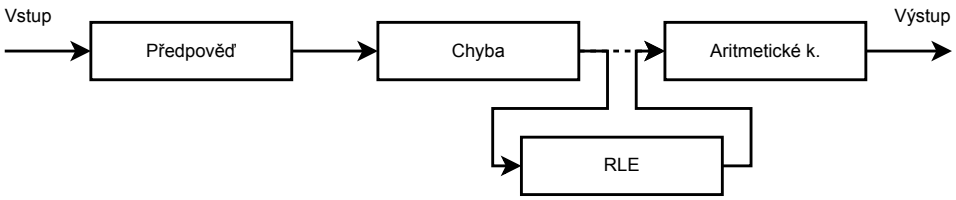
Mezi nedostatky tohoto kodeku patří využití jen jednoho vlákna.

Kodek i zdrojové kódy jsou distribuovány pod GNU GPL.

4.3.8 Lagarith

Jedná se o bezeztrátový kodek. Je vhodný pro editaci videa nebo archivaci. V případě dostatečně výkonného systému jej lze použít i pro kompresi zachytávaného videa. Nejčastěji bývá porovnáván s HuffYUV kodekem. Proti němu poskytuje lepší algoritmy komprese, tudíž velikost výsledného datového toku je nižší. Kóduje do RGB24, RGB32, RGBA, YUY2 a YV12.

Algoritmus použitý v kodeku Lagarith částečně vychází z HuffYUV a přidává další kompresní prvky jako je RLE – Run Length Encoding a Fibonacciho kódování. Samotný proces komprese je podobný. Prvním krokem je výpočet předpovědi pro daný bod a rozdíl od skutečné hodnoty bodu. V případě HuffYUV kodeku by následovalo použití Huffmanova kódování, v případě Lagarithu se uplatní RLE pro eliminaci sekvencí nul. Může být inicializováno při detekci jedné, dvou nebo tří po sobě jdoucích nul. V případě, že by použití RLE nevedlo ke zlepšení komprese, tak je sekvence ponechána beze změn. Dalším krokem je použití Aritmetického kódování, které poskytuje podstatně lepší výsledky než Huffmanovo kódování, komprese odpovídá mnohem přesněji entropii kódovaných dat.



Obrázek 4.10: Schéma Lagarith

Kapitola 5

Srovnání videokodeků

V této kapitole je pojednáváno o samotném srovnání vybraných kodeků. Je rozčleněna do jednotlivých částí reprezentující jednotlivé fáze srovnání. Kodeky jsou rozděleny na ztrátové a bezeztrátové.

5.1 Návrh testování

Kodeky jsou nasazovány v širokém spektru aplikacích, každý z nich má své výhody a nevýhody. Z toho vyplývá, že je těžké nebo až nemožné určit jednoznačně nejlepší kodek. Z této úvahy vychází i návrh testování a srovnávání kodeků, kdy se v potaz berou různá cílová použití a s tím spojené odlišné požadavky na konkrétní vlastnosti.

Testovací videa

Ideální situace by nastala, kdyby uživatel, který vybírá vhodný kodek pro zakódování videa, měl k dispozici výsledky srovnání nad jeho videem. To pochopitelně není realizovatelné z mnoha důvodů. Na prvním místě je časová a výpočetní náročnost. Pokud srovnáváme více než deset kodeků s různými nastaveními a pro několik datových toků, tak je vhodné omezit délku testovaného vzorku. Dále nelze testovat kodeky na každém možném videu, je proto důležité stanovit charakter testovacího videa tak, aby co nejlépe reprezentoval skutečná videa. Dále je důležité aby obsah byl pokud možno nekomprimovaný, ale nejen v aktuální podobě, ale především nikdy předtím, nebo minimálně kódovaný kodekem, který není součástí srovnání a ani není příbuzný s žádným z testovaných. Některé kodeky mohou vykazovat lepší výsledky při znovukódování obsahu jimi již jednou kódovaným.

Charakteristika testovací sekvence `composition`:

- 720p, 24 fps
- statická scéna, minimum pohybu
- scéna s pohybem kamery
- scéna se střední mírou pohybu
- titulky
- přechody, efekty

Charakteristika testovací sekvence `game`:

- 720p, 30 fps, 60 fps
- scéna se střední mírou pohybu

Obrázek 5.1: Ukázky jednotlivých scén ve videu `composition`

Obrázek 5.2: Ukázky jednotlivých scén ve videu `game`

- scéna s vysokou mírou pohybu
- umělá scéna – CG, hry
- změny barevnosti – CC
- scéna s *motion blur* – pohybovým rozmazáním

Rozlišení, fps a datový tok

Tyto tři hodnoty mají mezi sebou určitou formu závislosti, je tedy potřeba zvolit správný rozsah datového toku pro konkrétní rozlišení a fps. V tomto srovnání jsem se rozhodl použít oblíbené rozlišení 720p, při 24 fps. Rozsah datového toku byl stanoven na 400–7000 kbitů/s.

Vybrané kodeky byly také testovány s 30 fps a 60 fps.

Nastavení kodeků

Nastavení kodeků je zřejmě nejdůležitějším momentem testování. Vzhledem k počtu a různorodosti testovaných kodeků jsem ponechával detailní nastavení kodeků implicitní. Profily byly vybrány takové, které tvůrce kodeku doporučoval jako nejlepší v poměru kvality/výkonu. Pokud bylo dále nutné nastavit kvalitu, byla vybrána nejvyšší možná.

Testovací sekvence byla kódována jak s pomocí kvantizéru, tak s nastavením datového toku.

5.2 Techniky srovnávání

Při hledání optimálního kodeku pro konkrétní oblast nasazení sledujeme několik parametrů. V tomto případě je na prvním místě poměr kvality a datového toku videa. Zatímco datový tok lze určit poměrně snadno, u kvality je to složitější. Existují subjektivní a objektivní metody.

Subjektivní metody jsou založeny na vnímání kvality videa pozorovatelem, jejich výhoda spočívá v tom, že vyhodnocení probíhá lidským vnímáním, je tedy nejbližší cílové skupině (lidé), která bude sledovat komprimované video. Nevýhodou je časová náročnost, lidské zdroje, nekonzistentnost výsledků atd.

Objektivní metody jsou matematické modely, které se snaží přiblížit výsledkům subjektivních metod, přičemž k určení kvality není potřeba člověk, výpočet je exaktní a objektivní. Díky tomu jej lze automatizovat a opakovaně aplikovat i na velké objemy dat. Hlavním nedostatkem je právě nedokonalá aproximace vnímání kvality videa člověkem, což může vést k situaci, kdy objektivní metoda poskytuje dobré výsledky, zatímco člověk vnímá kvalitu videa jako horší a vice versa.

PSNR a SSim

Asi nejčastěji používanou objektivní technikou je PSNR (Peak Signal to Noise Ratio), špičkový odstup signálu od šumu [17][19]. Je založen na MSE (Mean Square Error), střední kvadratická odchylka 5.1, kde A je původní obraz, B reprezentuje komprimovaný obraz.

$$MSE(A,B) = \frac{1}{mn}\sum_{x=0}^{m-1}\sum_{y=0}^{n-1}\left[A\left(x,y\right) - B\left(x,y\right)\right]^2 \tag{5.1}$$

Vzhledem k tomu, že výsledné hodnoty MSE jsou velkého rozsahu, je PSNR 5.2 zobrazeno logaritmicky, jednotkou je dB. Ztrátové kodeky obvykle dosahují od 30 do 50 dB, čím vyšší je dosažená hodnota, tím menší je rozdíl oproti originálu.

$$PSNR(A,B) = 10 \cdot \log_{10}\frac{\left(2^{bit\_depth}-1\right)^2}{MSE(A,B)} \tag{5.2}$$

Metoda SSim (Structural Similarity) 5.3 [31] dosahuje vyšší úrovně aproximace než PSNR. μ_A je průměr A, σ_A je rozptyl, σ_{AB} je kovariance a c_2, c_2 jsou konstanty vycházející z bitové hloubky obrazu.

$$SSIM(A,B) = \frac{\left(2\mu_A\mu_B + c_1\right)\left(\sigma_{AB} + c_2\right)}{\left(\mu_A^2 + \mu_B^2 + c_1\right)\left(\sigma_A + \sigma_B + c_2\right)} \tag{5.3}$$

Jak je ze vztahů patrné, obě metody vyžadují dostupnost původního i komprimovaného videa, takové metody označujeme jako *Full Reference – FR*.

Pakliže opakujeme srovnání PSNR nebo SSim na stejném videu, které je kódováno s různou úrovní kvantizace nebo různým datovým tokem (ostatní nastavení kodeků musí zůstat stejné), je možné z vypočtených hodnot sestavit graf s datovým tokem na horizontální ose a kvalitou vyjádřenou pomocí PSNR a SSim na ose vertikální. Z něj lze poměrně snadno určit, který kodek poskytl lepší poměr kvality k datovému toku [22].

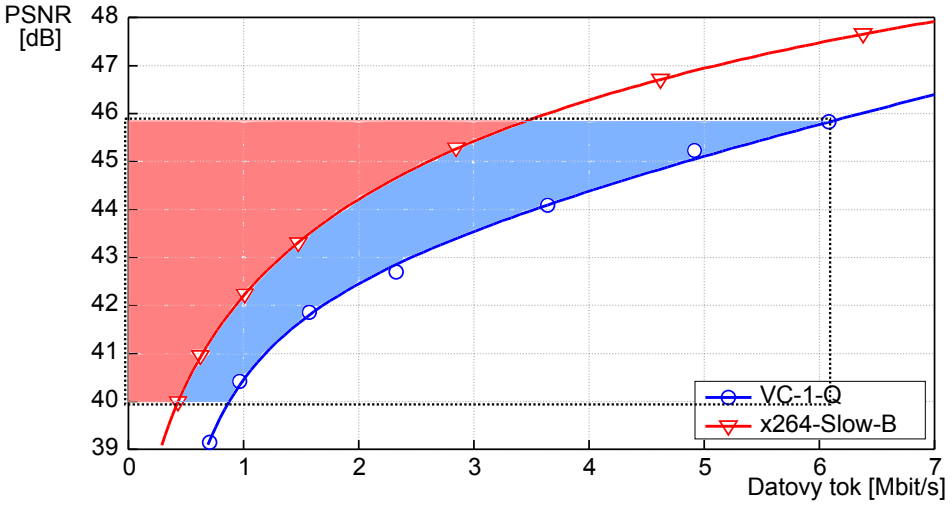
Bjontegaard Delta PSNR

Mnohem obtížněji jde však říci, o kolik procent daný kodek poskytl lepší kompresi než jiný kodek. Zde srovnání probíhá v grafu na horizontální úrovni, kdy se díváme na vzdálenost mezi jednotlivými kodeky pro jednu konkrétní hodnotu kvality. Pokud by se toto srovnáni zopakovalo pro všechny úrovně kvality a udělal se z těchto hodnot průměr, dostali bychom hodnoty, z kterých by již snadno šlo odvodit procentuální rozdíl mezi kompresí jednotlivých kodeků. Realizace tohoto postupu však s sebou přináší problémy. Především jde o to, že k dispozici je jen velmi omezené množství konkrétních hodnot PSNR/SSIM, navíc tyto hodnoty mezi jednotlivými kodeky jen velmi zřídka leží na stejné úrovni kvality. Proto je nutné chybějící hodnoty nějakým způsobem určit. Metoda, která řeší tento problém se nazývá BD-PSNR (Bjontegaard Delta PSNR) [11].

