VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ
ÚSTAV TELEKOMUNIKACÍ

FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION
DEPARTMENT OF TELECOMMUNICATIONS

ZTRÁTOVÁ KOMPRESE POHYBLIVÝCH OBRAZŮ

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

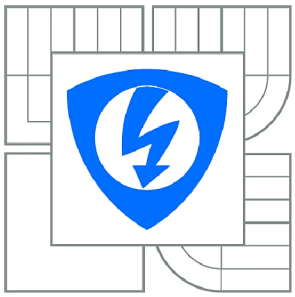
AUTOR PRÁCE
AUTHOR

Bc. MICHAL ŠIŠKA

BRNO 2011

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ

ÚSTAV TELEKOMUNIKACÍ

FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION

DEPARTMENT OF TELECOMMUNICATIONS

ZTRÁTOVÁ KOMPRESE POHYBLIVÝCH OBRAZŮ

LOSSY VIDEO COMPRESSION

DIPLOMOVÁ PRÁCE

MASTER'S THESIS

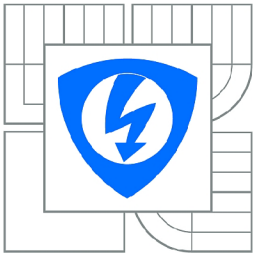
AUTOR PRÁCE — Bc. MICHAL ŠIŠKA

AUTHOR

VEDOUCÍ PRÁCE — Ing. JIŘÍ SCHIMMEL, Ph.D.

SUPERVISOR

BRNO 2011



VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

Fakulta elektrotechniky a komunikačních technologií

Ústav telekomunikací

Diplomová práce

magisterský navazující studijní obor
Telekomunikační a informační technika

Student:	Bc. Michal Šiška	***ID:***	100290
Ročník:	2	***Akademický rok:***	2010/2011

NÁZEV TÉMATU:

Ztrátová komprese pohyblivých obrazů

POKYNY PRO VYPRACOVÁNÍ:

Prostudujte metody ztrátové komprese videosignálu podle standardů ITU H.26x a doporučení MPEG. V programovacím jazyku Java realizujte program, který bude demonstrovat jednotlivé kroky kódování videosignálu podle standardu MPEG až po vstup bloku kódování vzorků pomocí VLC. Vstupem programu bude sekvence obrázků uložená jako bitové mapy ve formátu RGB. Zaměřte se zejména na blok podvzorkování, kvantizaci a kompenzaci pohybu. Program bude zobrazovat vstupy a výstupy jednotlivých bloků kodéru během procesu kódování, zejména vstupní snímek kodéru, paměť snímků, rozdílový snímek, mapu vektorů, rekonstruovaný snímek a chybový snímek. Obdobný program realizujte pro dekodér, opět počínaje blokem obnovení vzorků dekodérem VLD.

DOPORUČENÁ LITERATURA:

[1] I. E.G. Richardson, H.264 and MPEG-4 Video Compression. Wiley, 2003. ISBN 978-0-470-84837-1
[2] L. Hanzo, P. Cherriman,, J. Streit, Video Compression and Communications, 2nd ed. John Wiley & Sons, Ltd, 2007. ISBN 978-0-470-51849-6
[3] M. Ghanbari, Image Compression to Advanced Video Coding. Institution Electrical Engineers; 2003; 430s; ISBN 0852967101
[4] ITU-T Recommendation H.263 Annex O; 1998

Termín zadání:	7.2.2011	***Termín odevzdání:***	26.5.2011

Vedoucí práce: Ing. Jiří Schimmel, Ph.D.

prof. Ing. Kamil Vrba, CSc.
Předseda oborové rady

UPOZORNĚNÍ:

ABSTRAKT

Tato diplomová práce se zabývá popisem ztrátové komprese pohyblivých obrazů. V kapitolách jsou popsány základní informace o kompresi obrazu a dále jednotlivé standardy pro ztrátovou i bezeztrátovou kompresi statických i pohyblivých obrazů. Praktická část popisuje program realizovaný v jazyce Java, který simuluje funkce kodeku MPEG.

KLÍČOVÁ SLOVA

Ztrátové kódování obrazu, kompenzace pohybu, vektory pohybu, HVS, kodek, JPEG, MPEG, ITU H.264, Java.

ABSTRACT

This thesis deals with description of lossy video compression. Theoretical part of the work describes the fundamentals of the video compression and standarts for lossy as well lossless video and still image compression. The practical part follows up with design of Java program for simulation of MPEG codec.

KEYWORDS

Lossy Video Compression, Motion Compensation, Motion Vectors, HVS, codec, JPEG, MPEG, ITU H.264, Java.

ŠIŠKA, Michal *Ztrátová komprese pohyblivých obrazů*: diplomová práce. Brno: Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, Ústav telekomunikací, 2010. 65 s. Vedoucí práce byl Ing. Jiří Schimmel, Ph.D.

PROHLÁŠENÍ

Prohlašuji, že svou diplomovou práci na téma „Ztrátová komprese pohyblivých obrazů“ jsem vypracoval samostatně pod vedením vedoucího diplomové práce a s použitím odborné literatury a dalších informačních zdrojů, které jsou všechny citovány v práci a uvedeny v seznamu literatury na konci práce.

Jako autor uvedené diplomové práce dále prohlašuji, že v souvislosti s vytvořením této diplomové práce jsem neporušil autorská práva třetích osob, zejména jsem nezasáhl nedovoleným způsobem do cizích autorských práv osobnostních a jsem si plně vědom následků porušení ustanovení § 11 a následujících autorského zákona č. 121/2000 Sb., včetně možných trestněprávních důsledků vyplývajících z ustanovení § 152 trestního zákona č. 140/1961 Sb.

Brno ....

(podpis autora)

PODĚKOVÁNÍ

Děkuji vedoucímu diplomové práce Ing. Jiřímu Schimmelovi, Ph.D. za věnovaný čas, odbornou pomoc a cenné rady nejen při zpracování mé diplomové práce. Dále děkuji celé své rodině, přítelkyni a přátelům za všestrannou podporu při studiu.

Brno ....

(podpis autora)

OBSAH

SEZNAM OBRÁZKŮ

SEZNAM TABULEK

ÚVOD

Úkolem diplomové práce je popsat metody ztrátového kódování pohyblivých obrazů a realizovat jednoduchý program v jazyce Java, který bude jednotlivé principy demonstrovat nad sekvencí obrázků podle standardu MPEG.

V textu jsou nejprve popsány důvody a způsoby komprese obrazu, jejich formáty, posuzování kvality obrazu a jiné základní poznatky. Dále je text zaměřen na jednotlivé standardy organizace ISO – standard pro zpracování statických obrazů JPEG a pohyblivých obrazů MPEG a standardy vytvořené skupinou ITU – H.26x a jejich použití.

Závěrem jsou shrnuty prostředky použité pro realizaci programu s popisem jeho struktury a ukázkou jeho výstupu.

1 ÚVOD DO KOMPRESE DAT

Dříve než budou popsány jednotlivé standardy, které se běžně používají v praxi, jsou nejprve uvedeny základní poznatky z komprese dat a lidského vnímání obrazu.

1.1 Důvod komprese dat

Důvod, proč se zaobírat kompresí digitalizovaných obrazů, a to jak ztrátovou, tak i bezeztrátovou, je jasně patrný z následujícího příkladu.

Uvažujme videoformát SDTV (Standard-definition television – televizní systém se standardním rozlišením), tzn. rozlišení 720×576 obrazových bodů. Dále uvažujme, že jeden obrazový bod je vyjádřen pomocí hodnot jednotlivých základních barev R, G a B. Při velikosti těchto základních prvků 1 B dostaneme velikost jednoho snímku $720 \times 576 \times 3\,\mathrm{B} \approx 1,2\,\mathrm{MB}$. Dále, aby bylo video plynulé, je potřeba, aby za jednu vteřinu bylo zobrazeno minimálně 25 snímků. Pokud tedy vyjádříme velikost nekomprimovaného videa o délce 1 s, získáme $1,2 \times 25 = 30\,\mathrm{MB}$. Hodina takto uloženého videa by pak představovala 108 GB dat.

Vezmeme-li v úvahu vysoké rozlišení HDTV, získali bychom za hodinu záznamu přibližně 540 GB dat.

Je zřejmé, že bez komprese by bylo takovéto video nepoužitelné pro jakýkoliv účel. Proto je nutné digitální signál před přenosem komprimovat, tzn. musí být několikanásobně snížena přenosová rychlost při udržení si dostatečné kvality.

1.2 Obrazové formáty

Během vývoje digitálního zpracování obrazu bylo zavedeno několik standardů rozlišení obrazu. Prvním typem rozlišení, které je definováno ve standardu H.324 pro telekonference a videotelefony je formát CIF (Common Intermediate Format). Základní rozlišení tohoto formátu je 352×288. Formáty z něj odvozené mají násobné rozlišení např. QCIF 176×144 a 4CIF 704×576 [4]. Tyto typy rozlišení najdeme např. i u standardů H.261, H.263, ...

Podobně standard MPEG [14] definuje rozlišení SIF (Source Input Format) 360×240 a odvozené rozlišení QSIF 180×120 [4].

Poslední z nejpoužívanějších rozlišení digitalizovaných obrazů je SDTV 720×576 a HDTV 1920×1080 používané pro digitální televizní vysílání DVB (Digital Video Broadcasting).

Přehled nejčastěji používaných standardů je uveden v tabulce 1.1.

Tab. 1.1: Přehled jednotlivých standardů rozlišení obrazu.

Formát	**Rozlišení**	**Vzorkování**	**Snímků za vteřinu**
sub-QCIF	128 × 96	4:2:0	30
QCIF	176 × 144	4:2:0	30
CIF	352 × 288	4:2:0	30
4CIF	704 × 576	4:2:0	30
QSIF	180 × 120	4:2:0	30p/25p
SIF	360 × 240	4:2:0	30p/25p
SDTV	720 × 576	4:2:0/4:2:2	24p/25p/30p/50i/60i
HDTV	1920 × 1080	4:2:0/4:2:2	24p/25p/30p/50i/60i

p=progressive – neprokládané řádkování, i=interlaced – prokládané řádkování

1.3 Barevné modely

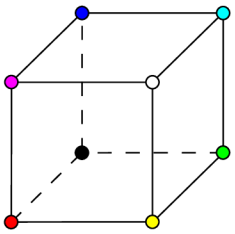
Barevný model je matematický aparát popisující jakým způsobem vzniká výsledná barva obrazového prvku [2]. Nejčastěji se v modelu vyskytují tři, někdy i čtyři, základní barevné složky. Každá barva v modelu je pak vyjádřena pomocí lineární kombinace těchto základních složek. Každému systému podle jeho potřeb lépe vyhovuje jiný barevný model. Např. zobrazovací jednotky nejčastěji používají model RGB, tiskárny model CMYK, v digitálním zpracování se často pracuje s modelem YC_BC_R, člověk při popisu barev nejčastěji hovoří v modelu HLS či HSB apod.

Barevný model může být *aditivní* nebo *subtraktivní* [2]. Aditivní model založený na vyzařování světla sčítá jednotlivé příspěvky světelných zdrojů. Světlost výsledné barvy záleží na množství jednotlivých příspěvků. Příkladem je model RGB. Oproti tomu model subtraktivní pracuje s odrazem bílého světla. Jednotlivé složky modelu tedy pohlcují světlo. Typickým zástupcem tohoto modelu je model CMYK.

1.3.1 Barevný model RGB

Barevný model RGB [5] používá k popisu obrazového bodu vždy kombinaci třech základních barev. Červenou – R, zelenou – G a modrou – B. Tento model je aditivní, tzn., že součtem těchto tří složek dostaneme barvu bílou. Jednotlivé složky bývají zpravidla vyjádřeny se stejnou přesností. Nejčastěji 8 bitů, tj. 24 bitů na pixel.

Model RGB je možné popsat pomocí jednotkové krychle a to i s jeho doplňkovým modelem CMY na obrázku 1.1. Výslednou barvu lze určit bodem v této krychli se souřadnicemi (r,g,b). Jelikož tento model obsahuje zdroje světla, je tento model vhodný pro zobrazovací jednotky.



Obr. 1.1: Barevný model RGB.