Nabízí se proložení bodů v grafu funkcí. Lineární ani polynomiální regrese neposkytuje dostatečně uspokojivé výsledky, naopak proložení logaritmickou funkcí $a \log x + b$ ve většině testovaný případů poměrně přesně procházelo vypočtenými body. Podle [12] lze dosáhnout ještě lepšího výsledku logaritmizací dat podle datového toku a následné proložení kubickým polynomem 5.4.

$$a \log^3 x + b \log^2 x + c \log x + d \tag{5.4}$$

Dále je nutné omezit rozsah úrovní kvality na takový, aby se do výpočtu nezanesly zbytečně velké chyby vzniklé extrapolací. Proto je zvoleno nejmenší z maxim hodnot kvality a největší z minim hodnot kvality. Tím jsou ohraničeny plochy mezi jednotlivými funkcemi, jejichž velikost udává rozdíl komprese kodeků. U logaritmického proložení lze funkce invertovat a následně spočítat určitý integrál. U polynomiálního vyjádření inverzní funkci najít není snadné a ani výpočet integrálu není přímočarý. Podoba problému přímo vede k použití numerické metody výpočtu určitého integrálu – Monte Carlo. Oba počítané integrály lze navíc určit jedním „průchodem“ metody Monte Carlo. Počet generovaných bodů byl stanoven na 1000000. Z vypočtených ploch lze určit procentuální rozdíl mezi naměřenými výsledky. Přestože je tato metoda určena především na vyhodnocení PSNR grafů, lze ji stejně aplikovat i na grafy SSim.



Graf 5.3: Příklad proložení s vyznačenými oblastmi a plochou Monte Carla

Přesnost těchto metod je značně omezená počtem původních bodů, přesností proložení a přesností výpočtu integrálu, nicméně pro účely shrnutí množství často nepřehledných výsledků do jedné hodnoty dostačuje.

5.3 Implementace srovnávacího nástroje

Pro samotné srovnání kodeků byl vytvořen C++ program. Jeho cílem je automatické srovnání několika kodeků najednou, včetně vhodné prezentace výsledků. Existuje několik důvodů k vytvoření tohoto programu, především nedostupnost obdobného řešení ve formě volného software, nicméně ani komerční podoby, která by poskytovala požadované funkce.

První implementace využívala knihoven OpenCV 2.2[1], pro načtení videa a efektivní operace nad jednotlivými snímky (přibližně trojnásobné zrychlení oproti cyklickému zpracování obrazu jako pole), při pozdějším testování se však ukázalo, že samotné OpenCV neposkytuje všechny potřebné informace a také v některých případech nepoužívá framework VfW (při použití pod Windows, jak je uvedeno v dokumentaci), ale FFmpeg. To vede k nejednoznačnosti a možnému chybnému chování. Proto byla aplikace upravena a část načtení videa byla implementována s přímo pomocí knihoven FFmpeg. Zpracování jednotlivých snímků dále využívá OpenCV. Pro urychlení srovnávacího procesu, který může být časově náročný, byla použita knihovna OpenMP[2] [8], která slouží ke snadné paralelizaci výpočetně náročných bloků kódu. Zrychlení na víceprocesorových nebo vícevláknových systémech je značné (na Phenom II X6 přibližně čtyřnásobné).

Na vstupu aplikace je netlist s cestami k originálním a zakódovaným souborům. Program zpracuje netlist a paralelně srovnává jednotlivá videa snímek po snímku, počítá datový tok, PSNR a SSIM. Následně provede proložení a výpočet BD-PSNR. Kromě textového výstupu jsou generovány vektorové grafy vypočtených hodnot ve formátu svg. Součástí výstupu jsou i grafy změn kvality jednotlivých snímků v čase pro PSNR a SSim.

5.4 Sestavení testovacích videí

Cílem této fáze bylo najít takové sekvence, které by co nejlépe pokryly skutečné použití kodeků. Omezujícím faktorem bylo dodržení autorských práv, které vylučuje použití například scény z

[1] Open Source Computer Vision Wiki, http://opencv.willowgarage.com/wiki/.
[2] The OpenMP API specification for parallel programming, http://openmp.org/wp/.

filmu. Dále bylo nutné najít takové sekvence, které neprošly nějakou výraznější kompresí – to by mohlo vést ke zvýhodnění některých kodeků.

Základní testovací video `composition` 5.1 bylo spojeno z několika částí (videa používaná k testování TMuC h265, dostupná s minimální kompresí, ve vysokém rozlišení a odpovídajícím fps). Video začíná přechodem z černé barvy do scénky s parní lokomotivou, pokračuje scénkou s restaurací, scénkou z parku přechodem do scénky tenisu zábleskem a scénkou z dálnice, zakončené titulky. Záměrně byly zvoleny scénky s různou barevností, úrovní jasu, pohybem kamery a dynamičností, aby bylo dosaženo alespoň částečně podoby filmu. Stejně tak i přechody byly doplněny, aby bylo možné pozorovat jak si s nimi poradí jednotlivé kodeky. Celá sekvence je v rozlišení 1280×720 a 24 fps, délka je 460 snímků. Na `composition` bylo provedeno základní porovnání kodeků v jednotlivých kategoriích. K vytvoření testovacích sekvencí byl použit editor videa Sony Vegas 10.

Bylo také sestaveno speciální testovací video `game` 5.2, které je velmi dynamické, jedná se o sestřih akční hry obsahující umělý motion blur.

5.5 Kódování testovacích videí

Většina kodeků umožňuje nastavovat kvalitu výstupního videa pomocí několika technik. Základní dvě skupiny tvoří nastavení datového toku a nastavení kvantizéru. Některé kodeky však nedovolují jedno z těchto nastavení, některé je nedodržují. S ohledem na tento fakt byly testované kodeky nastaveny takovým způsobem, jakým dovolovaly a jaký poskytovat relevantní výsledky. U kodeků, které umožňovaly jak nastavení datového toku, tak nastavení kvantizéru, bylo testované video `composition` zakódované oběma způsoby a následně bylo vyhodnoceno, který poskytuje lepší výsledky. Ve srovnání lze nastavení rozeznat podle písmena na konci názvu kodeku. B – *bitrate* reprezentuje datový tok, Q – *quantizer* představuje kvantizér.

Né všechny kodeky lze kódovat pomocí jedné aplikace, proto kromě základního nástroje FFmpeg bylo nutné použít VirtualDub, MEncoder a WMNicEnc. Nástroje jako Windows Media Encoder (VC-1), DivX Encoder Plus (DivX HDPlus, DivX HD720), Microsoft Expression Encoder 4 (VC-1) neumožňovaly z různých důvodů použití ve srovnání – placené (omezená funkčnost), nemožnost nastavit požadované hodnoty, padání aplikace, nedodržení zvoleného formátu atd. Nutno podotknout, že byly testovány na Windows XP 32bit a Windows 7 64bit. Pro VC-1 se podařilo nalézt alternativu WMNicEnc a pro DivX byl použit VirtualDub. Jediný DivX HDPlus (H.264) nebyl testován.

Verze použitých programů:

- VirtualDub 1.9.11
- WMNicEnc v1.02 beta
- MEncoder 1.0rc4-4.4.5

Verze FFmpeg (git-a304071) a externích knihoven:

- Schroedinger 1.0.10
- libvpx 0.9.6
- x264 git-aa21558b
- Xvid 1.3.1

Samotná srovnávací aplikace používá FFmpeg k načtení videa, v některých případech však selhává (HM 1.0), nebo nedekóduje video korektně. Proto byly tyto problematické sekvence zakódovaný bez komprese ve stejném barevném prostoru a testovány dále v tomto formátu.

Z toho plyne několik omezení (za názvem kodeku *U* od anglického *uncompressed*), především nemožnost zjistit typy snímků a datový tok nebo také značná velikost těchto sekvencí. Shodnost původního zakódovaného videa a videa bez komprese byla ověřena v následující kapitole 5.6.1.

Samotné kódování bylo prováděno skripty (v případech, kdy to bylo možné – FFmpeg, MEncoder, VirtualDub).

Verze testovaných kodeků pod Windows:

- VP8 v1.0
- Xvid v1.3.1
- DivX 6.9.2
- ffmpeg2theora 0.27

5.6 Srovnání ztrátových kodeků

Srovnání deseti kodeků s více nastaveními není možné provést najednou z důvodu nepřehlednosti grafické prezentace. Proto byly kodeky rozděleny do skupin, částečně podle příbuznosti, částečně podle očekávaných výsledků.

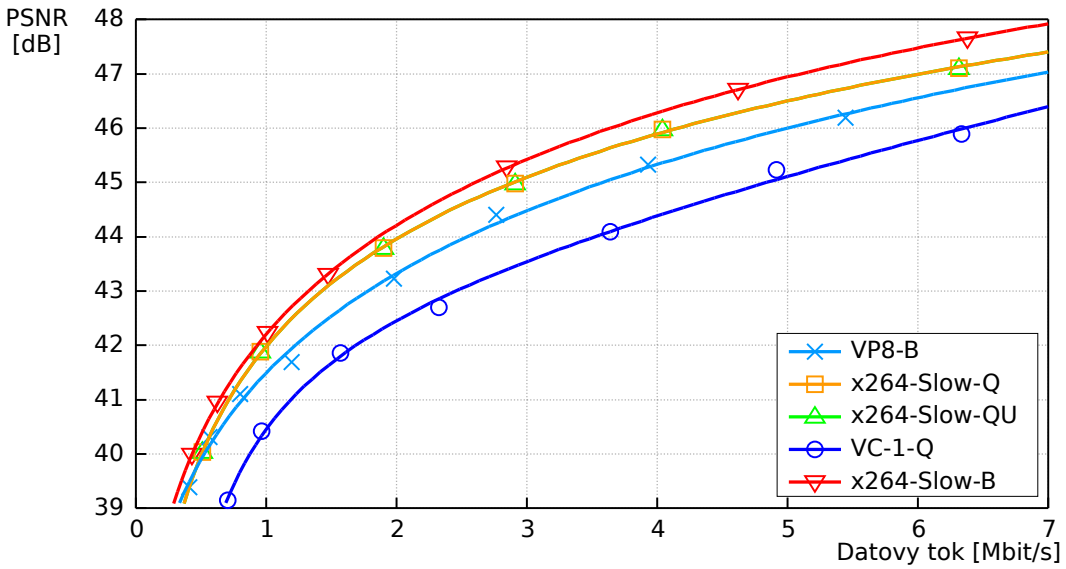
Základní skupiny tvoří:

- x264, VP8, VC-1
- Xvid, DivX, MPEG-4 Part2 (libavcodec)
- Dirac, Schrödinger, Theora
- HM 1.0

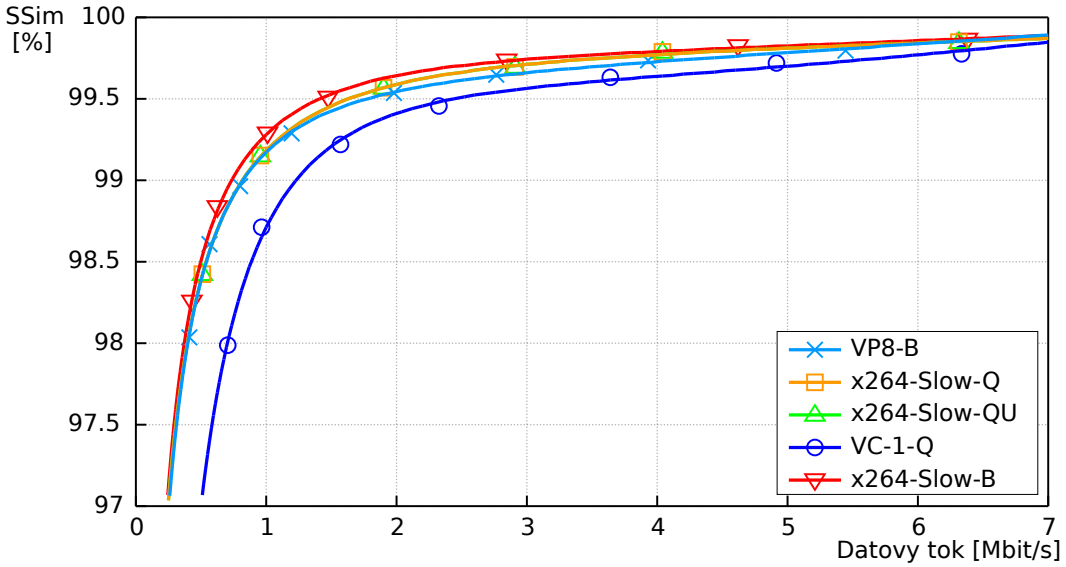
Jeden z kodeků v první skupině je vybrán jako referenční, je srovnáván ve všech ostatních skupinách. Všechny srovnání v těchto skupinách jsou provedeny na základním testovacím videu `composition`. Vybrané kodeky jsou dále srovnávány na dalších testovacích videích.

5.6.1 x264, VP8, VC-1

Jedním z cílů první části srovnání bylo rozhodnout, který kodek bude použit jako referenční i pro další části srovnání. Předpoklad byl x264, nicméně v testu byly přítomny verze B i Q, dále i VP8, který se v předběžných testech blížil x264. Součástí srovnání bylo i ověření shody výsledků kodeku u videa zakódovaného (*x264-Slow-Q*) a následně dekódovaného (*x264-Slow-QU*).



Graf 5.4: Proložení PSNR



Graf 5.5: Proložení SSim

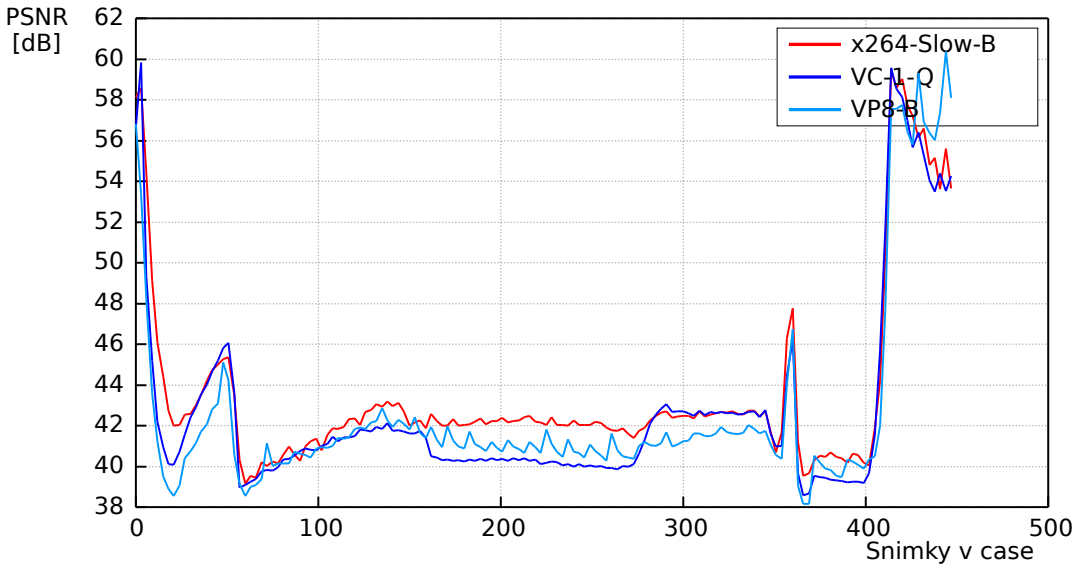
Z grafů je patrné, že *x264-Slow-Q* a *x264-Slow-QU* („dekomprimovaná“ verze *x264-Slow-Q*) dosahují totožných výsledků, potvrzují to i hodnoty BD – rozdíl 0,0% pro přímé srovnání, rozdíl cca. 0,5% pro nepřímé srovnání, chyba Monte Carla.

	Velikost datového toku v poměru k x264-Slow-B (100[%])			
Metrika	x264-Slow-Q	x264-Slow-QU	VP8-B	VC-1-Q
PSNR	111,4	111,5	135,5	176,7
SSim	112,2	112,6	123,1	171,5

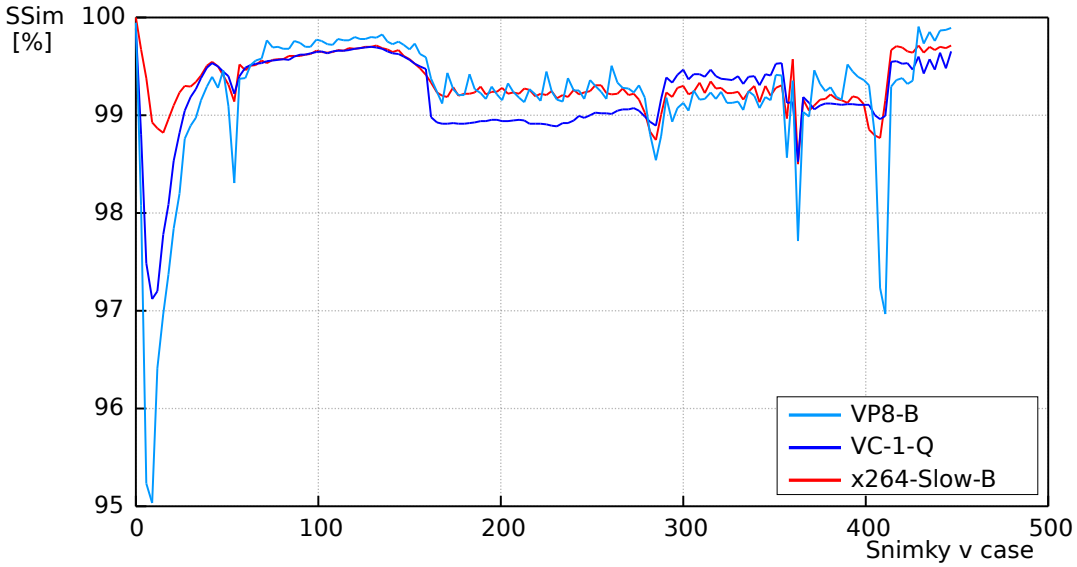
Tabulka 5.1: Výsledky metody BD pro PSNR a SSim – `composition`

V grafu 5.4 dosahuje nejlepších výsledků pro všechny úrovně datového toku x264-Slow-B

(dále jen x264). Jen o 11% horší je x264-Slow-Q. VP8 se přiblížilo kodeku x264 také, rozdíl je 35%. Nejhůře v této části dopadl VC-1-Q, jehož datový tok musel být o 76% větší, aby dosáhl stejné kvality jako x264. Výsledky metody SSim 5.5 jsou podobné jak u x264-Slow-Q, tak u VC-1-Q. VP8 se přiblížilo x264 na rozdíl pouhých 23%. Na základě těchto výsledků byl jako referenční kodek vybrán x264 s nastavením datového toku.



Graf 5.6: Průběh PSNR



Graf 5.7: Průběh SSim

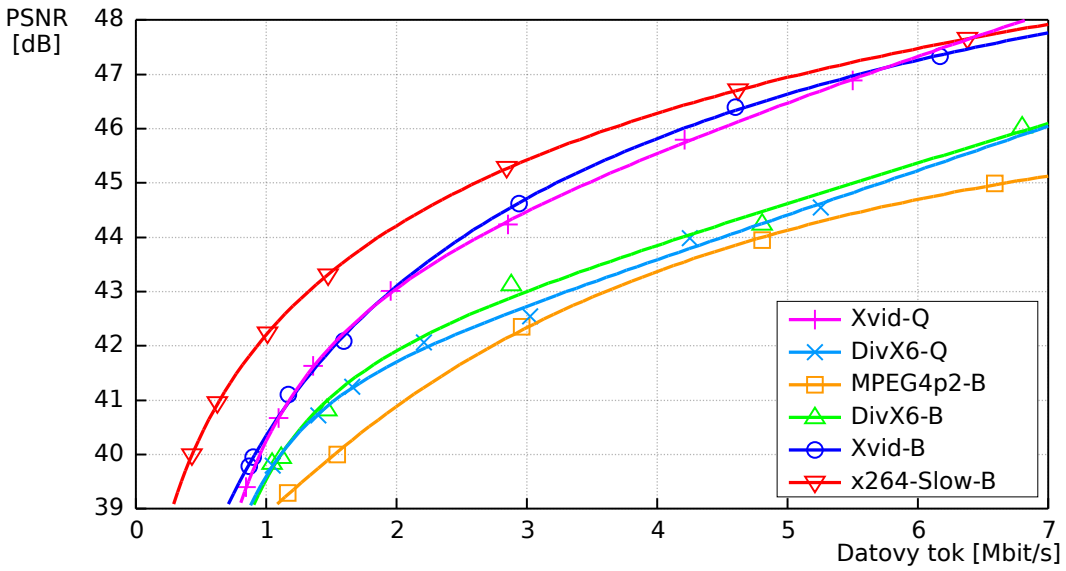
Na grafech průběhu PSNR 5.6 a SSim 5.7 lze snadno rozpoznat změny scén (skoky v úrovni PSNR/SSim). Malé výkyvy nastaly v situacích, kdy byl přechod plynulý nebo skokový. V místech, kde bylo použito přechodů do/z černé nebo záblesků (zvýšení jasu), nastaly značné výkyvy v měřené kvalitě. U VP8 lze pozorovat pravidelné změny kvality zhruba každých deset snímků.