1.3.2 Barevný model YC_BC_R

Model RGB nemusí být nutně nejefektivnějším nástrojem, jak reprezentovat barvy. U digitálního zpracování obrazu se často používá model YC_BC_R [17]. Primárními složkami tohoto modelu jsou luminance–Y, a složky chrominanční–C_B a C_R. Toho se dá velice dobře využít pro kompresi obrazu, protože lidské oko je daleko více citlivé na jas obrazu než na jeho barvu. Vyplývá to z HVS, jenž je popsaný v kapitole 1.4. Luminanční neboli jasová složka je získána jako váhovaný součet primárních složek barevného modelu RGB. Podle ITU-R je

$$Y = 0,299r + 0,587g + 0,114b, \tag{1.1}$$

$$C_B = 0,564(b - Y) \tag{1.2}$$

a

$$C_R = 0,713(r - Y). \tag{1.3}$$

Váhovací koeficienty u rovnice 1.1 jsou získány díky znalostem HVS. Na první pohled je patrné, že lidské oko je nejvíce citlivé na zelenou barvu [5]. Proto je koeficient u příspěvku zelené barvy největší. Chrominanční složky jsou získány jako rozdíl modré, resp. červené barvy a celkového jasu. Rovnice lze uvést i v maticovém počtu, kde je zápis přehlednější, viz rovnice 1.4. Obor hodnot při osmibitovém kvantování je pro jasovou složku 0 až 255 a pro jasové složky -127,5 až 127,5.

$$\begin{pmatrix} Y \\ C_B \\ C_R \end{pmatrix} = \begin{pmatrix} 0,299 & 0,587 & 0,114 \\ -0,169 & -0,331 & 0,500 \\ 0,500 & -0,419 & -0,081 \end{pmatrix} \cdot \begin{pmatrix} r \\ g \\ b \end{pmatrix} \tag{1.4}$$

Převody mezi modely RGB a YC_BC_R jsou samozřejmě *bezeztrátové*. Příklad obrázku v modelu RGB a jednotlivé složky modelu YC_BC_R ilustruje obr. 1.2 [1].

[1] Složky C_B a C_R jsou transponovány do střední hodnoty jasu a to z důvodu, aby byly patrné i záporné hodnoty složky, které by bez transpozice nebyly viditelné.

Obr. 1.2: Obrázek v RGB a jednotlivé složky modelu YC_BC_R.

1.4 Vnímání obrazu člověkem

O to, jak obraz vnímá člověk, se zajímá tzv. HVS (Human Visual System – systém lidského vizuálního vnímání). Základní poznatky plynou z anatomie lidského oka. Světlo, které dopadá na lidské oko, projde nejprve rohovkou a čočkou. Tyto části oka mají za úkol paprsek světla zaostřit tak, aby dopadal přesně na sítnici, která je na druhé stěně oka [5].

Sítnice je pokryta světlocitlivými receptory, které vysílají do mozku nervové impulsy. Receptory jsou dvojího typu – *tyčinky* a *čípky*. Tyčinky reagují pouze na intenzitu světla, a proto dokáží rozlišovat pouze jas. Barevný vjem zprostředkovávají čípky. Čípků je přibližně dvacetkrát méně a jsou soustředěny hlavně kolem žluté skvrny, což je místo na sítnici s nejostřejším viděním. Při dostatečné hladině osvětlení převládá vnímání čípky, tudíž vidíme barevně. Při slabém osvětlení vnímá oko spíše tyčinkami, které nejsou citlivé na barvu, a proto vidíme předměty pouze v odstínech šedi.

Z poměru tyčinek a čípků plyne velice důležitý poznatek, a to, že lidské oko je daleko více citlivé na jas než na barvu. Změnu jasu člověk zaregistruje daleko více než změnu odstínu barvy [5]. Navíc jednotlivé čípky nejsou citlivé na všechny barvy stejně. Poměr citlivosti na základní barvy RGB lze vypozorovat na rovnici 1.1 pro výpočet celkového jasu obrazu.

Další psychovizuální vlastnosti lidského oka:

- Člověk má velice omezené vnímání rychle se měnících částí v obrazu [23]. Oko je více citlivé na statické prvky v obraze. Navíc při skokové změně obrazu lidskému oku trvá určitý čas, po který nevnímá detaily, než se obrazu přizpůsobí. Tyto skutečnosti jsou často označovány jako *časové maskování*.
- Lidské oko je více citlivé na souvislé plochy než na časté změny v obraze. Tomuto říkáme *prostorové maskování*.
- HVS má omezenou citlivost pro detaily (vysoké frekvence) v obraze. Označu-

jeme též jako *frekvenční maskování*.

- Pokud budeme jednotlivé snímky měnit s dostatečnou frekvencí, bude lidské oko vnímat obraz jako plynule se měnící. Tato frekvence musí být nejméně 24 Hz.

Toto byly irelevantní prvky v obrazu, které je zbytečné přenášet nebo zobrazovat, protože pro lidské oko jsou nepodstatné. V obrazu však lze vypozorovat i redundantní prvky. *Časová redundance* nám říká, že po sobě jdoucí snímky mají vysoký koeficient korelace. *Redundance prostorová* popisuje fakt, že sousední obrazové body mívají podobný jas.

1.5 Formáty vzorkování obrazu

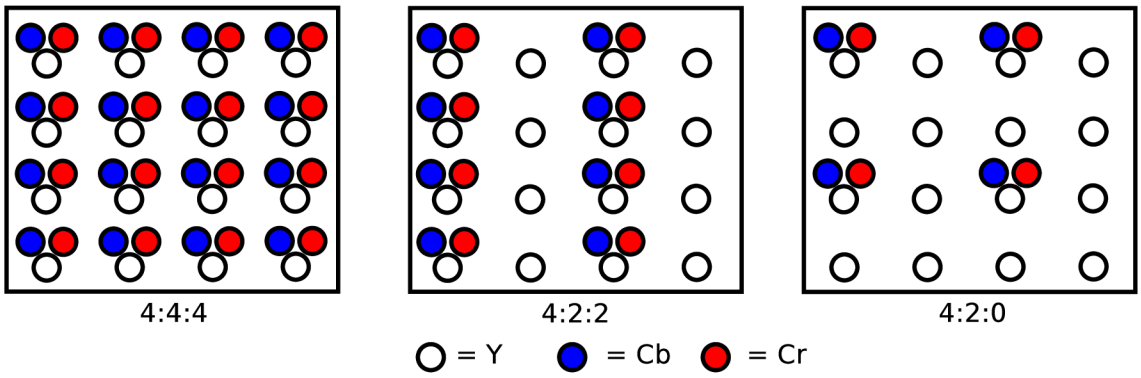
Vyjdeme-li z modelu YC_BC_R, kde je oddělená jasová a chrominanční složka obrazového signálu, můžeme si dovolit odstranit v obraze irelevantní prvky podvzorkováním chrominančních složek. Jelikož je lidské oko na změnu barvy méně citlivé, ve výsledném obrazu nezpozoruje téměř žádnou změnu [5]. Podvzorkování můžeme provést v horizontálním i vertikálním směru. Formát vzorkování se značí třemi číslicemi oddělené dvojtečkami [21].

Formát vzorkování neupraveného obrázku je označován jako 4:4:4. Matice $\mathbf{Y}$, $\mathbf{C_B}$ i $\mathbf{C_R}$ mají stejnou velikost a nedochází zde ke ztrátě informací. Jeden prvek v jakékoli matici odpovídá právě jednomu obrazovému bodu.

U vzorkování 4:2:2 dochází k horizontálnímu podvzorkování chrominančních složek. Jasová složka zůstává neměnná a matice $\mathbf{C_B}$ a $\mathbf{C_R}$ mají poloviční velikost. Jedna hodnota v matici $\mathbf{C_B}$ a $\mathbf{C_R}$ je společná pro dva v horizontálním směru po sobě následující prvky jasové složky. U tohoto i všech ostatních typů vzorkování dochází nenávratně ke ztrátě informací. Jedná se tedy o *ztrátovou* kompresi.

Posledním z nejčastěji používaných formátů vzorkování je formát 4:2:0. Zde je jasová složka opět neměnná a chrominanční složky jsou podvzorkovány v obou směrech. To má za následek zmenšení velikosti matic $\mathbf{C_B}$ a $\mathbf{C_R}$ na čtvrtinovou velikost oproti matici s jasovou informací. Jeden prvek z matice $\mathbf{C_B}$, resp. $\mathbf{C_R}$ je pak při rekonstrukci obrazu rozkopírován ke čtyřem jasovým prvkům v jeho okolí. Tímto uspoříme 50 % z počáteční velikosti dat, aniž by lidské oko zaznamenalo změnu kvality.

Přehled formátů vzorkování je vyobrazen na obrázku 1.3.



Obr. 1.3: Formáty vzorkování obrazu.

1.6 Určování kvality obrazu

Při vyhodnocování kvality obrazu se vychází z porovnání originálního snímku, který vstupuje do určité transformace, např. do kodéru a snímku rekonstruovaného. Existují dva základní principy hodnocení kvality – *subjektivní* a *objektivní* [18].

Při subjektivním hodnocení provádíme pozorování zkoumané množiny snímků reprezentativní skupinou osob, která subjektivně posuzuje kvalitu obrazu podle předem určené stupnice. Často používanou stupnicí je např. pětiškálová stupnice MOS (Mean Opinion Score) definovaná v doporučení ITU-R BT.500. Tato metoda má značné nevýhody, a proto se pro rychlé určení kvality obrazu používá objektivní hodnocení.

Objektivní hodnocení klasifikuje rozdíly mezi originálním a rekonstruovaným snímkem pomocí matematicky definovaných postupů. Tyto metody nazýváme též jako *pixel-based metrics* [6]. Nejpoužívanějšími metrikami jsou:

- MAE – Mean Absolute Error – střední absolutní chyba

$$MAE = \frac{1}{MN} \cdot \sum_{i=1}^{M} \sum_{j=1}^{N} \left| x\left(i,j\right) - \widetilde{x}\left(i,j\right) \right|, \tag{1.5}$$

- SAE – Sum of Absolute Errors – součet absolutních chyb

$$SAE = \sum_{i=1}^{M} \sum_{j=1}^{N} \left| x\left(i,j\right) - \widetilde{x}\left(i,j\right) \right|, \tag{1.6}$$

- MSE – Mean Square Error – střední kvadratická chyba

$$MSE = \frac{1}{MN} \cdot \sum_{i=1}^{M} \sum_{j=1}^{N} \left[x\left(i,j\right) - \widetilde{x}\left(i,j\right) \right]^2, \tag{1.7}$$

- NMSE – Normalized Mean Square Error – normalizovaná střední kvadratická chyba

$$NMSE = \frac{1}{L^2} \cdot MSE, \tag{1.8}$$

- PSNR – Peak Signal to Noise Ratio – špičkový poměr signálu k šumu

$$PSNR_{\mathrm{dB}} = 10 \cdot \log_{10}\left(\frac{1}{NMSE}\right). \tag{1.9}$$

kde vždy: M, N jsou rozměry snímku, $x, \widetilde{x}$ jsou originální a rekonstruovaný snímek, L je dymanický rozsah hodnoty jasu obrazového bodu a i, j jsou souřadnice obrazového bodu ve snímku.

Tyto veličiny, vycházející pouze z jasové informace, lze sice jednoduše vypočítat, ovšem nezačleňují nějak psychovizuální vlastnosti lidského oka. Ne vždy se tak můžeme spolehnout na vyhodnocení kvality podle takto získaných ohodnocení. Referenční hodnotou pro přijatelnou kvalitu bývá PSNR $\approx 30\,\mathrm{dB}$, viz obr. 1.4

a)MSE = 6,8; MAE = 1,7; PSNR = 40 dB

b)MSE = 65; MAE = 5,8; PSNR = 30 dB

c)MSE = 187; MAE = 10; PSNR = 25 dB

d)MSE = 409; MAE = 14,5; PSNR = 22 dB

Obr. 1.4: Příklady objektivních metod vyhodnocení kvality obrazu.