Nastavení kodeků:

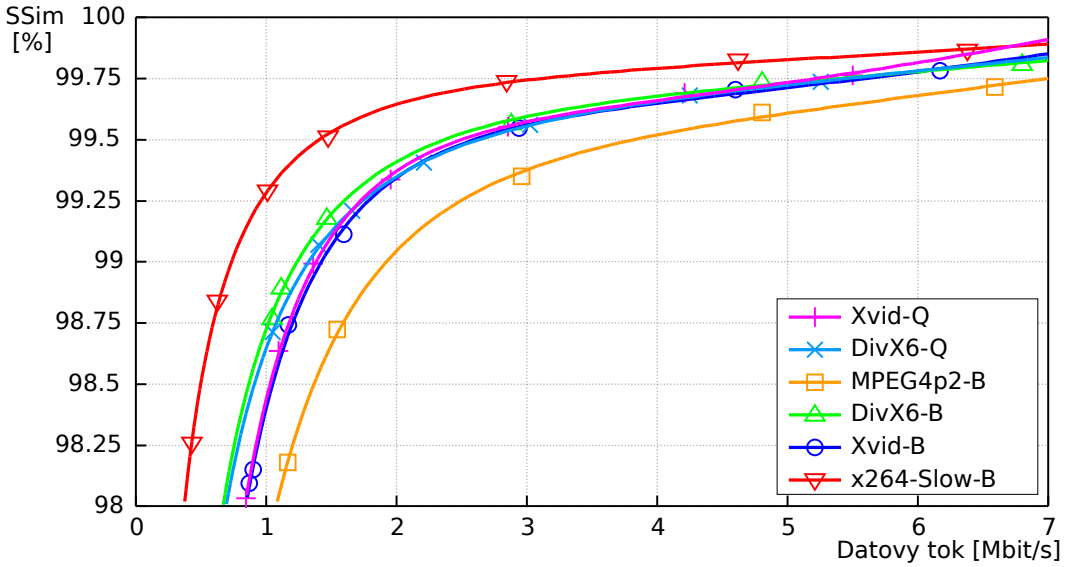
- **x264-Slow-Q**: profile: main, encoding preset: slow, single pass – quantizer-based: [22; 25; 27; 30; 35; 40]
- **x264-Slow-B**: profile: main, encoding preset: slow, single pass – bitarte-based: [400k; 600k; 1M; 1,5M; 3M; 5M; 7M]
- **VP8-B: profile**: profile: unrestricted, quality preset: best, two pass – bitrate-based: [400k; 600k; 1M; 1,5M; 3M; 5M; 7M]
- **VC-1-Q: profile**: WMV9 Advanced Profile, single pass – quality-based, VBR Quality: [10; 40; 60; 70; 80; 85; 88]

5.6.2 Xvid, DivX, MPEG-4 Part2 (libavcodec)

Do druhé skupiny testů byly zařazeny kodeky vycházející ze standardu MPEG-4 Part 2. Oproti referenčnímu x264 lze očekávat potřebu okolo 200% datového toku pro dosažení stejné úrovně kvality.



Graf 5.8: Proložení PSNR

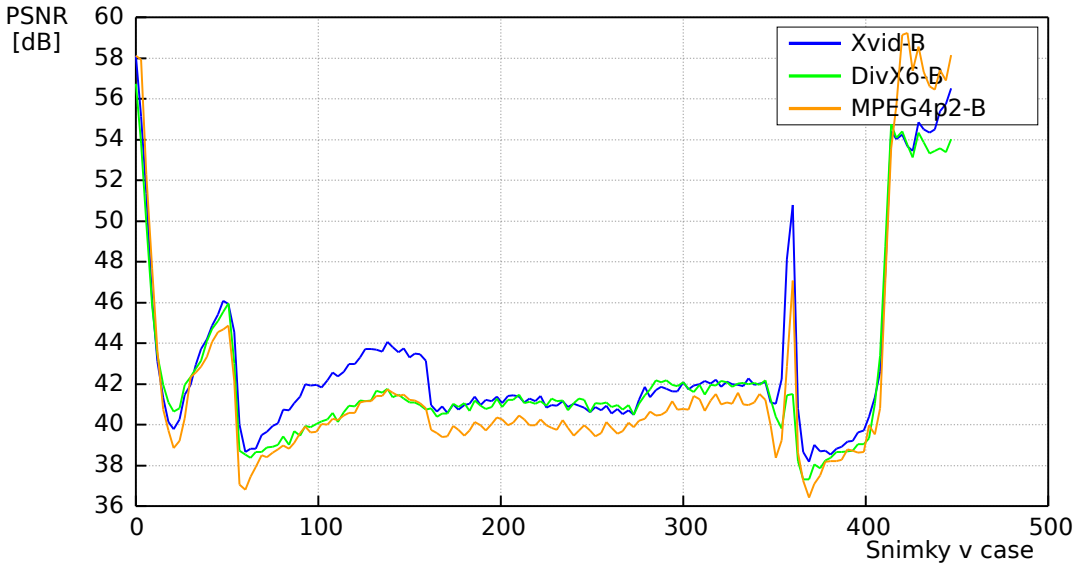


Graf 5.9: Proložení SSim

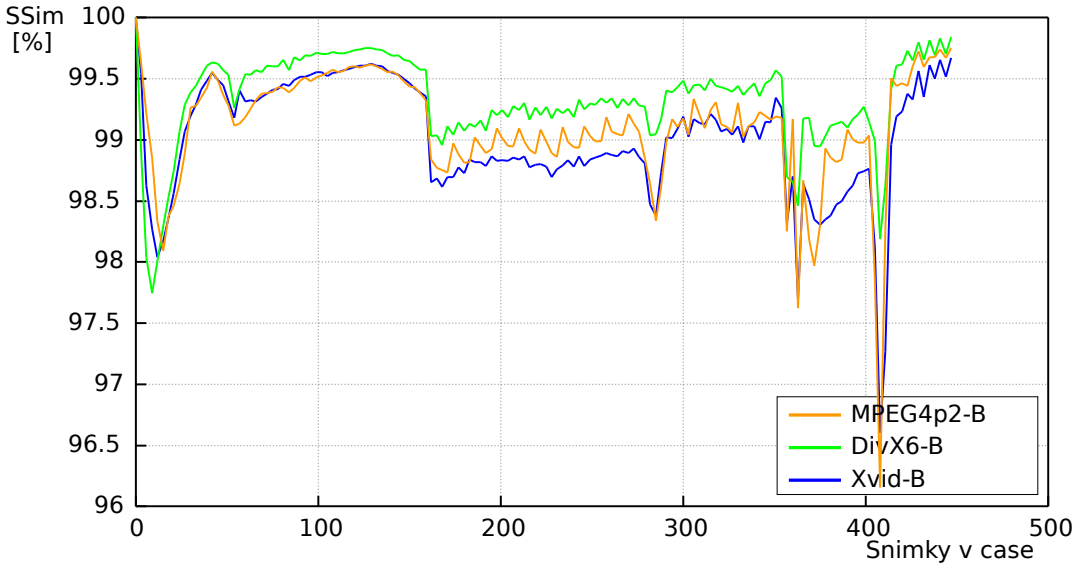
	Velikost datového toku v poměru k x264-Slow-B (100[%])				
Metrika	Xvid-B	Xvid-Q	DivX6-B	DivX6-Q	MPEG4p2-B
PSNR	127,9	137,9	215,1	230,1	262,6
SSim	187,6	186,6	169,1	178,1	268,0

Tabulka 5.2: Výsledky metody BD pro PSNR a SSim – `composition`

Hned na začátek je potřeba zdůraznit, že Xvid se v obou konfiguracích v metrice PSNR velmi přiblížil x264, což je pro kodek standardu MPEG-4 Part 2: ASP vynikající výsledek. V oblasti SSim nedosahuje tak dobrých výsledků, stále se však jedná o nadprůměrné hodnoty. Co se týká DivX 6, výsledky BD-PSNR pohybující se přes 200% nelze považovat za dobré, avšak v metrice SSim dosahuje lepších výsledků než Xvid. Na posledním místě se jak v PSNR tak SSim umístil libavcodec, který potřebuje téměř trojnásobný datový tok pro dosažení stejné úrovně kvality jako x264. Dalším zjištěným faktem je, že nastavení kodeku pomocí datového toku (B) dosahuje lepších výsledků než nastavení pomocí kvantizéru (Q), stejně jako v první skupině kodeků.



Graf 5.10: Průběh PSNR



Graf 5.11: Průběh SSim

Co se týká průběhů PSNR 5.10 a SSim 5.11 je na první pohled znát podstatně větší poklesy na přechodech scén. U libavcodecu lze pozorovat podobné kolísání jako u VP8. Za zmínku také stojí to,

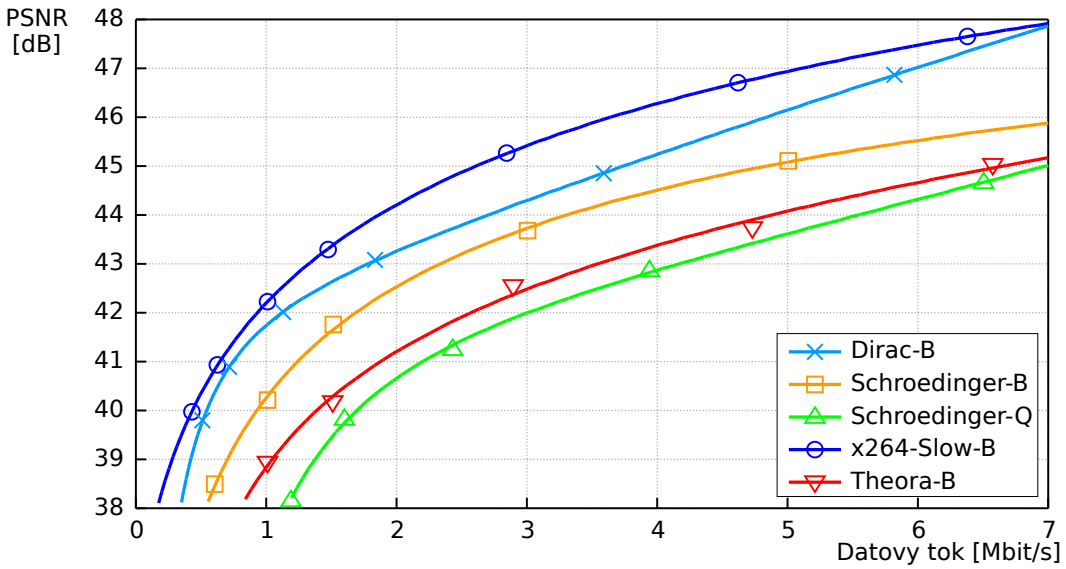
Nastavení kodeků:

- **Xvid-Q**: profile: Xvid HD 1080, quality preset: general purpouse, single pass – quantizer-based: [4; 5; 7; 10; 15; 20; 30]
- **Xvid-B**: profile: Xvid HD 1080, quality preset: general purpouse, single pass – bitarte-based: [400k; 600k; 1M; 1,5M; 3M; 5M; 7M]
- **DivX6-B**: profile: 1080HD, encoding preset: 7, single pass – bitarte-based: [400k; 600k; 1M; 1,5M; 3M; 5M; 7M]

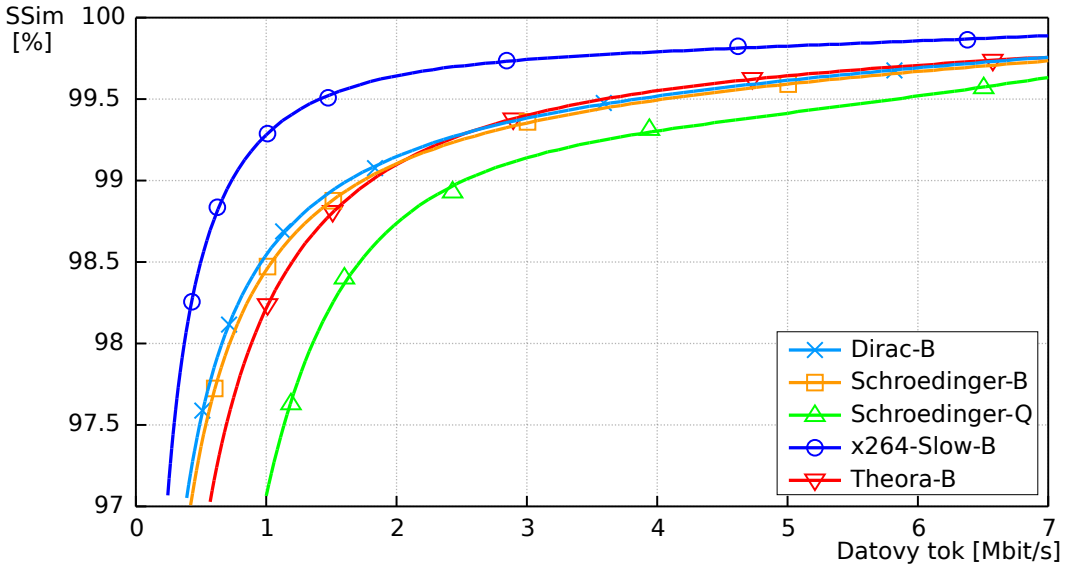
- **DivX6-Q**: profile: 1080HD, encoding preset: 7, single pass – quality-based: [4; 5; 6; 8; 10; 13; 15; 20]
- **MPEG4p2-B**: single pass - bitrate–based: [1M; 1,5M; 3M; 5M; 7M]

5.6.3 Dirac, Schrödinger, Theora

Základní testovací sestavu uzavírá trojice Dirac, Schrödinger a Theora. Schrödinger je implementací standardu Dirac, tudíž tyto dva kodeky by měly dosahovat podobných výsledků, od ostatních kodeků se zásadně liší v použité transformaci – využívají DWT nikoli DCT jako drtivá většina kodeků. Theora pracuje na podobném principu jako MPEG kodeky, lze tedy očekávat výsledky okolo 200% datového toku x264.



Graf 5.12: Proložení PSNR

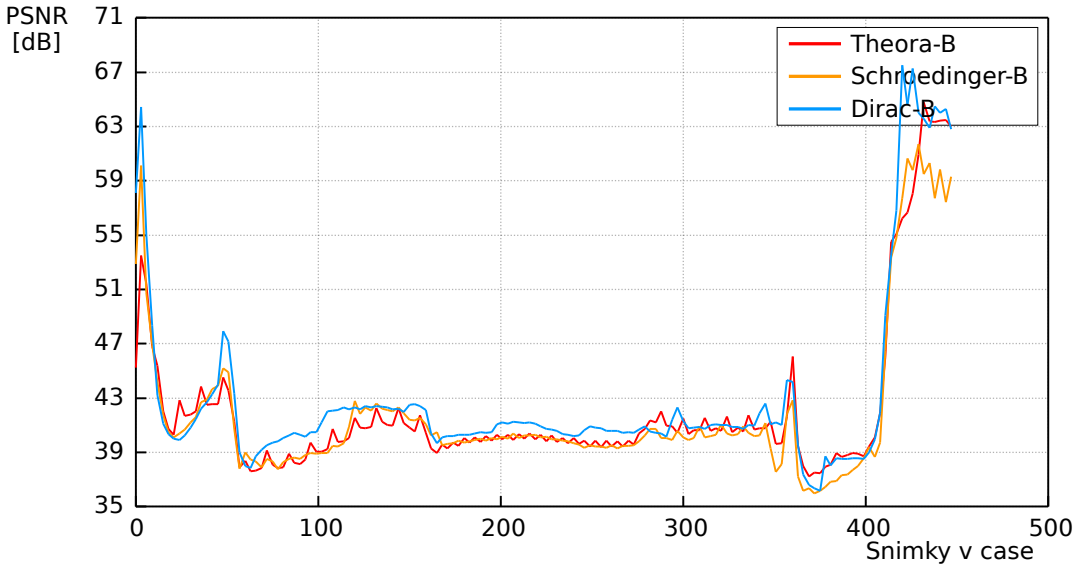


Graf 5.13: Proložení SSim

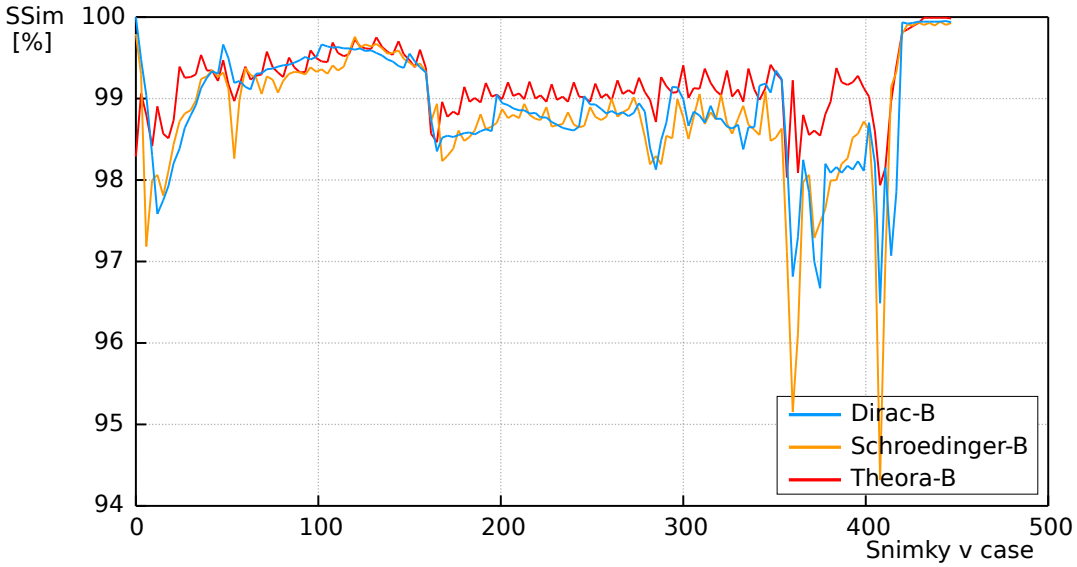
	Velikost datového toku v poměru k x264-Slow-B (100[%])			
Metrika	Dirac-B	Schrödinger-B	Schrödinger-Q	Theora-B
PSNR	132,9	179,6	306,1	271,8
SSim	236,5	244,7	353,1	246,5

Tabulka 5.3: Výsledky metody BD pro PSNR a SSim – `composition`

Naměřené hodnoty BD-PSNR jsou poměrně překvapivé u Diracu, rozdíl 32% je vynikající. V metrice SSim už nedosáhl zdaleka tak dobrých hodnot, nicméně jeho výsledky jsou stále značně nadprůměrné. Schrödinger-B dosáhl o něco horších výsledků, pořád se však řadí mezi průměr. Naopak nastavení pomocí kvantizéru dopadlo velice špatně, výsledky jsou v podstatě nejhorší v celém testu. Theora kódovala testovací video podprůměrně, o něco lépe než libavcodec.



Graf 5.14: Průběh PSNR



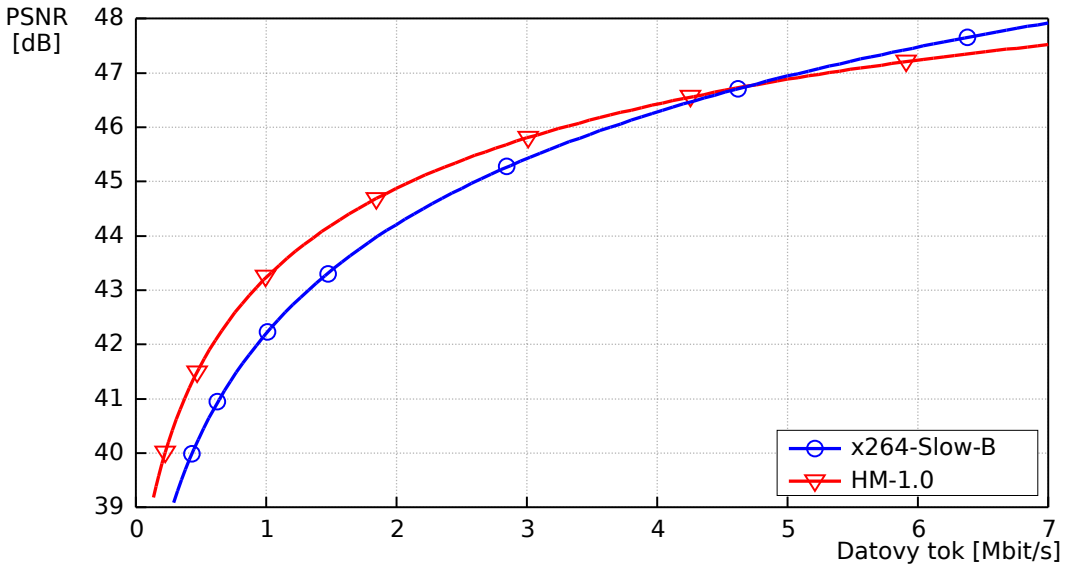
Graf 5.15: Průběh SSim

Opět lze pozorovat značné kolísání SSim i PSNR hodnot ve srovnání s referenčním x264. Největší propady nastaly u Schrödingeru.