2 STANDARD JPEG

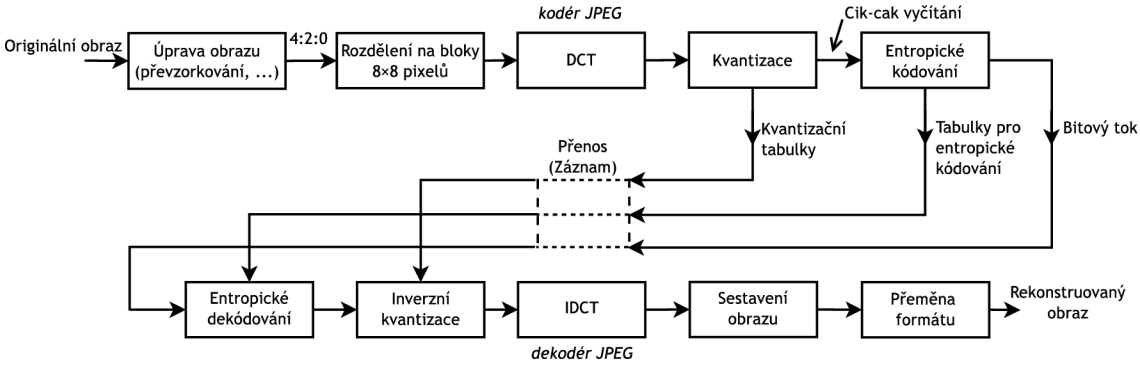
Mezinárodní standard JPEG (Joint Photographic Experts Group) definovaný skupinou ISO v doporučení ISO/IEC IS 10918-1 je nejpoužívanějším digitálním formátem pro barevné statické obrazy. Je navrhnut tak, aby dosahoval dobrého kompresního poměru při zachování vysoké kvality obrazu, a byl přitom výpočetně co možná nejméně náročný.

2.1 Princip kodéru a dekodéru JPEG

Základní struktura kodéru a dekodéru JPEG je uvedena na obr. 2.1. Vstupní obraz je nejprve převeden do barevného modelu YC_BC_R a poté podvzorkován v poměru 4:2:0 [13].

Následuje rozdělení do bloků o velikosti 8×8 obrazových bodů neboli pixelů. Rozdělení probíhá směrem zleva doprava a shora dolů. Tato velikost bloku byla vybrána jako poměr mezi dostatečnou kompresí a přijatelnými náklady [21]. Nad blokem 8×8 pixelů je provedena dopředná dvourozměrná diskrétní kosinova transformace, označovaná zkratkou DCT (Discrete Cosine Transform – diskrétní kosinova transformace). Princip zpracování obrazu po blocích vnáší do výsledného obrazu blokový artefakt – nespojitost v návaznosti jednotlivých bloků.

DCT provádí transformaci z tzv. *prostorové oblasti* do *oblasti kmitočtové*. Transformace má přínos v tom, že převede vstupní vzorky, které jsou mezi sebou silně korelovány na jiné vzorky, které na sobě závislé nejsou. Další výhodou transformace je, že počet nenulových výsledných vzorků, tzv. *kmitočtových koeficientů*, je výrazně méně než počet vstupních vzorků. DCT lze odvodit z Fourierovy transformace vhod-



Obr. 2.1: Kodér a dekodér standardu JPEG.

nou substitucí [21]. Pro dopřednou a zpětnou DCT platí:

$$G_{[i,j]} = \frac{2}{\sqrt{mn}} C_i C_j \sum_{x=0}^{n-1} \sum_{y=0}^{m-1} p_{[x,y]} \cos\left[\frac{(2y+1)\, j\pi}{2m}\right] \cos\left[\frac{(2x+1)\, i\pi}{2n}\right], \tag{2.1}$$

$$p_{[x,y]} = \frac{2}{\sqrt{mn}} \sum_{i=0}^{n-1} \sum_{j=0}^{m-1} C_i C_j G_{[i,j]} \cos\left[\frac{(2x+1)\, i\pi}{2n}\right] \cos\left[\frac{(2y+1)\, j\pi}{2m}\right], \tag{2.2}$$

kde:

- i, j, resp. x, y jsou souřadnice bodu v kmitočtové, resp. prostorové oblasti,
- G, resp. p je blok vzorků v kmitočtové, resp. prostorové oblasti,
- m, n jsou rozměry bloku, v našem případě 8,
- $C_f = \frac{1}{\sqrt{2}}$ pro $f = 0$ a $C_f = 1$ pro $f > 0$.

Stejnosměrný koeficient $G_{[0,0]}$ (DC koeficient) reprezentuje stejnosměrnou složku, tedy střední hodnotu jasu transformovaného bloku. Ostatní koeficienty, označované jako střídavé (AC koeficienty), představují střídavé složky a mohou dosahovat i záporných hodnot. Příklad, jak vypadá vstupní blok před a po transformaci, ilustruje obr. 2.2.

127	129	122	130	121	134	133	142
118	127	129	137	126	125	111	109
130	131	132	131	130	131	134	136
131	125	125	114	122	122	141	146
129	125	139	122	137	117	126	126
124	121	144	124	148	121	133	121
119	122	129	102	136	126	138	131
139	127	148	113	146	126	140	129

a)

1029	-8	2	-4	-4	-5	3	30
-6	2	0	-5	6	6	-2	-39
2	0	-4	6	-5	-1	3	-4
-7	-1	-10	-2	2	-1	-1	1
10	-4	20	-2	3	2	2	2
2	-26	27	3	-4	-2	-3	-3
21	-1	-1	-1	3	4	1	4
3	-1	3	0	-1	-4	0	-2

b)

4	3	2	4	6	10	15	16
3	3	4	5	6	14	17	18
4	3	5	6	10	13	17	18
4	4	5	6	12	21	20	17
5	5	9	13	19	26	24	18
6	8	13	17	19	25	29	23
12	18	19	22	25	30	30	25
18	22	22	22	27	25	25	25

c)

257	-3	1	-1	-1	-1	0	2
-2	1	0	-1	1	0	0	-2
1	0	-1	1	-1	0	0	0
-1	0	-2	0	0	0	0	0
2	-1	2	0	0	0	0	0
0	-3	2	0	0	0	0	0
2	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0

d)

Obr. 2.2: a)Blok vstupních vzorků obrazu, b)blok koeficientů po transformaci, c)kvantizační matice, d)blok koeficientů po kvantování.

Po provedení dopředné DCT je blok kmitočtových koeficientů kvantován pomocí kvantizačních tabulek. Jedná se o skalární kvantizaci, tedy obyčejné dělení se

zaokrouhlením na nejbližší celé číslo. Kvantizační tabulky jsou standardem pouze doporučeny a velice často si jednotlivé subjekty vytvářejí kvantizační tabulky své [13]. Zpravidla bývá kvantizační tabulka pro jasovou složku jiná než pro chrominanční složky, protože lidské oko je více citlivé na jas než na barvu. To nám dovolí chrominanční složky kvantovat hruběji než jas. Podobou kvantizačních tabulek tak můžeme upravovat kvalitu výsledného obrazu, resp. kompresní poměr [21]. Příklad kvantizační tabulky je uveden v obr. 2.2 c).

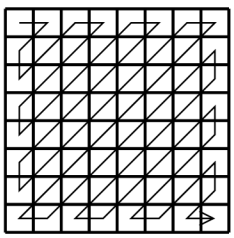
Součástí kvantizačního procesu je i *prahování*, které vynuluje takové koeficienty, jejichž absolutní hodnota po kvantizaci je menší než zvolený práh. Tento proces je jednosměrný, tedy nevratný a zavádí ztrátovost. Pokud by bylo v příkladu na obr. 2.2 nastaveno prahování na 1, byly by modře podbarvené prvky vynulovány.

Dalším krokem je vyčítání koeficientů z bloku do sériového toku. Toto je realizováno tzv. cik-cak čtením podle úhlopříčky bloku jak ilustruje obr. 2.3. Vyčítání odpovídá významově zvyšujícímu se kmitočtu jednotlivých koeficientů [21]. Důležité je, že od určitého koeficientu jsou všechny následující (významem se stále vyšším kmitočtem) koeficienty nulové.

DC koeficienty jsou přenášeny odděleně jako diference od předešlého kódovaného bloku [13]. Ostatní koeficienty postupně vstupují do procesu entropického kódování s proměnnou délkou slova.

Dekodér postupuje obráceně, tedy nejprve provede zpětné entropické kódování, poté provede inverzní kvantování (místo nulových kmitočtových koeficientů dosadí odpovídající hodnoty z kvantizační tabulky). Pokračuje zpětnou dvourozměrnou DCT a nakonec sestaví obraz z jednotlivých bloků a obraz převede do modelu RGB.

JPEG definuje také i bezeztrátový typ komprese [13], který je založený na DPCM (Differential Pulse Code Modulation - rozdílová pulsní kódová modulace). Každý pixel je predikován ze tří nejbližších pixelů a tato predikovaná hodnota je entropicky kódována a přenášena. Nedochází tak k žádnému zkreslení vlivem ztrátového procesu, ovšem komprese nedosahuje takových hodnot jako u ztrátového typu, a proto je téměř nepoužívaná.



Obr. 2.3: Tzv. cik-cak vyčítání kmitočtových koeficientů v kodéru JPEG.

2.2 MJPEG

MJPEG (Motion JPEG) je kodek pro pohyblivý obraz. Jednotlivé snímky videosekvence kóduje jako samostatný snímek JPEG. I přesto, že komprese oproti standardům používaných pro kompresi videosekvencí je několikanásobně menší, je používán v několika obrazových komunikačních systémech a aplikacích, kde je k dispozici velká šířka pásma přenosového kanálu. Jeho výhody jsou v malých požadavcích na výpočetní nároky a nenáročné implementaci algoritmu [18].

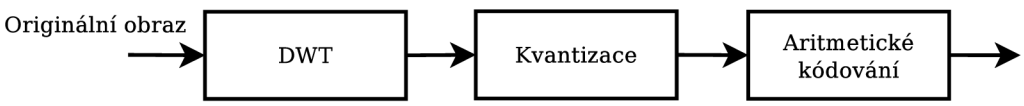
2.3 JPEG 2000

Novější standard JPEG 2000 vychází na rozdíl od svého předchůdce z diskrétní waveletovy transformace DWT (Discrete Wavelet Transform – diskrétní waveletova transformace) [13]. To přináší zvýšení kompresního poměru při zachování stejné kvality, ovšem za cenu vyšší výpočetní náročnosti. Pro stejnou kvalitu obrazu dosahuje JPEG 2000 nejméně dvakrát vyšší komprese než JPEG.

JPEG 2000 obsahuje ztrátový i bezeztrátový způsob komprese, podporuje přenos víceúrovňových a sloučených obrazů a má otevřenou architekturu. Novinkou je také kódování ROI (Region of Interest – oblast zájmu). Tato vlastnost dovoluje během procesu kódování zacházet s určitou oblastí uvnitř snímku odlišně, např. zvýšit kvalitu [18].

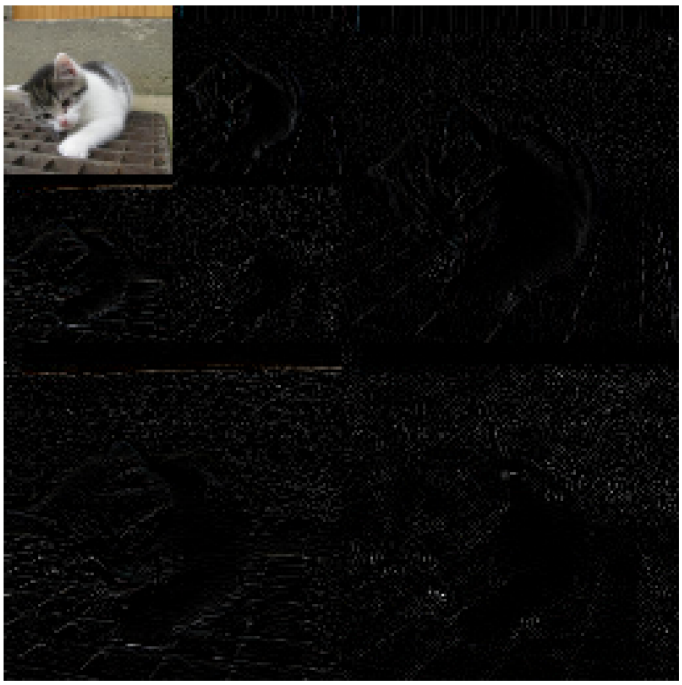
Blokové schéma kodéru je na obr. 2.4. Největším rozdílem oproti kodéru JPEG je použití vlnkové transformace místo DCT a dále využití stejné architektury pro ztrátový i bezeztrátový typ komprese. Základním kódovacím prvkem je zde část snímku o rozměrech $2^n \times 2^n$. S každou částí je provedena vlnková transformace, která každou část rozloží do subpásem. Koeficienty vlnkové transformace jsou kvantovány a následně aritmeticky kódovány. Jak vypadá obraz po provedení vlnkové transformace se stupněm 2, ilustruje obr. 2.5 a 2.6.