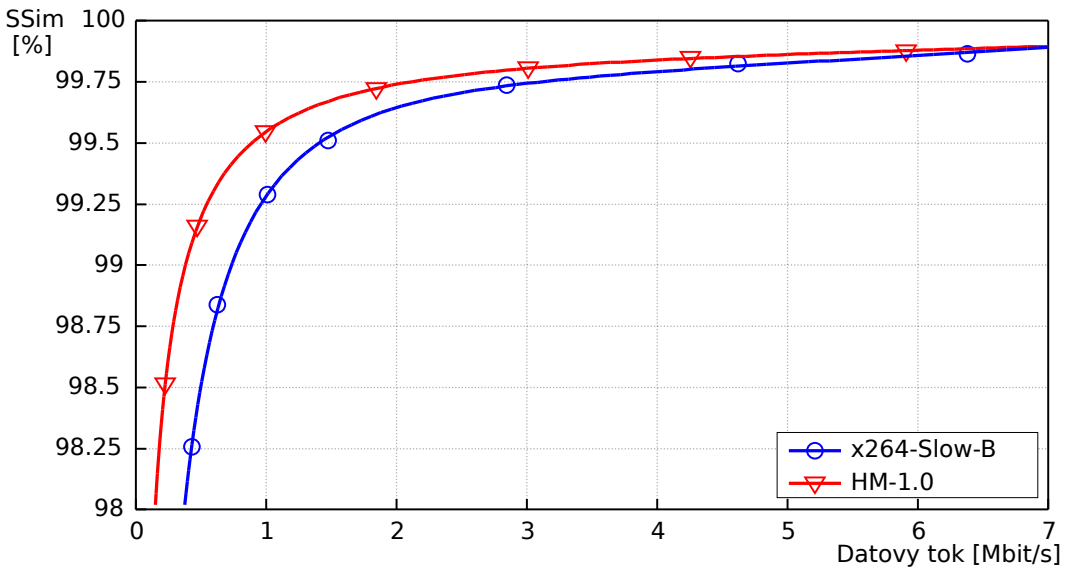
Nastavení kodeků:

- **Dirac-B**: implicit, single pass – bitarte-based: [400k; 600k; 1M; 1,5M; 3M; 5M; 7M]
- **Schrödinger-B**: implicit, single pass – bitarte-based: [400k; 600k; 1M; 1,5M; 3M; 5M; 7M]
- **Schrödinger-Q**: implicit, single pass – quality-based: [4; 5; 6; 8; 10; 13; 15; 20]
- **Theora-B**: implicit, single pass – bitarte-based: [400k; 600k; 1M; 1,5M; 3M; 5M; 7M]

5.6.4 HM 1.0



Graf 5.16: Proložení PSNR

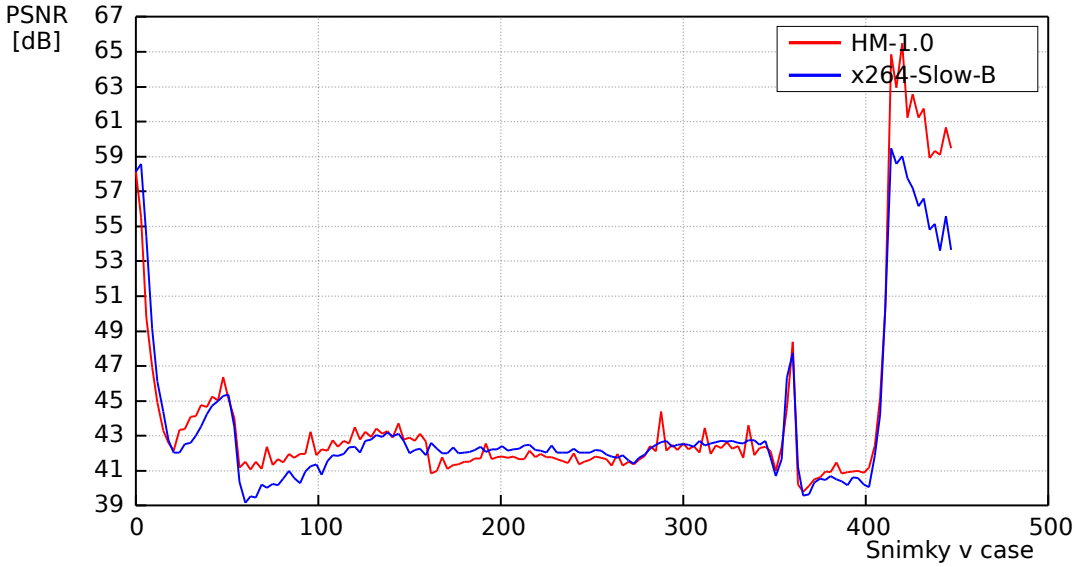


Graf 5.17: Proložení SSim

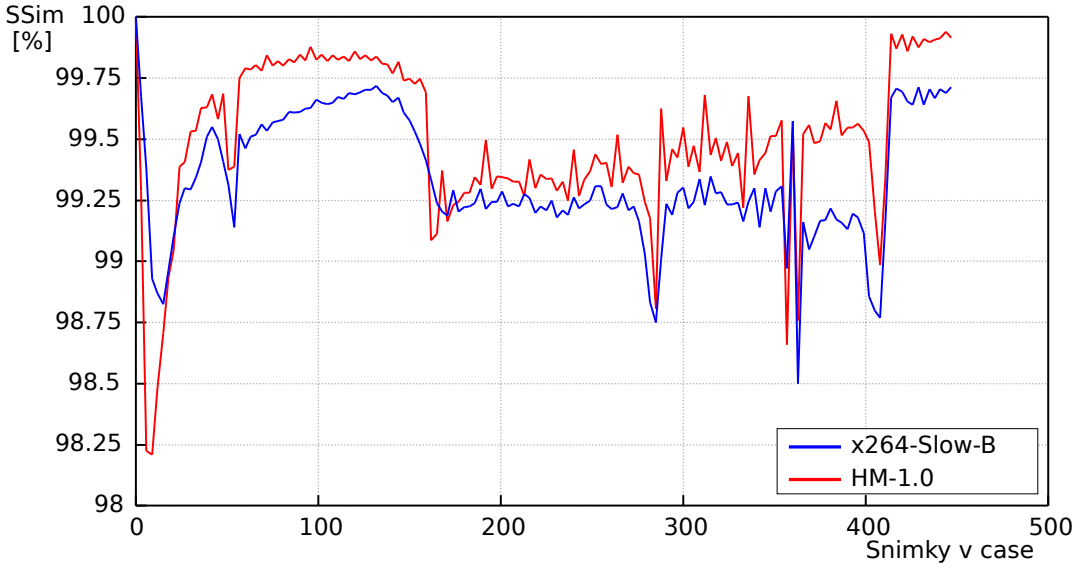
	Velikost datového toku v poměru k x264-Slow-B (100[%])
Metrika	HM 1.0
PSNR	82,4
SSim	63,2

Tabulka 5.4: Výsledky metody BD pro PSNR a SSim – `composition`

Srovnání implementace budoucího standardu H.265 a nejlepší současné implementace H.264 poskytlo zajímavé výsledky. Zatímco v metrice PSNR došlo ke snížení datového toku „pouze“ o 20%, výsledky BD-SSim – pokles na 60% datového toku x264 – svědčí o potenciálu budoucího H.265.



Graf 5.18: Průběh PSNR

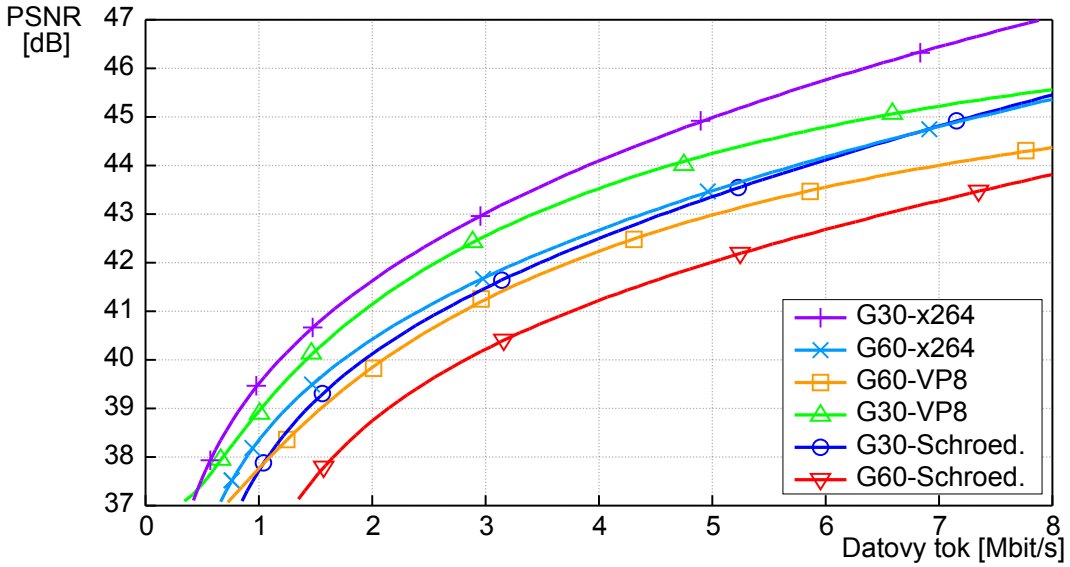


Graf 5.19: Průběh SSim

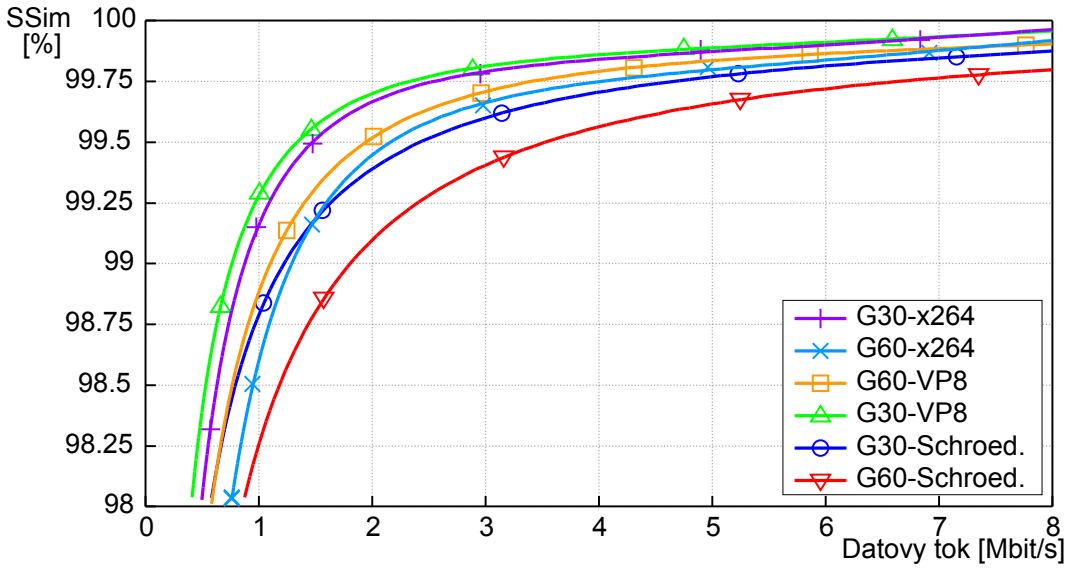
5.6.5 Vztah fps – datový tok

Pakliže je potřeba datový tok b pro dosažení kvality q, jaký bude potřeba datový tok pro dosažení stejné kvality q při dvojnásobné snímkové frekvenci? Primitivní odpověď by byla, že pro dvojnásobný počet snímků je potřeba dvojnásobný datový tok. To by platilo možná v případě, že video by bylo pouze intra kódované. Testované kodeky používají také inter snímkové kódování. Úvaha za tímto testem je následující: při vyšším fps budou změny mezi snímky menší, tím pádem inter kódování by mohlo být efektivnější, a nemuselo by být potřeba dvojnásobný datový tok pro dosažení stejné kvality.