Z principu komprese nevznikají u tohoto standardu blokové artefakty a nativně podporuje postupný přenos obrazu, tzn., že obraz je zobrazován se vzrůstající pixelovou přesností [13]. Nevýhodou oproti JPEG je vyšší výpočetní náročnost, což se může odrazit např. v delší odezvě při ukládání nebo zobrazení snímků na digitálním fotoaparátu.



Obr. 2.4: Blokové schéma kodéru JPEG 2000.

Obr. 2.5: Originální obrázek.



Obr. 2.6: Obrázek po rozkladu pomocí DWT se stupněm 2.

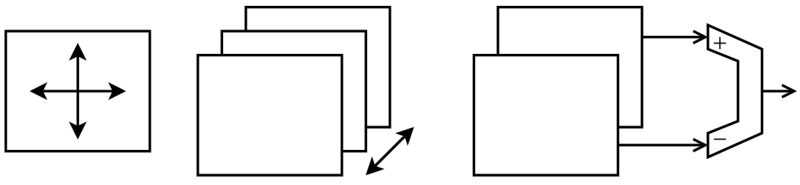
3 OBECNÉ METODY PRO KOMPRESI VIDEOSEKVENCÍ

Během vývoje standardů pro kompresi videosekvencí bylo vyvinuto několik metod, které lze nalézt v mnoha typech standardů a doporučení. Základní myšlenkou pro dosažení vysoké komprese při zachování dostatečné kvality je maximální využití nedokonalostí lidského oka v kombinaci s obecnými kompresními metodami kódování jako je např. kódování s proměnnou délkou slova apod.

3.1 Způsoby odstranění redundance

Metody MPEG se snaží odstraňovat nadbytečné informace z videosekvence pomocí *inter-* a *intra-kódování* [23]. Intra-kódování odstraňuje nadbytečné informace uvnitř snímku. Zabývá se tedy pouze *prostorovou redundancí*, tzn., že využívá vysoké korelace mezi sousedními pixely ve snímku. Z toho plyne, že lze ušetřit bitový tok pomocí kódování pouze rozdílů mezi těmito pixely. U inter-kódování dochází k odstraňování *časové redundance*. Zde se vychází z velké shody právě kódovaného snímku a snímku předcházejícího - referenčního. Opět není nutné kódovat kompletní informace o daném snímku, ale kódovat pouze snímek rozdílový. Ten je získán odečtením navzájem si odpovídajících obrazových bodů ve snímcích. Dekodér si může ukládat referenční snímky v mezipaměti, tudíž je velice jednoduché rekonstruovat originální snímek. Tyto metody ilustruje obr. 3.1.

Pokud je ovšem obraz v po sobě následujících snímcích posunut, např. při videosekvenci projíždějícího auta po náměstí, bude rozdílový snímek získaný diferencí odpovídajících si pixelů velmi neefektivní. Navzájem odpovídající si pixely budou pravděpodobně velice odlišné, přestože jednotlivé snímky obsahují velice podobné informace. Rozdílový snímek pak bude obsahovat velmi mnoho informací a efektivita kódování poklesne. Z tohoto důvodu zavádí MPEG *kompenzaci pohybu* [17]. Ta má za úkol v referenčním snímku vyhledat blok pixelů s největší shodou s blokem



Obr. 3.1: Grafické znázornění inter- a intra-predikce a způsob získání rozdílového snímku.

pixelů z právě kódovaného snímku. Kompenzace pohybu tak neporovnává pouze navzájem odpovídající si pixely, ale hledá největší shodu v určité *vyhledávací oblasti*. V tomto případě se kódují, tzv. *vektory pohybu*, které při sestavování snímku říkají, na kterém místě a jak vzdálený v referenčním snímku je nejpodobnější blok pixelů. Protože si bloky nemusí být naprosto shodné, je nutné kódovat navíc i jejich rozdíl. Kompenzace pohybu, kdy se tedy dekodéru posílají pouze vektory pohybu s rozdílem mezi danými bloky, je velice dobrý nástroj k dosažení vysokého komprimačního poměru.

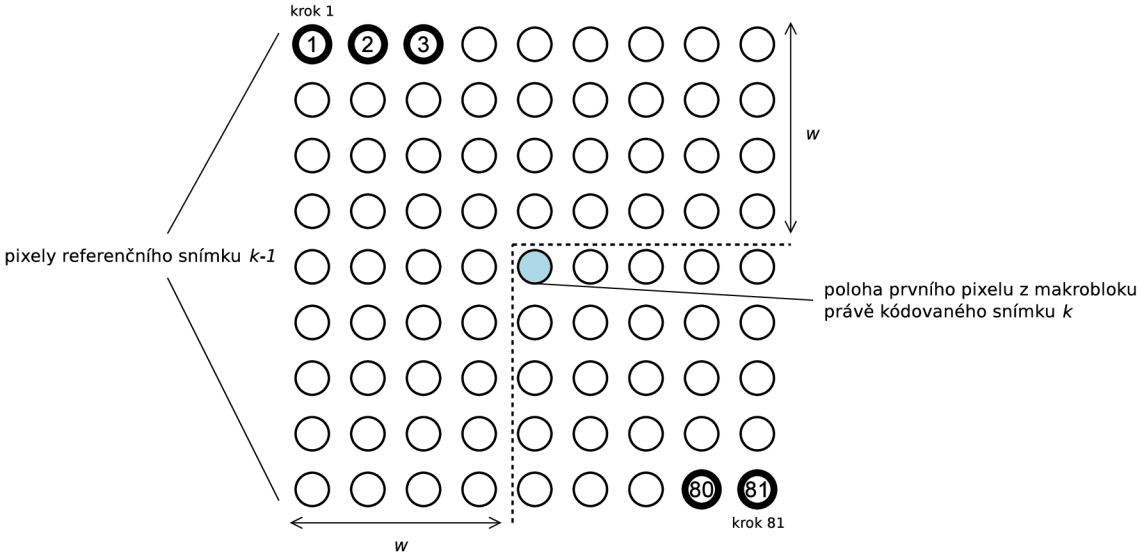
3.2 Metody kompenzace pohybu

K nalezení vektorů pohybu existuje mnoho metod, některé z nich jsou popsány v této kapitole, více v [18]. Řeší se kompromis mezi dobou hledání, potažmo výpočetní náročností, a nalezení nejlepší možné shody, což má za následek minimalizaci diferencí mezi nalezeným blokem a blokem referenčním. Vždy je hledána nejlepší shoda k bloku pixelů o rozměrech $M \times N$ v prohledávací oblasti o $U \times V$ pixelů. Velikost bloku i prohledávací oblasti jsou typicky čtvercové, tedy $M = N$ a $U = V$. Pokud není určeno jinak, vyhledávání probíhá pouze v jasové složce obrazu. Vektory pohybu u chrominačních složek se pouze přidruží. K vyhodnocení shody se používá některá z metod objektivního posuzování kvality, viz kapitola 1.6.

3.2.1 Metoda Full-Search

Tento velice jednoduchý algoritmus, který jako jeden z mála nepatří mezi techniky rychlé predikce, porovnává každý možný blok v prohledávací oblasti. Pořadí, v jakém bude porovnávání prováděno, se může lišit. Pokud platí $M = N$ a $U = V$, je možné zavést veličinu prohledávací vzdálenosti $w = (U - M)/2$. Počet kombinací k porovnání odpovídá $(2w + 1)^2$. Pokud např. hledáme blok o velikosti 8×8 pixelů v oblasti 16×16 pixelů, dostáváme $w = 4$, tj. 81 pozic pro porovnání shody. Toto je nejčastěji používaná varianta [6]. Tento příklad je znázorněn na obr. 3.2. Prohledávací oblast lze samozřejmě zvětšovat, ovšem výpočetní náročnost se bude razantně zvyšovat. Ideálně by měla prohledávací oblast být stejná jako je velikost snímku, což je vzhledem k výpočetní náročnosti nereálné. V praxi bylo zjištěno, že výrazné zvětšování prohledávací oblasti nad 16×16 pixelů nepřináší podstatné zlepšení kompenzace pohybu [6].

Nevýhodou této metody je, že z důvodu porovnávání každé možné prohledávací pozice je výpočetní náročnost velmi vysoká. Naopak výhodou je, že je nalezen skutečně blok s největší možnou shodou = nejmenším rozdílem.



Obr. 3.2: Metoda Full-Search pro $w = 4$.

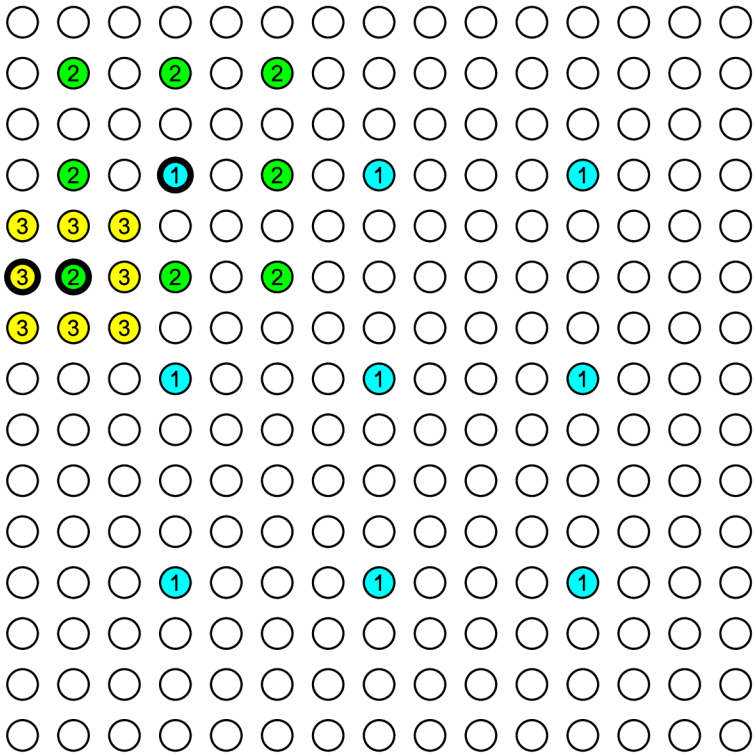
3.2.2 Metoda *N*-Step-Search

Jak název napovídá, prohledávání touto metodou probíhá v N krocích. Nejčastěji se používá tříkroková varianta. Prohledávací oblast je opět čtvercová o velikosti strany $2 \times (2^N - 1)$ pixelů. Postup je následující [6]:

1. Výchozím bodem je pozice (0,0) a velikost kroku $k = 2^{N-1}$.
2. Nalezení osmi pozic v bodech $\pm k$ od výchozí pozice.
3. Z těchto osmi pozic a pozice výchozí je vybrán blok s největší shodou na základě některé z metod objektivního posuzování kvality.
4. Pozice bloku s největší shodou je vybrána jako nová výchozí pozice.
5. Velikost kroku k je snížena na polovinu.
6. Jsou opakovány kroky 3 až 5 dokud není $k = 1$.

Obr. 3.3 ilustruje průběh algoritmu pro $N = 3$.

Pro získání vektorů pohybu je potřeba provést pouze $8N + 1$ porovnání, což výrazně snižuje výpočetní náročnost. Metoda ovšem nemusí nutně najít blok s největší globální shodou. Princip metody totiž nevylučuje, že metoda nalezne pouze blok s největší lokální shodou.

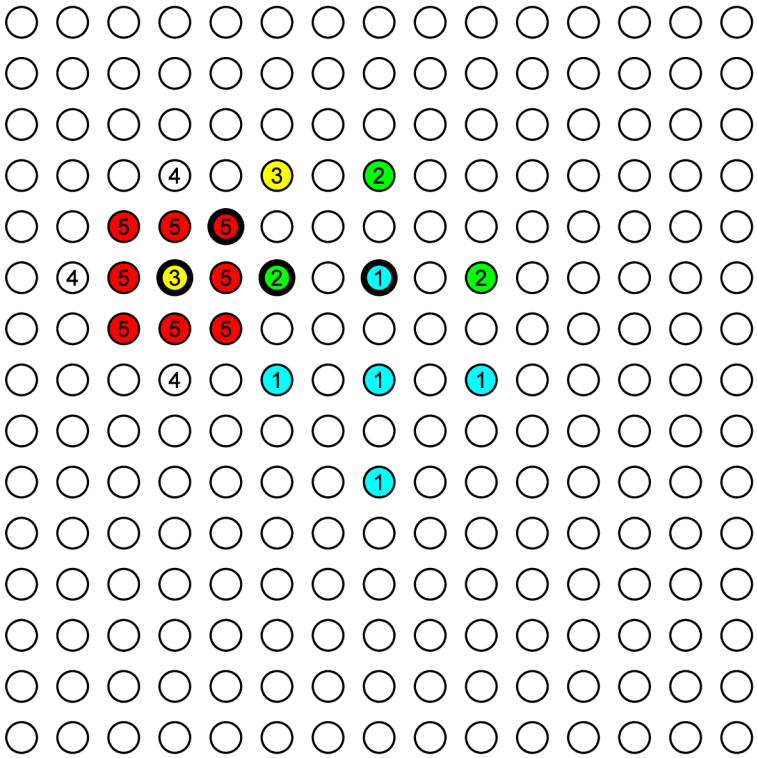


Obr. 3.3: Metoda N-Step-Search pro $N = 3$.

3.2.3 Metoda Logarithmic-Search

Metoda s logaritmickým výběrem probíhá podobně jako metoda N-Step-Search. Postup znázorněný na obr. 3.4 lze popsat následujícími kroky [18]:

1. Výchozím bodem je pozice (0,0) a velikost kroku k.
2. Nalezení čtyř pozic v bodech $\pm k$ od výchozí pozice (pouze ve vertikálním a horizontálním směru).
3. Z těchto čtyř pozic ve tvaru „+“ a pozice výchozí je vybrán blok s největší shodou na základě některé z metod objektivního posuzování kvality.
4. Pozice bloku s největší shodou je vybrána jako nová výchozí pozice. Pokud je nová výchozí pozice stejná jako předcházejí, je krok zmenšen na polovinu. V jiném případě je krok nezměněn.
5. Opakují se kroky 2 až 5, dokud $k \neq 1$.
6. Při $k = 1$ je nalezeno osm pozic přímo sousedících s výchozí pozicí.
7. Z těchto osmi pozic a pozice výchozí je vybrán blok s největší shodou.



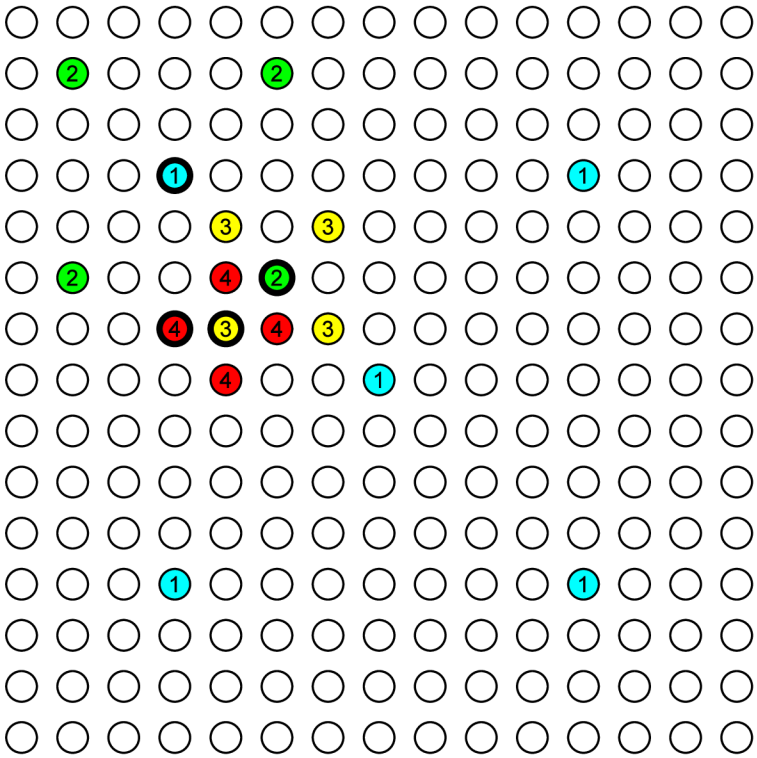
Obr. 3.4: Metoda Logarithmic-Search s krokem $k = 2$.

3.2.4 Metoda Cross-Search

Tato metoda opět vychází z metody N-Step-Search. Počet porovnání v každém kroku je pouze pět jako u metody Logarithmic-Search, ovšem pozice pro porovnání tentokrát tvoří tvar „X“. Algoritmus probíhá podle těchto bodů [3]:

1. Výchozím bodem je pozice (0,0) a velikost kroku $k = 2^{N-1}$.
2. Nalezení čtyř pozic v bodech $\pm k$ od výchozí pozice (pouze v krajních bodech prohledávací oblasti).
3. Z těchto čtyř pozic a pozice výchozí je vybrán blok s největší shodou na základě některé z metod objektivního posuzování kvality.
4. Pozice bloku s největší shodou je vybrána jako nová výchozí pozice.
5. Pokud je $k > 1$ je velikost kroku snížena na polovinu.
6. Opakují se kroky 2 až 6, dokud $k > 1$.
7. Pokud je nová výchozí pozice vlevo nahoře nebo vpravo dole tvaru „X“ probíhá nalezení nových pozic opět ve tvaru „X“ se vzdáleností ± 1 od výchozí pozice. V jiném případě probíhá nalezení nových pozic ve tvaru „+“ se vzdáleností ± 1 od výchozí pozice.
8. Z těchto čtyř pozic a pozice výchozí je vybrán blok s největší shodou.

Obr. 3.5 ilustruje příklad průběhu algoritmu pro $N = 3$.



Obr. 3.5: Metoda Cross-Search pro $N = 3$.

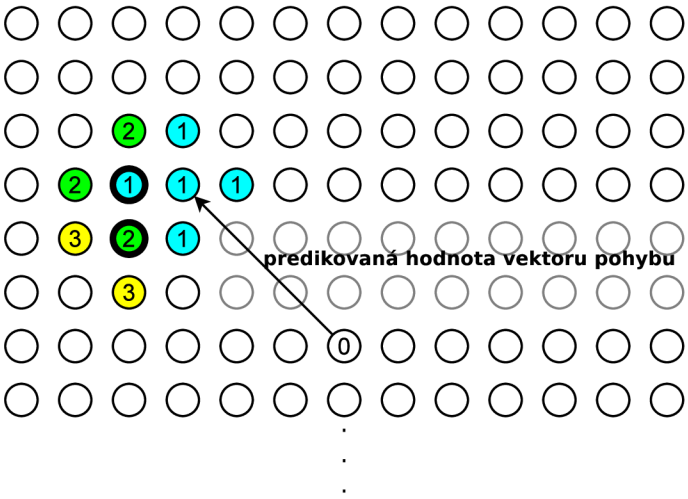
3.2.5 Metoda Nearest Neighbours-Search

Metoda nejbližšího souseda [18] navržená pro kodeky H.263 [11] a MPEG-4 [17] využívá skutečnosti, že vektory pohybu sousedních bloků jsou často velmi korelovány. Každý vektor pohybu předem očekává určitý posun od výchozí pozice nejčastěji stanovený pomocí mediánu předcházejících vektorů pohybu. Postup hledání vektorů pohybu je následující:

1. Výchozím bodem je pozice (0,0).
2. Je nalezena nová výchozí pozice pomocí předpovídaného vektoru pohybu.
3. Nalezení čtyř pozic ve vzdálenosti ± 1 od výchozí pozice ve tvaru „+“.
4. Z těchto čtyř pozic a pozice výchozí je vybrán blok s největší shodou na základě některé z metod objektivního posuzování kvality.
5. Pokud má nejlepší shodu výchozí pozice, jsou vektory pohybu stanoveny z ní. V jiném případě je pozice s největší shodou vybrána jako nová výchozí pozice a pokračuje se bodem 3.

Iterační proces lze také ukončit, pokud je dosaženo hrany prohledávací oblasti. Příklad algoritmu je na obr. 3.6.

Tato metoda dosahuje nejlepších výsledků, pokud v mapě vektorů pohybu nejsou obsaženy náhlé změny.

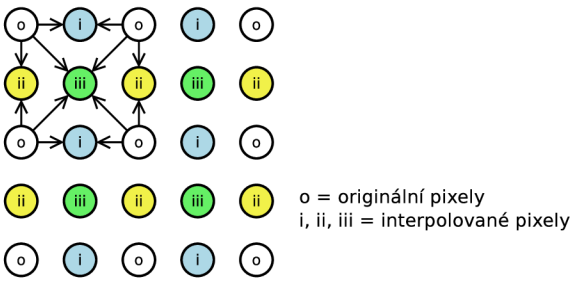


Obr. 3.6: Metoda Nearest Neighbours-Search.

3.3 Sub-pixelová kompenzace pohybu

Předchozí metody pro kompenzaci pohybu pracovaly s prohledávací oblastí vyjádřenou pomocí celých čísel – celých pixelů. Pro mnoho bloků obrazových bodů je ovšem možné nalézt nejlepší shodu pomocí interpolace v prohledávací oblasti. Toto vyhledávání nazýváme *se sub-pixelovou přesností*. Nejčastěji uvažujeme lineární interpolaci. Přínosem interpolace je zvýšení rozlišení prohledávací oblasti a získání pixelů, které nemusí referenční snímek vůbec obsahovat. Na takto rozšířenou prohledávací oblast je pak nasazena jedna z klasických metod pro získání vektorů pohybu. Nevýhodou je navýšení výpočetní náročnosti.

Nejčastěji používanou sub-pixelovou metodou je půl-pixelová kompenzace pohybu. Obsahuje jí např. standard H.263. Šipky v obr. 3.7 popisují, jakým způsobem jsou získávány interpolované hodnoty. Sub-pixelové metody s vyšším stupněm interpolace, např. $\frac{1}{4}$-pixelová kompenzace pohybu, se teprve prosazují, protože zvýšením stupně interpolace roste výpočetní náročnost a složitost implementace algoritmu.



Obr. 3.7: Půl-pixelová interpolace.

3.4 Rozšíření kompenzace pohybu

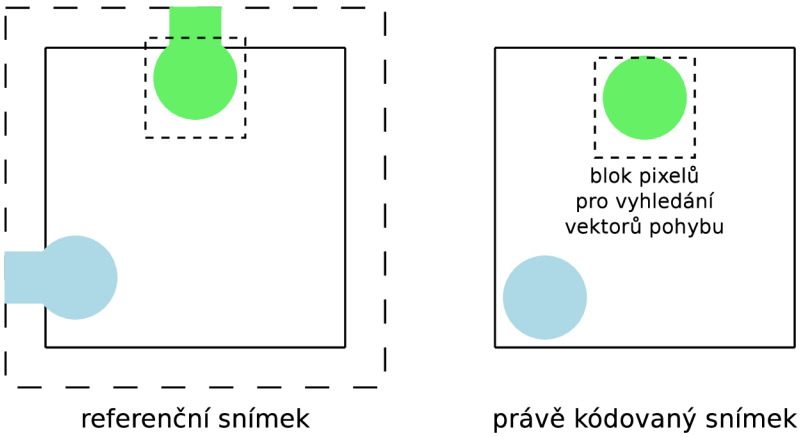
Kromě výše zmíněných metod existují další způsoby jak zvýšit efektivitu komprese. Některá řešení se nabízejí přímo ze základních metod.

3.4.1 Proměnná velikost bloku

Pokud budeme vyhledávat nejlepší shodu bloku o velikosti 16×16 pixelů v referenčním snímku, může se snadno stát, že v místech s hustými změnami pohybu bude toto kódování neefektivní. Naopak menší blok přináší navýšení výpočetní náročnost. V místech, kde videosekvence má spíše statický charakter, je použití menšího bloku neefektivní. Např. standard H.263 Annex F proto umožňuje během kódování jednoho snímku přepínat mezi dvěma velikostmi hledaného bloku (16×16 a 8×8 pixelů). Tento způsob optimalizuje vyhledávání vektorů pohybu. Nevýhodou je, že je nutno přidat informaci o jakou velikost bloku se jedná. Obecně může velikost bloků nabývat více hodnot.

3.4.2 Vektory pohybu bez hranic

Pokud se pohyb ve videosekvenci odehrává blízko hranic snímků, může se nejlepší shoda bloků právě kódovaného snímku a snímku referenčního vyskytovat částečně za hranicemi snímku. Proto se hledají způsoby jak zajistit, aby vektory pohybu mohly směrovat za hranice snímku. Standard H.263 [11] toto řeší rozšířením referenčního snímku extrapolováním hodnot z jeho okrajů. Tento způsob je graficky znázorněn na jednoduchém příkladu na obr. 3.8.