Pro toto srovnání byla použita sekvence `game`, která obsahuje velmi rychlý pohyb v obraze a *motion blur*. Do testu byly zařazeny tyto kodeky: x264, VP8 a Schrödinger.



Graf 5.20: Proložení PSNR



Graf 5.21: Proložení SSim

	Velikost datového toku 60 fps v poměru k 30 fps [%]		
Metrika	x264	VP8	Schrödinger
PSNR	145.7	143.8	140.5
SSim	149.6	141.0	142.5

Tabulka 5.5: Srovnání vlivu fps na potřebný datový tok – `composition`

Z tabulky 5.5 jasně vyplývá, že pro zakódování videa v 60 fps v porovnání s 30fps není potřeba dvojnásobný datový tok, nýbrž pouze 40–50% navíc. V případě, že bychom chtěli ušetřit tím, že dramaticky snížíme počet snímků za sekundu, je pravděpodobné, že nedosáhneme požadované kvality, případně budeme nuceni snížit datový tok jen částečně.

5.6.6 Výkon

Výkon, nebo také rychlost kodeků může být v mnoha situacích klíčovým elementem. Měření rychlosti kódování však v tomto případě nelze považovat za zcela korektní, protože byla použita různá prostředí (Windows XP, Ubuntu 10) a různé nástroje pro kódování videí (FFmpeg, VirtualDub, WMNicEnc, v závorce za hodnotou je uvedeno jaký nástroj byl použit ke kódování). Naopak dekódování bylo realizováno zcela v režii FFmpegu se třemi opakováními, pomocí příkazu `ffmpeg -i <video> -an -vcodec rawvideo -f rawvideo -y /dev/null`. Hodnoty v tabulce 5.6 jsou zaokrouhlené průměry všech získaných hodnot fps. U jednotlivých konfigurací nebylo zakazováno ani vynucováno vícevláknové zpracování, řízení bylo ponecháno na kodeku.

	Rychlost [fps] – `composition`	
Kodek	Kódování	Dekódování
MPEG4p2-B	56 (ff)	326
DivX-B/Q	26/29 (vd)	312/319
Xvid-B/Q	36/34 (vd)	286/271
Theora-B	30 (ff)	266
VC-1-Q	25 (wmne)	263
VP8-B	11 (ff)	149
x264-B/Q	16/20 (ff)	133/128
Schrödinger-B/Q	23/22 (ff)	104/103
Dirac-B	5 (ff)	38

Tabulka 5.6: Výkon kodeků

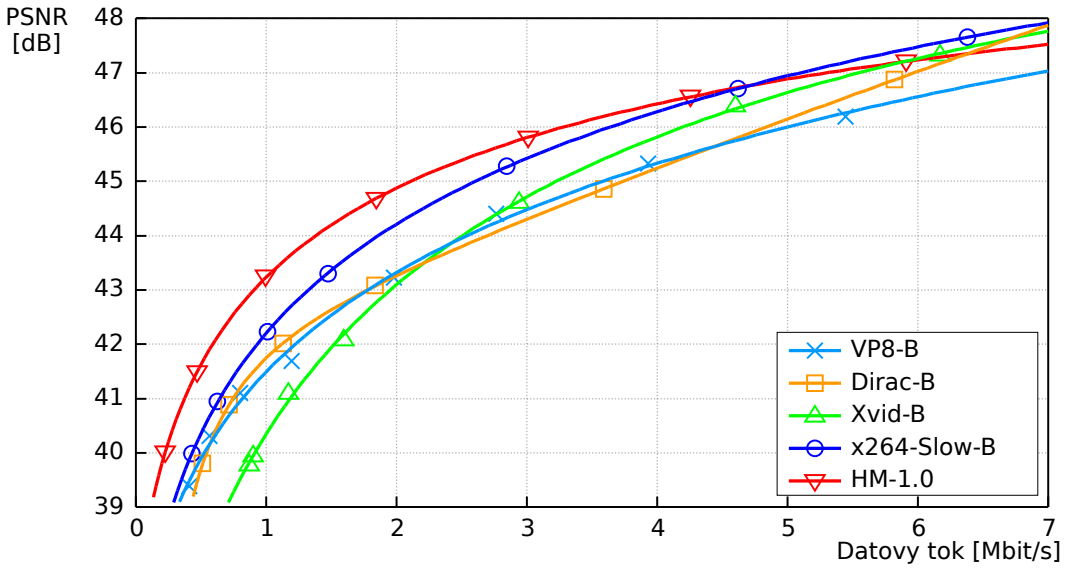
Z výsledků je patrné, že nejrychlejší kodeky jsou založeny na standardu MPEG-4 Part 2, konkrétně verze libavcodec vynikla v rychlosti kódování i dekódování, šlo by uvažovat o nasazení v real-time aplikacích, popřípadně v zařízeních s malým výkonem. Theora a VC-1 také dosahují dobrých výsledků, především v dekódování. Naopak Dirac se projevil jako velice pomalý kodek, jehož výsledky jsou zlomkem ostatních kodeků. „High performance“ implementace Diracu Schrödinger však dosahuje podstatně lepších výsledků a staví se na úroveň x264. Speciální případ je VP8, který dosahuje průměrné rychlosti dekódování, ale druhého nejhoršího výsledku v kódování. Vzhledem k jeho stáří lze očekávat další optimalizace a zrychlení.

Do tabulky nebyl zařazen kodek HM 1.0, jeho rychlost kódování i dekódování je řádově nižší než u ostatních kodeků. Je to dáno několika fakty, především se jedná o testovací verzi, která nepodporuje více vláken, pravděpodobně ani není zatím optimalizovaná a současně provádí nejsložitější výpočty ze všech testovaných kodeků.

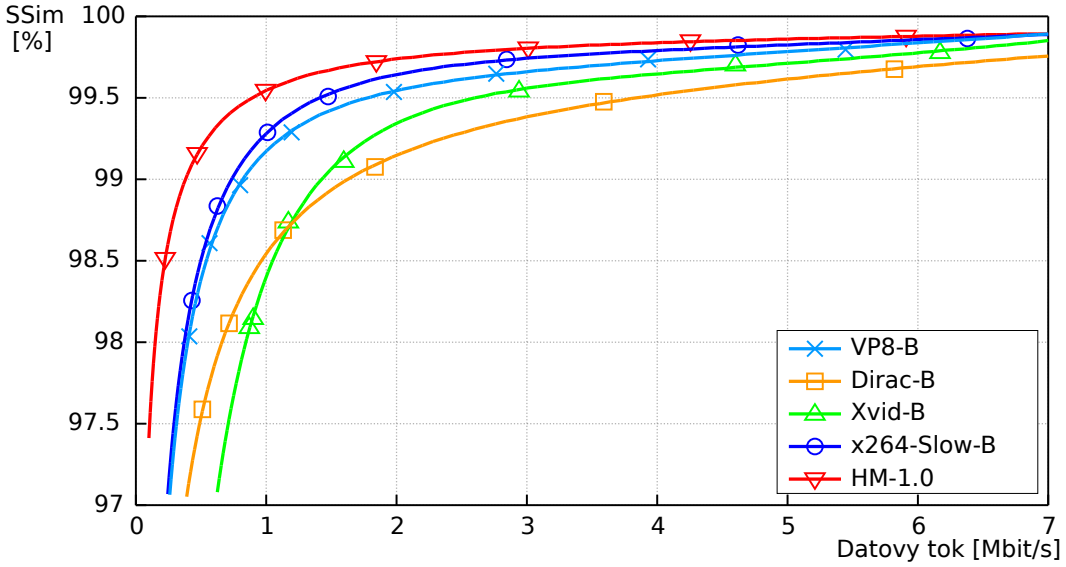
Toto srovnání je nutné brát s nadhledem, v současné době je součástí mnoha zařízení (od grafických karet až po mobilní telefony) hardwarová akcelerace základních formátů. Je proto možné, že x264 bude přehráváno plynuleji a s menší spotřebou než např. Theora, která dosáhla dvojnásobného dekódovacího výkonu, nicméně nemá v daném zařízení HW podporu.

5.6.7 Vyhodnocení výsledků srovnání

Do závěrečného srovnání ztrátových kodeků byly vybráni zástupci z každé kategorie, do testu byl zařazen i HM 1.0.



Graf 5.22: Proložení PSNR



Graf 5.23: Proložení SSim

V této kapitole bylo představeno srovnání současných i budoucích ztrátových videokodeků. Byly představeny techniky srovnání, používané nástroje i testovací sekvence. Ze samotných testů 5.7 vyplynulo, že nejlepším současným kodekem je x264. Jeho výsledky byly co se týče

Obrázek 5.24: Výřezy z komprimovaných sekvencí `composition` – dynamická scéna (kouř lokomotivy); zleva a shora: původní, x264, VC-1; VP8, HM 1.0, Theora; Dirac, Xvid, libavcodec

poměru kvalita/komprese špičkové, výkon kodeku se pohyboval v přijatelných hodnotách. Pro jeho použití svědčí rozšířená podpora jak v SW tak HW. Jediný kodek, který v testech překonal x264 je HM 1.0. Snížení potřebného datového toku přibližně o jednu třetinu lze považovat za dobrý výsledek, s ohledem na to, že se jedná pouze o testovací implementaci plánovaného standardu. Další kodek, který si v testech vedl velmi dobře je VP8. Jedná se o relativně „nový“ kodek, přesto jeho poměr kvality/komprese jen lehce zaostává za x264. Rychlost kódování byla nízká, lze očekávat další optimalizace. Kodeky založené na standardu MPEG-4 Part 2 vynikly v oblasti výkonu, Xvid dosáhl i poměrně dobrých výsledků v testech kvality. Kodeky založené na diskrétní vlnkové transformaci poskytly také slušné výsledky. Kodek DivX 6 se ukázal jako průměrný stejně jako VC-1. Nejhůře ve srovnání dopadly kodeky libavcodec a Theora, které vyžadovaly přibližně 2,5násobek datového toku, aby dosáhly stejné kvality jako x264.

	BD [%]		Rychlost [fps]	
Kodek	PSNR	SSim	Kódování	Dekódování
HM-1.0	82,4	63,2	<1	<1
x264-B	100,0	100,0	16	133
x264-Q	111,4	112,2	20	128
VP8-B	135,5	123,1	11	149
Xvid-B	127,9	187,6	36	286
Xvid-Q	137,9	186,6	34	271
Dirac-B	132,9	236,5	5	38
VC-1-Q	176,7	171,5	25	263
DivX-B	215,1	169,1	26	312
DivX-Q	230,1	178,1	29	319
Schrödinger-B	179,6	244,7	23	104
Theora-B	271,8	246,5	30	266
MPEG4p2-B	262,6	268,0	56	326
Schrödinger-Q	306,1	353,1	22	103

Tabulka 5.7: Celkové srovnání kodeků – `composition`

5.7 Srovnání bezeztrátových kodeků

Úkolem bezeztrátových kodeků je komprese, při které nedojde ke ztrátě kvality. Toto tvrzení není ovšem úplně přesné, většina bezeztrátových kodeků umožňuje kódovat do různých barevných prostorů, což v některých případech znamená ztrátu části obrazové informace. Z hlediska využití takovýchto kodeků je však tato možnost často vítána (není důvod uchovávat informaci o barvě, která stejně nakonec nebude využita – oblast předzpracování videa při editaci atd.). Z tohoto faktu vyplývá i základní konfigurace testů, kde jsou vlastnosti bezeztrátových kodeků testovány v barevných prostorech RGB24 a YUV. Protože se jedná o bezeztrátové kodeky, odpadá potřeba měřit kvalitu (PSNR, SSim). Naopak klíčovými faktory jsou zde kompresní poměr a rychlost kódování.