Obr. 3.8: Rozšíření referenčního snímku pro vyhledání vektorů pohybu bez hranic.

3.5 Typy predikování

Referenčním snímkem při predikování nemusí být nutně snímek minulý. Lepší komprese lze v některých případech dosáhnout při výběru z více referenčních snímků. Referenční snímky také nemusí být pouze snímky předešlé, ale i snímky, které budou teprve následovat. Způsoby predikování jsou znázorněny na obr. 3.9.

3.5.1 Dopředná predikce

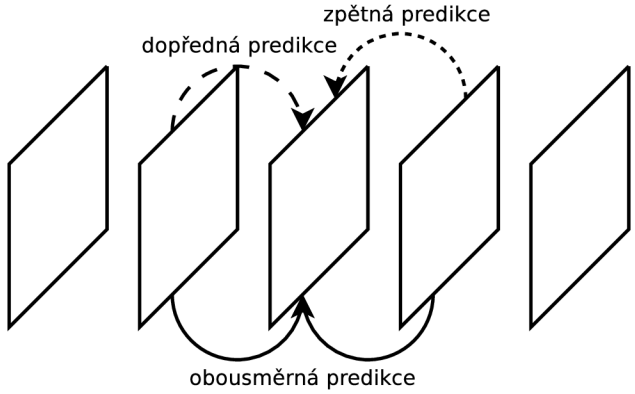
Dopředná predikce používá jako referenční snímek ten, který je v časovém pořadí starší, tedy snímek minulý. Toto má nevýhodu v případech, kdy v právě kódovaném snímku je obraz oproti snímku předešlém velmi pozměněn a jejich vzájemná korelace je minimální. Taková situace může nastat např. při střihu, kdy minulý snímek může obsahovat naprosto odlišné informace.

3.5.2 Zpětná predikce

Pokud bychom kódovali snímky mimo časové pořadí, bylo by možné předejít nedostatkům dopředné predikce. Zpětná predikce vychází ze snímku, který je v časovém sledu před právě kódovaným snímkem. K tomu je ovšem nutné uchovávat v paměti více snímků a časově je přeskládávat.

3.5.3 Obousměrná predikce

Při obousměrné predikci je možné brát jako referenční snímek buď snímek minulý, budoucí nebo takový snímek, který vznikl zprůměrováním snímku minulého a budoucího. Tuto metodu obsahují téměř všechny standardy pro kódování videosekvencí.



Obr. 3.9: Způsoby predikování.

4 STANDARDY MPEG

Skupina MPEG je pracovní skupinou mezinárodní standardizační organizace ISO [8]. Jejím smyslem je vývoj standardů pro kompresi zvuku a pohyblivého obrazu a jejich přenos [17]. Od svého založení v roce 1988 vytvořila několik celosvětově používaných standardů.

Společným jmenovatelem veškerých standardů jsou:

- kodér je složitější než dekodér,
- standardizován je pouze dekodér,
- struktura kodéru je nestandardizována,
- je definován formát bitového proudu a jeho sémantika.

Tyto vlastnosti umožňují vyvíjení kompresních algoritmů při zachování kompatibility se stávajícími dekodéry. Jeden dekodér je tak schopen dekódovat zakódovaný bitový tok jakýmkoliv kodérem všech výrobců.

4.1 MPEG-1

Tento velmi úspěšný standard nesoucí označení ISO/IEC 11172 byl navržen pro kompresi pohyblivého obrazu a jeho zvuku pro digitální datové nosiče. Primárním nosičem bylo plánováno CD s maximálním datovým tokem 1,5 Mbit/s. Bylo požadováno, aby bylo možné přistupovat k videu náhodně v čase maximálně po půl vteřinách. Dalším požadavkem kromě minimální složitosti dekodéru, byl samozřejmě dobrý poměr kvality obrazu a efektivity komprese.

MPEG-1 je používán pod komerčními názvy Video-CD, SVCD a může být použit také pro DVD Video s nízkou kvalitou obrazu [23]. Před příchodem kodeku MPEG-2 [7] byl také používán pro kabelové TV a digitální satelitní vysílání. Zvuk je možné komprimovat i do dvou stop. Velice známým kodekem pro kódování zvuku, který je obsažený v tomto standardu je MPEG-1 Audio Layer III (MP3), viz [23].

4.1.1 Vstupní formát

MPEG-1 existuje ve variantě se snímkovou frekvencí 24 Hz při rozlišení 352×288 obrazových bodů (formát CIF) a 30 Hz při rozlišení 352×240 pixelů. V obou variantách podporuje pouze progresivní řádkování [21] s vzorkováním typu 4:2:0. Každý vstupní snímek je rozdělen do základních elementů zvaných *makrobloky*, jenž odpovídají části snímku o velikosti 16×16 pixelů. Jednotlivé makrobloky pak vstupují do vlastního procesu komprese. Jelikož pracujeme v barevném modelu YC_BC_R ve formátu 4:2:0, sestává se makroblok z bloku 16×16 pixelů jasové složky Y a z bloků

8×8 pixelů chrominančních složek C_B a C_R. Chrominanční složky mají poloviční velikost, protože byly podvzorkovány v horizontálním i vertikálním směru, viz kapitola 1.5.

4.1.2 Typy snímků

Každý snímek je kódován jedním ze tří způsobů a to jako *I-*, *P-* nebo *B-snímek*. V definici standardu je popsán i snímek typu D, který obsahuje pouze DC koeficienty. Tyto snímky nelze s ostatními snímky slučovat a jsou využívány při rychlém vyhledávání. I-snímek je kódován pouze pomocí intra-kódování bez jakékoli kompenzace pohybu. Podobný snímek bychom získali pomocí kodéru JPEG. Kompresní poměr je tedy oproti ostatním typům snímků velice nízký. Proto se I-snímků snažíme používat ve videosekvenci co nejméně. I-snímek slouží jako referenční snímek pro snímky jiného typu.

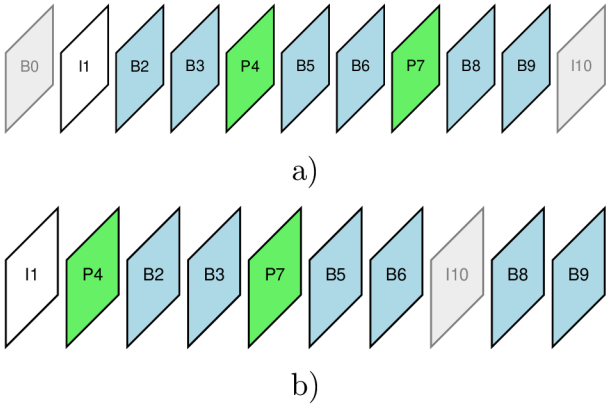
Inter-kódování a kompenzace pohybu je použita při kódování P-snímku. Jedná se o predikovaný snímek dopřednou predikcí. Jako referenční snímek může být použit předcházející I- nebo P-snímek. Kompresní poměr tohoto typu snímku je vzhledem k I-snímku mnohem vyšší. P-snímek tedy může sloužit jako referenční snímek pro P- nebo B-snímky.

Nejlepšího kompresního poměru dosahuje B-snímek, jenž používá opět inter-kódování a kompenzaci pohybu. Jako reference mu slouží předcházející I- a/nebo P-snímek. Pro každý makroblok jsou vygenerovány dva vektory pohybu. První pro dopřednou predikci a druhý pro zpětnou predikci. Predikovaný makroblok tak může vzniknout třemi způsoby:

1. dopřednou predikcí za použití vektoru pohybu pro dopřednou predikci,
2. zpětnou predikcí za použití vektoru pohybu pro zpětnou predikci,
3. obousměrnou predikcí za použití průměru dopředné a zpětné predikce.

Algoritmus vybere tu možnost, jenž obsahuje nejmenší energii v diferenčním makrobloku. Výpočetní náročnost tohoto procesu je velmi vysoká. B-snímek se nepoužívá jako referenční snímek.

Snímky jsou shlukovány do skupin, které tvoří sekvenci videa určité délky. Skupina snímků se nazývá GOP – Group of Pictures a obvykle obsahuje snímky v pořadí IBBPBBPBB. Velikost GOP je definována jako vzdálenost dvou I-snímků. I-snímek může sloužit jako resynchronizační bod videosekvence a aby byl dodržen požadavek na náhodný přístup k videu po 0,5 s, je při snímkové frekvenci 24 Hz maximální možná délka GOP 12 snímků. Složení GOP není pevně definované. Je však potřeba, aby GOP začínala I-snímkem, jenž nevyžaduje k zakódování či dekódování žádnou předchozí referenci. K zakódování B-snímku je potřeba mít v paměti nejprve jeho sousední I- a P-snímek. Obr. 4.1 a) zobrazuje příklad nejčastěji se vyskytující



Obr. 4.1: a) Pořadí MPEG-1 snímků v jakém jsou zobrazovány, b) pořadí MPEG-1 snímků v jakém jsou vysílány.

sekvence snímků jak jsou zobrazovány, obr. 4.1 b) pořadí v jakém jsou vysílány. K zakódování snímku B_2 je zapotřebí mít zpracované snímky I_1 a P_4 a uložené v paměti. Poté je možné zakódovat obousměrnou predikcí snímky B_2 a B_3. Nutnost uchovávání snímků v paměti zavádí zpoždění, které je tak velké, kolik B-snímků je mezi snímky jiného druhu.

Pořadí vysílání snímků tedy není IBBPBBPBB, což by odpovídalo časovému sledu, nýbrž IPBBPBBIBB. Tento sled je pořadí, v jakém jsou snímky kódovány. Dekodér musí před zobrazením snímků provést jejich přeuspořádání, aby zobrazení probíhalo ve správném pořadí. I-snímek je dekódován a rovnou zobrazen. Následující snímek P_4 je dekódován a uložen do paměti. Nyní jsou k dispozici oba referenční snímky potřebné k dekódování snímků B_2 a B_3, které tak mohou být ihned dekódovány a zobrazeny. Teprve po nich je možné zobrazit uložený snímek P_4. Postup dekódování a zobrazení je přehledně uveden v tabulce 4.1.

Tab. 4.1: Pořadí zobrazení a dekódování jednotlivých snímků MPEG-1.

Dekódování	Zobrazení
I_1	I_1
P_4	–
B_2	B_2
B_3	B_3
–	P_4
P_7	–
B_5	B_5
atd.	atd.

4.1.3 Hierarchie videosekvence

Struktura videosekvence podle standardu MPEG-1 má pevnou hierarchii, kterou popisuje obr. 4.2. Tento model lze rozdělit do šesti vrstev, kde obecně každá vrstva symbolizuje určitý ucelený soubor prvků.

Nejvyšší vrstvou je sekvence snímků, která tvoří videosekvenci o určité době trvání. V záhlaví této vrstvy jsou uvedeny základní informace o videu jako je rozlišení, snímková frekvence, použité kvantizační tabulky apod. Videosekvenci lze rozdělit do skupin snímků GOP.

Skupinu snímků GOP tvoří minimálně jeden I-snímek následovaný několika snímky typu P a B. Kodér nemusí nutně udržovat stále stejnou strukturu GOP a její složení může být závislé na okolnostech. Typická délka GOP je 0,5 s.

Následuje vrstva, kterou tvoří jeden snímek. MPEG-1 podporuje pouze progresivní řádkování [21]. Záhlaví této vrstvy nese mimo jiné informace o typu snímku (I, P nebo B) a údaj, kdy má být snímek zobrazen. Každý snímek je tvořen minimálně jednou posloupností makrobloků zvanou *proužek*.

Proužek představuje další vrstvu modelu. MPEG-1 definuje maximální velikost jednoho proužku jako velikost celého snímku. Minimální velikost stanovena není a je volena kodérem. Důležité je, že jednotlivé proužky jsou na sobě nezávislé a pokud se během dekódování vyskytne chyba, je ovlivněn pouze jeden proužek. Dekodér může započíst nové dekódování následujícím proužkem. V hlavičce nalezneme např. informace o poloze daného proužku [23].