Konfigurace testů

Samotné zakódování bylo provedeno pomocí aplikace VirtualDub 1.9.11 (ffmpeg neumožňuje kódovat do formátu MSU). Verze testovaných kodeků jsou:

- HuffYUV v2.1.1 - CCESP Patch v0.2.5
- Lagarith 1.3.20
- FFV1 jako součást ffdshow revize 3487
- MSU v0.6.0

Jako testovací video bylo použito základní `composition` o velikosti 1263,531 MB. Pro snížení vlivu rychlosti čtení a zápisu dat na disku byl použit SSD.

Kodek (RGB24)	Výsledná velikost [MB]	Rychlost kódování [fps]	Kompresní poměr
HuffYUV	710,173	50	1,77
Lagarith	523,819	45(31)	2,41
FFV1	484,157	11	2,60
MSU	497,451	2	2,54

Tabulka 5.8: Výsledky bezeztrátových kodeků v RGB24

Kodek (YUV)	Výsledná velikost [MB]	Rychlost kódování [fps]	Kompresní poměr
HuffYUV	392,274	76	3,22
Lagarith	249,600	38(27)	5,06
FFV1	230,961	17	5,47
MSU	216,806	4	5,82

Tabulka 5.9: Výsledky bezeztrátových kodeků v YUV

Vyhodnocení výsledků

Jak je patrné z tabulek 5.8 a 5.9 nejlepšího kompresního poměru dosahují kodeky FFV1 a MSU. Jejich rychlost kódování je ale vylučuje z použití v real–time aplikacích, především MSU dosahoval velmi nízkých hodnot fps. Jako vhodné uplatnění by připadly v úvahu aplikace vyžadující co největší kompresi, například archivace. Naopak kodek HuffYUV vykazoval v oblasti rychlosti kódování vysoké hodnoty. Kompresní poměr je však poměrně nízký v porovnání s ostatními kodeky. Lagarith dosahoval nadprůměrných výsledků jak v kompresním poměru tak v rychlosti (hodnoty fps v závorkách představují rychlost se zakázanou podporou pro více vláken), což ho předurčuje k univerzálnímu použití.

Kapitola 6

Závěr

Tuto práci lze rozdělit na tři základní části, teoretickou, implementační a srovnávací. V první, teoretické části, která tvoří značnou část tohoto textu byly postupně prezentovány informace o problematice kódování videa, standardů a kodeků. Byly popsány techniky kódování jako DCT, DWT, inter snímkové kódování nebo entropická kódování. Z rozebíraných standardů lze připomenout H.264/AVC, MPEG-4 Part 2, VP8, VC-1 a další. Následně byly popsány jednotlivé kodeky odpovídající vybraným standardům, především x264, DivX, Xvid, Theora, Dirac atd.

Druhá část spočívala v návrhu a implementaci srovnávacího nástroje. Tato část byla realizována v C++, k načítání vstupních videí byly použity knihovny FFmpeg, pro optimalizované operace nad maticemi a obrazem bylo využíváno OpenCV. K samotnému srovnání byly implementovány metody PSNR, SSim a BD-PSNR. Součástí výpočtů bylo použití polynomiální regrese a numerické metody výpočtu integrálů Monte Carlo. V neposlední řadě byla implementována podpora více-vláknového zpracování OpenMP, značně urychlující proces srovnání. Výstupem aplikace byly kromě textu i vektorové grafy v SVG prezentující naměřené a vypočítané hodnoty.

Srovnání kodeků byla třetí část, která zúročila poznatky z první části a plně využila implementaci z druhé části. Srovnání bylo provedeno po skupinách příbuzných kodeků s x264 jako referencí. Výsledky byly prezentovány formou grafů hodnot PSNR a SSim včetně proložení a pomocí grafů průběhu hodnot PSNR a SSim. Výsledky metody BD-PSNR byly prezentovány formou tabulek. Naměřené a vypočtené hodnoty stejně jako grafy byly slovně popsány a byly vyvozeny závěry.

Nejlepším současným kodekem je x264, následovaný VP8 a kodeky MPEG-4 Part 2. Kodeky založené na DWT poskytly dobré výsledky. Kodek HM 1.0 dosáhl až o třetinu lepší komprese než x264.

Cíle práce:

- Seznamte se se v současnosti používanými standardy ke kompresi videa a kodeky, které je implementují.
- Vyberte významné standardy a podrobně je popište (postup využívaný ke kódování videa). Vyberte a popište k nim také významné kodeky.
- Formáty a kodeky srovnejte. U kodeků především kompresní poměr a výkon. Srovnání rozdělte na ztrátové a bezeztrátové formáty.
- Ze srovnání vyvoďte závěry, diskutujte nad vhodností využití jednotlivých formátů a kodeků k různým účelům.

Tyto základní cíle práce považuji za splněné, alespoň na úrovni bakalářské práce. Jeden upřesňující požadavek se nepodařilo splnit (problém implementace), a to srovnání kodeků na

úrovni jednotlivých snímků. Naopak byl zahrnut a srovnáván budoucí standard H.265, resp. jeho implementace HM 1.0. Dále byla taky implementována a použita metoda BD-PSNR, která vede ke značnému zpřehlednění a zobecnění výsledků.

Část práce byla též zařazena do soutěže EEICT, byla zařazena do sborníku a postoupila do finále (nebylo dosaženo vítězných pozic).

Literatura

[1] IIT Kharagpur NPTEL Course, Prof. Sabyasachi Sengupta, lekce 16–26. `http://nptel.iitm.ac.in/video.php?subjectId=117105081`.

[2] Inside WebM Technology: The VP8 Alternate Reference Frame. `http://blog.webmproject.org/2010/05/inside-webm-technology-vp8-alternate.html`.

[3] ISO/IEC 14496-10:2010 Information technology - Coding of audio-visual objects - Part 2: Visual, Second edition.

[4] ISO/IEC 14496-2:2001(E) Advanced video coding for generic audiovisual services.

[5] MPEG-4 – The Media Standard: The landscape of advanced multimedia coding. The MPEG-4 Industry Forum, Březen 2002.

[6] BSD License Definition. `http://www.linfo.org/bsdlicense.html`, 2004-04-19 [cit. 2011-01-10].

[7] Dirac Specification. `http://diracvideo.org/download/specification/dirac-spec-latest.pdf`, Září 2008.

[8] OpenMP Application Program Interface. `http://www.openmp.org/mp-documents/spec30.pdf`, Květen 2008.

[9] Theora Specification. `www.theora.org/doc/Theora.pdf`, Březen 2011.

[10] Balan, V.; Condea, C.: *Wavelets and Image Compression*. Telecommunication Standardization Sector of ITU, Leden 2003.

[11] Bjontegaard, G.: *Calculation of average PSNR differences between RD-curves*. International Telecommunication Union, Video Coding Experts Group, Duben 2001.

[12] Bjontegaard, G.: *Improvements of the BD-PSNR model*. International Telecommunication Union, Červenec 2008.

[13] de Haan, G.; Bellers, E. B.: De-interlacing – an overview. *Proceedings of the IEEE*, ročník 86, č. 9, Září 1998.

[14] ISO/IEC: *Information technology – Coding of audio-visual objects – Part 10: Advanced Video Coding*. International Organization for Standardization, 2004.

[15] ISO/IEC, I.: *Test Model under Consideration*. Joint Collaborative Team on Video Coding, Červenec 2010.

[16] ITU-R BT.601: *Studio encoding parameters of digital television for standard 4:3 and wide-screen 16:9 aspect ratios*. ITU, 2007.

[17] ITU-T: *Objective perceptual multimedia video quality measurement in the presence of a full reference*. Telecommunication Standardization Sector of ITU, Srpen 2008.

[18] Khayam, S. A.: *The Discrete Cosine Transform (DCT): Theory and Application*. Department of Electrical and Computer Engineering, Michigan State University, 2003.

[19] Knee, M.: *A Single-ended Picture Quality Measure for MPEG-2*. Snell and Wilcox, UK, 2010.

[20] Kwon, S.; A. Tamhankar and, K. R.: *Overview of H.264/MPEG-4 part 10*. Srpen 2005.

[21] Marpe, D.; Wiegand, T.; Sullivan, G. J.: *The H.264/MPEG4 Advanced Video Coding Standard and its Applications*. IEEE, New York, USA, 2006.

[22] Ohwovoriole, E.; Andreopoulos, Y.: *Rate-Disortion Performance of Contemporary Video Codecs: Comparison of Google/WebM VP8, AVC H.264, HEVC TMuC*. 2010.

[23] Richardson, E. G.: *H.264 and MPEG-4 Video Compression: Video Coding for Next-generation Multimedia*. Iain, Říjen 2003.

[24] Said, A.: *Introduction to Arithmetic Coding – Theory and Practice*. Duben 2004.

[25] Segall, C.; Katsaggelos, A.: Pre– and Post–Processing Algorithms for Compressed Video Enhancement. In *Signals, Systems and Computers*, 2000, s. 1369–1373.

[26] Sikora, T.: *Digital Video Coding Standards and Their Role in Video Communications*. IOS Press, 1999.

[27] Sullivan, G. J.: *DirectX Video Acceleration Specification for Windows Media Video v8, v9 and vA Decoding (Including SMPTE 421M VC-1)*. Microsoft Corporation, 2007.

[28] Sullivan, G. J.; Wiegand, T.: *Video Compression – From Concepts to the H.264/AVC Standard*. Červenec 2004.

[29] Taubman, D.: High Performance Scalable Image Compression with EBCOT. *IEEE Transactions on Image Processing*, ročník 9, č. 7, Červenec 2000.

[30] Valens, C.: *Embedded Zerotree Wavelet Encoding*. 1999.

[31] Wang, Z.; Bovik, A. C.; Sheikh, H. R.; aj.: Image Quality Assessment: From Error Visibility to Structural Similarity. *IEEE Transactions on Image Processing*, ročník 13, č. 4, Duben 2004.

[32] WebM Project: *VP8 Data Format and Decoding Guide*. Google On2, New York, USA, 2010.

[33] Wiegand, T.; Sullivan, G. J.; Bjontegaard, G.; aj.: Overview of the H.264/AVC Video Coding Standard. *IEEE Transactions on Circuits and Systems for Video Technology*, ročník 13, č. 7, Červenec 2003.

[34] WWW stránky: H265.net. `http://www.h265.net/`.