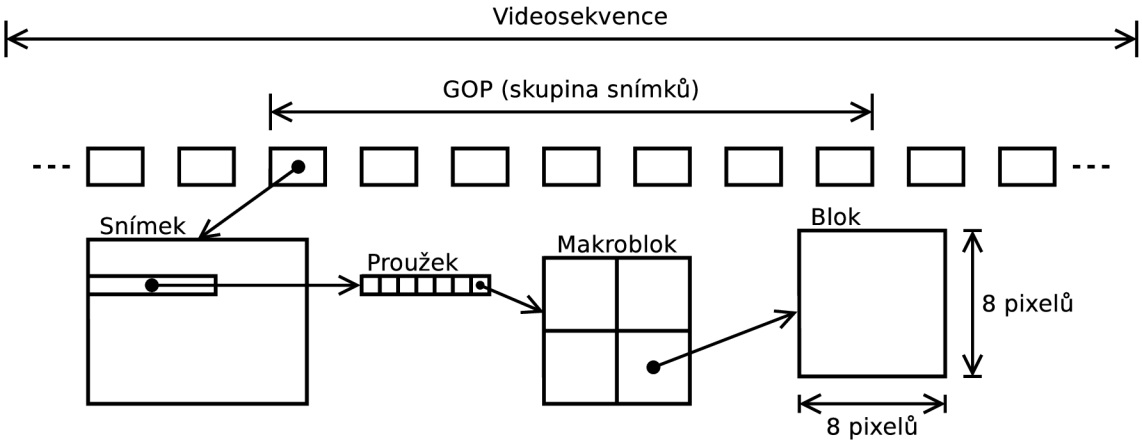
Další vrstvou je makroblok, který nese kompletní informace o dané oblasti snímku. Makroblok se skládá ze šesti bloků (4 jasové a 2 chrominanční). V záhlaví této vrstvy nalezneme např. vektory pohybu. Typ snímku (I,P nebo B) určuje výchozí způsob predikce každého makrobloku. Pokud ovšem není nalezena dobrá shoda v referenčním snímku, může být způsob predikce makrobloku změněn na intra-kódování, nebo může být i kódován bez jakékoli kompenzace pohybu.

Poslední vrstvou je samotný blok, který je tvořen oblastí 8×8 pixelů.

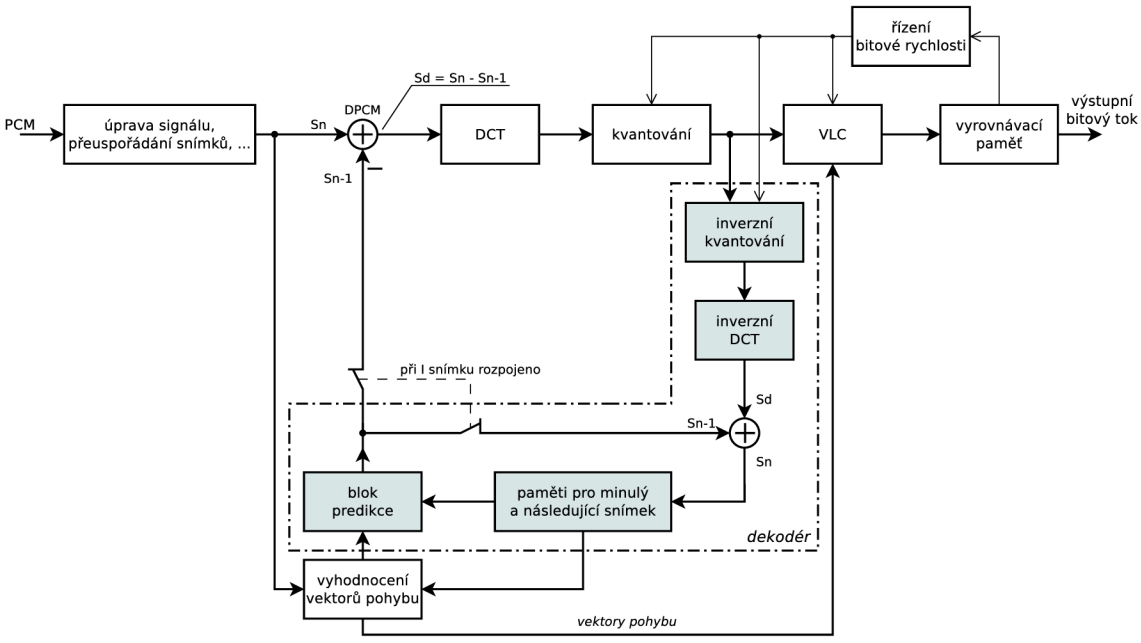
4.1.4 Kodér a dekodér

Zjednodušené blokové schéma kodéru pro MPEG-1 je na obr. 4.3. To, že kodér v sobě zároveň obsahuje i dekodér, ukazuje, proč je kodér složitější než dekodér.

Prvním blokem je úprava signálu a přeuspořádání snímků. Obraz je upraven na standard CIF se vzorkováním 4:2:0. Jako první je kódován I-snímek, u kterého není použito žádné predikování – spínač v kodéru je rozpojen. Po DCT jsou makrobloky kvantovány kvantovací tabulkou. Kvantovací tabulka může být měněna blokem řízení bitové rychlosti podle aktuálního stavu výstupní vyrovnávací paměti. Pokud se blíží její zahlcení, přepne se na kvantovací tabulku s vyššími koeficienty. Toto zaručí,



Obr. 4.2: Hierarchie kodeku MPEG-1.



Obr. 4.3: Zjednodušené blokové schéma kodéru MPEG-1.

že po kvantování zůstane v makrobloku méně nenulových koeficientů. Tento proces se pochopitelně projeví ve kvalitě výsledného obrazu. Tento stav je většinou krátkodobý, protože jakmile se paměť uvolní, přepne se na původní kvantovací tabulku a kvalita se vrátí na předcházející úroveň. Po procesu kvantování probíhá kódování s proměnnou délkou slova VLC (Variable Lenght Coding) a k sériovému bitovému toku jsou přidány informace pro záhlaví. Více v [4].

Snímek, který projde kvantováním, je navíc v části dekodéru dekódován zpět do původní podoby. Tzn. inverzně kvantován a následně převeden do prostorové oblasti zpětnou kosinovou transformací IDCT. Takto získaný snímek je uložen do paměti, kde bude sloužit jako referenční snímek.

Nyní je k dispozici referenční snímek, takže kodér může zvolit jako další snímek jeden z typů snímků s predikcí (P- nebo B-snímek). Se vstupním snímkem a snímkem referenčním je provedena predikce a jsou nalezeny vektory pohybu. Pomocí vektorů pohybu se vytvoří predikovaný snímek. Ten je důležitý pro získání rozdílového snímku, který vznikne jako rozdíl mezi vstupním snímkem a snímkem predikovaným. Dále je tedy zpracováván pouze rozdílový snímek snímku vstupního a predikovaného, což minimalizuje počet nenulových koeficientů po provedení DCT a kvantizace. To vede ke snížení bitového toku. V části dekodéru je opět snímek rekonstruován a uložen jako další referenční snímek pro další predikování.

4.2 MPEG-2

Standard ISO/IEC 13818, známý pod označením MPEG-2, byl navržen organizací ISO jak pro ukládání video a přidruženého audio signálu na digitální média, tak i pro jeho přenos ve ztrátovém prostředí. Požadavkem byla zpětná kompatibilita se standardem MPEG-1, ale zároveň i podpora různých formátů a rozlišení videa. Přestože maximální výstupní bitový tok z kodéru je podle standardu 492 Gb/s, kodér dosahuje řádově nižších hodnot, typicky kolem 5 Mbit/s [7].

Kodek MPEG-2 vznikl jako následovník kodeku MPEG-1 a v současnosti je široce používán např. na DVD nosičích či pro digitální televizní vysílání. Mezi hlavní změny oproti MPEG-1 patří:

- podpora vyššího rozlišení,
- podpora prokládaného řádkování,
- oddělení kvantovacích tabulek pro jasovou a chrominanční složky,
- možnost *škálovatelnosti*,
- proužek nesmí opustit řádek snímku,
- alternativní vyčítání koeficientů z makrobloku,
- vypuštění snímku typu D.

Tab. 4.2: Přehled nejpoužívanějších profilů a úrovní kodeku MPEG-2.

Profily / **Úrovně**	jednoduchý	hlavní	odstupňovaný podle SNR	prostorově odstupňovaný	vysoký
vysoká	-	HDTV 80 Mbit/s	-	-	HDTV 100 Mbit/s
vysoká 1440	-	1440 × 1152 60 Mbit/s	-	1440 × 1152 80 Mbit/s	1440 × 1152 80 Mbit/s
hlavní	SDTV 15 Mbit/s	SDTV 15 Mbit/s	SDTV 15 Mbit/s	-	SDTV 20 Mbit/s
nízká	-	CIF 4 Mbit/s	CIF 4 Mbit/s	-	-

4.2.1 Profily a úrovně

Kodek MPEG-2 může pracovat v několika *profilech* a *úrovních*. Podle úrovně určujeme obrazové rozlišení videosignálu. Profil má souvislost s použitou metodou kódování a tudíž má vliv i na složitost kodéru a dekodéru. Rozlišení videa může být v rozmezí CIF až HDTV a vzorkování obrazu 4:2:0, 4:2:2 i 4:4:4. To, jaký bude formát výsledného videosignálu a jaké funkcionality kodeku budou dostupné, určuje zvolený profil a úroveň. Standard MPEG-2 definuje několik doporučení jaké kombinace profil-úroveň používat [7]. Přehled nejpoužívanějších profilů a úrovní s jejich rozlišením a orientačním datovým tokem je uveden v tabulce 4.2.

Jednoduchý profil používá pouze snímky typu I a P při vzorkování 4:2:0. Tímto je výrazně redukována složitost kodéru a dekodéru, protože získání B-snímků je výpočetně velice náročné. Výsledná implementace je tak velice jednoduchá. Absence B-snímků však výrazně snižuje dosažitelný kompresní poměr a pro přenos takto kódovaného videa je potřeba větší šířka pásma.

Hlavní profil zpřístupňuje všechny stěžejní funkce kodeku MPEG-2. Povoluje snímky typu I,P i B a podporuje prokládané řádkování. Formát vzorkování je pouze 4:2:0. Tento profil společně s *hlavní úrovní* je nejpoužívanější kombinací profil-úroveň. Tato kombinace je používána v digitálním televizním vysílání.

Odstupňovaný profil podle SNR (Signal to Noise Ratio – poměr signálu k šumu) je stejný jako hlavní profil. Zavádí však škálovatelnost podle SNR, viz dále.

Prostorově odstupňovaný profil je opět stejný jako hlavní profil. Změnou je zavedení prostorové škálovatelnosti.

Vysoký profil je nejsofistikovanější profil, který dovoluje použít veškeré funkcionality kodeku a to i při vzorkování 4:2:2. Složitost kodéru či dekodéru s tímto

profilem je velmi velká. Kombinaci vysoký profil a *vysoká úroveň* nalezneme např. v televizním vysíláním s vysokým rozlišením.

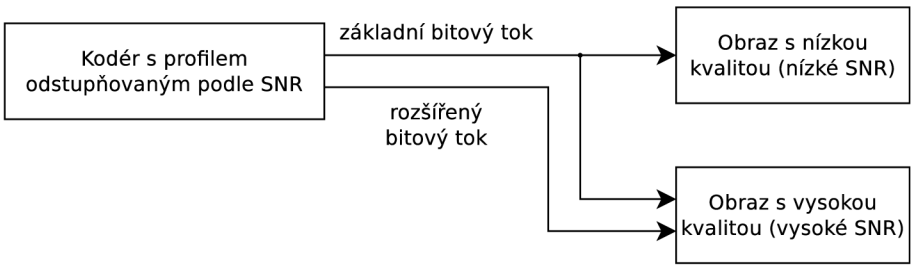
4.2.2 Škálovatelnost

Různé aplikace mají různé potřeby, např. na kvalitu nebo rozlišení výsledného dekódovaného obrazu. Také propustnost přenosového kanálu bývá odlišná. Abychom nemuseli pro každou aplikaci kódovat video signál zvlášť, byla zavedena tzv. *škálovatelnost* [7].

Škálovatelnost je technika, jenž umožňuje kódovat video ve více formátech do jednoho bitového toku. Takto kódovaný bitový tok se skládá z několika vrstev. Jedné základní a jedné nebo více rozšiřujících. Základní vrstva poskytuje omezenou kvalitu. Přidáním informací z vyšších vrstev můžeme kvalitu základní vrstvy rozšířit. Platí, že základní vrstva je kódována nezávisle na ostatních, zatímco vyšší vrstvy jsou na sobě závislé. To dovoluje koexistenci levných a nízkovýkonných dekodérů s dekodéry, které mají vyšší výkon a mohou tak zpracovávat i informace z vyšších vrstev a poskytnout tak vyšší kvalitu obrazu. Přitom všechny dekodéry přijímají stejný bitový tok. V případě, že základní vrstva bude kódována podle standardu MPEG-1, můžeme dosáhnout zpětné kompatibility s dekodéry MPEG-1. Existuje více způsobů, jak škálovatelnosti dosáhnout. Standard MPEG-2 popisuje 4 způsoby škálovatelnosti, které podporuje vysoký profil kodeku MPEG-2 [10].

První technikou je tzv. *dělení dat* [10]. Tato technika rozděluje data podle jejich priority do dvou kanálů. V každém kanálu je přenášena jedna vrstva bitového toku. Hlavní vrstva, která je přiřazena do méně chybového kanálu, obsahuje nejdůležitější informace pro dekódování, jako jsou např. vektory pohybu, nízkofrekvenční kmitočtové koeficienty, řídící informace apod. Rozšiřující vrstva obsahuje méně kritické informace, které mají relativně menší dopad na kvalitu výsledného obrazu, a proto může být umístěna do kanálu, který je méně spolehlivý. Hlavní vrstva je také velmi často doplňována o samoopravné kódy, protože chybovost v hlavní vrstvě má veliké dopady na kvalitu obrazu. Výhodou této metody je nízká režie a jednoduchost.

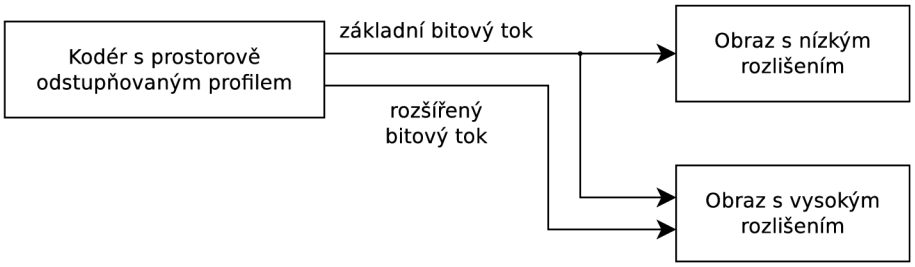
Škálovatelnost podle SNR [10] je obdobou předchozí techniky. Tato technika byla primárně navržena pro přenos videa skrze ztrátové prostředí. Princip techniky je zobrazen na obr. 4.4. Základní vrstva obsahuje pouze hrubě kvantované nízkofrekvenční kmitočtové koeficienty obrazu. Výsledný obraz má velmi nízkou kvalitu, resp. SNR. Vyšší vrstvy obsahují pouze rozdíly k jemněji kvantovaným, popř. méně významným kmitočtovým koeficientům obrazu. Přidáním těchto rozdílů k základní vrstvě dosahujeme vyšší kvality obrazu, resp. vyššího odstupu signálu od šumu v obraze. Všechny vrstvy bitového toku používají stejné prostorové i časové rozlišení. Tento způsob škálovatelnosti vkládá více režie a je složitější než technika dělení dat.



Obr. 4.4: Princip škálovatelnosti podle SNR.

Prostorová škálovatelnost [7] rozděluje bitový tok do vrstev tak, že v základní vrstvě je obraz kódován s malým prostorovým rozlišením a přidáváním informací z vyšších vrstev můžeme dosáhnout zvýšení rozlišení obrazu viz obr. 4.5. Příjemce takto kódovaného videa se může na základě počtu dekódovaných vrstev rozhodnout, jaké rozlišení obrazu je pro něj vyhovující. V případě, že obraz je v základní vrstvě kódován podle jiného standardu, můžeme pomocí prostorové škálovatelnosti zajistit kompatibilitu s jinými standardy jako např. H-261 či MPEG-1. Vyšší vrstvy nemusí sloužit pouze ke zvýšení prostorového rozlišení, ale mohou změnit typ z prokládaného videa na neprokládané a naopak. Proto existují 4 varianty prostorové škálovatelnosti. Více v [10].

Posledním nejpoužívanějším způsobem, který standard MPEG-2 podporuje je *časová škálovatelnost*. V tomto způsobu škálovatelnosti poskytuje základní vrstva určitý počet snímků zobrazených za jednu vteřinu. Vyšší vrstvy rozšiřují tento počet o mezisnímky a dochází tak k navýšení počtu zobrazených snímků za jednu vteřinu. Protože můžeme mít obraz zobrazován prokládaně nebo neprokládaně existuje více variant tohoto způsobu škálovatelnosti [6].



Obr. 4.5: Princip prostorové škálovatelnosti.

4.3 MPEG-4

Novou filozofii kódování pohyblivých obrazů představila skupina MPEG v roce 1998 v novém standardu nesoucí označení ISO/IEC 14496 [23] [1]. Specifikace MPEG-4 popisuje více částí zabývající se kromě kódování obrazu i kódováním zvuku, řízením bitového toku, popisem scény a interakcí uživatele s ní atd. Zpracování obrazu popisuje MPEG-4 Part 2, známý též jako *MPEG-4 Visual* a MPEG-4 Part 10 nazývaný *MPEG-4 AVC* [23]. Záměrem nového standardu bylo vyvinout kodek, který by při dobré kvalitě obrazu produkoval velmi nízký datový tok.

4.3.1 Video objekty

Předchozí standardy mají „klasický“ pohled na kódování obrazu a obraz kódují postupně po makroblocích. V praxi je ale tento přístup málo efektivní, protože objekty v obraze nejsou zarovnány zároveň s makrobloky. Nový standard MPEG-4 umožňuje obraz popisovat z hlediska objektů v obraze. Scéna se tedy může rozdělit na několik video objektů libovolného tvaru existující po určitou dobu [15]. Tyto objekty mohou být „přirozené“–vznikající pomocí kamery a mikrofonu nebo mohou být vytvořeny uměle–generovány počítačem. K objektům můžeme přistupovat nezávisle na sobě. Pro efektivní kódování je pro tento standard důležité vědět, jak snímaná scéna vznikla. Např. snímání rotujícího trojrozměrného tělesa generuje v po sobě jdoucích snímcích velké změny. MPEG-2 a podobné starší standardy na tyto diference špatně reagují a výsledkem je snížení efektivity kódování, resp. generování vyššího bitového toku. Pokud ovšem víme jak scéna vznikla, stačí nám k dekodéru přenést pouze informace pro rekonstrukci objektu a rotaci poté popíší vektory pohybu. Datový tok je pak několikanásobně nižší.

Video objekt je tvořen pomocí tzv. *VOP* (Video Object Plane) [23]. Standard MPEG-4 používá toto označení pro instanci video objektu v určitém čase. Pokud je celý snímek pohyblivého obrazu kódován jako jeden pravoúhlý VOP, odpovídá VOP pojmu snímek ze standardu MPEG-1 a MPEG-2 [18]. Ve standardu jsou definovány tyto druhy VOP:

- I-VOP: vzniká kódováním bez predikce z ostatních snímků,
- P-VOP: VOP predikovaný z předchozích I-VOP nebo P-VOP s půlpixelovou přesností,
- B-VOP: VOP podporovaný ve složitějších profilech vytvořený zpětnou, dopřednou nebo obousměrnou predickí pomocí inter-predikce z předchozích nebo z následujících VOP.

[1] Vývoj standardu MPEG-3 byl pozastaven a nebyl zaveden do praxe. Jeho přínosné vlastnosti byly zakomponovány do standardu MPEG-2.

4.3.2 Profily a úrovně

MPEG-4 dále rozvíjí myšlenku standardu MPEG-2, který přišel s flexibilitou kódování podle zvoleného profilu a úrovně kodeku. Oproti MPEG-2 zavádí větší flexibilitu ve zvolených kódovacích nástrojích. Navíc umožňuje budoucí rozšiřování o nové nástroje a další funkcionality. Každý profil zpřístupňuje určité množství kódovacích nástrojů a funkcionalit. Hlavním nástrojem pro kódování videa v MPEG-4 je nástroj nesoucí označení *VLBV* (Very Low Bit-rate Video – video s velmi nízkým bitovým tokem).

Nástroj pro kódování videa s velmi nízkým bitovým tokem kóduje video velmi podobně jako standard H.263, který je popsán v kapitole 5.2, a za určitých podmínek je výsledný bitový tok přímo identický [18]. Tuto kompatibilitu zpřístupňuje mód krátké hlavičky – *short header mode*. V tomto profilu může mít video rozlišení sub-QCIF až 16CIF se snímkovou frekvencí až 30 snímků/s. Používá se vzorkování typu 4:2:0 a stejně jako v ostatních standardech se pracuje v barevném modelu YC_BC_R. Tento profil nepodporuje P-VOB a velmi tím redukuje složitost kodéru.

V současné době existuje velké množství navržených profilů [15]. Mezi nejčastěji podporované nástroje patří:

- mód krátké hlavičky,
- prostorová i časová škálovatelnost,
- dělění dat,
- prokládání,
- podpora všech typů VOP,
- resynchronizace proužků.

Méně často používané nástroje jsou např. čtvrt-pixelová přesnost, globální pohybová kompenzace, úprava zrnitosti, ...

5 STANDARDY ITU

Paralelně se standardy vytvářené organizací ISO byly vytvářeny standardy organizací ITU (International Telecommunication Union – mezinárodní telekomunikační unie). Standardy organizace ITU pro zpracování pohyblivých obrazů patří do skupiny standardů nesoucí označení H.26x. Hlavním prvkem ve kterém se tyto standardy odlišovaly, a to hlavně v dřívějších dobách, bylo zaměření na kódování použitelné v reálném čase k jednosměrné nebo obousměrné videokomunikaci.

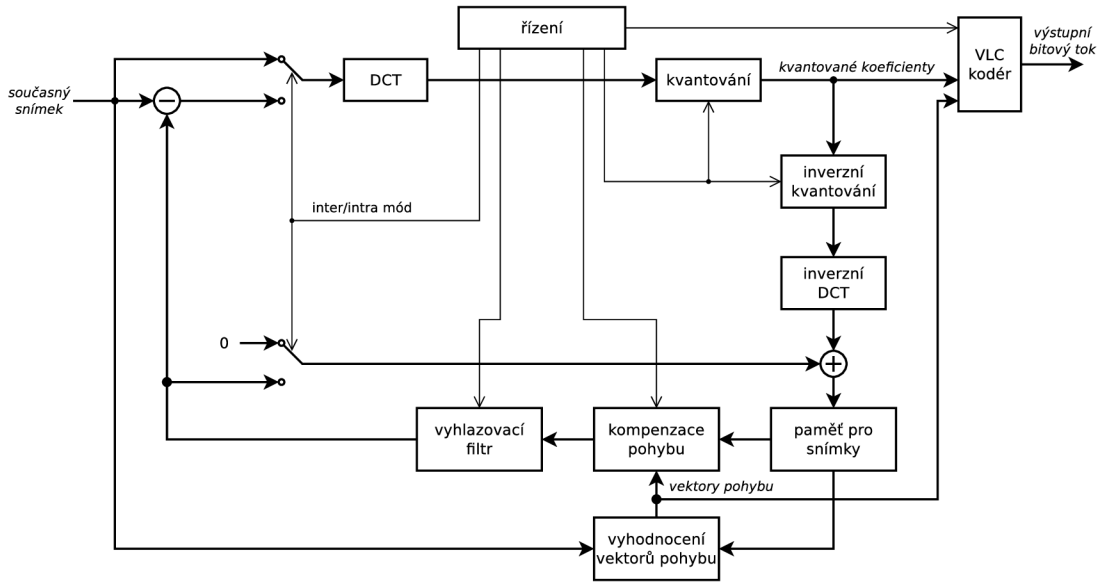
5.1 H.261

Prvním ze standardů pro zpracování pohyblivých obrazů, který organizace ITU představila, byl standard H.261. Tento standard vznikl na přelomu 80. a 90. let 20. století a byl vyvinut za účelem přenosu videokonference prostřednictvím linky ISDN (Integrated Services Digital Network – digitální síť integrovaných služeb). ISDN garantuje dostupnou šířku pásma v celočíselných násobcích 64 kbit/s s nízkým zpožděním. Výstupní datový tok H.261 se pohybuje mezi 40 kbit/s až 2 Mbit/s [9]. Této bitové rychlosti je částečně dosaženo díky tomu, že H.261 podporuje pouze neprokládané video se snímkovým kmitočtem 30 snímků/s a s rozlišením QCIF nebo CIF.

Princip kódování je velice podobný principu kodéru MPEG-1. Kodér je vyobrazen na obr. 5.1. Obraz je nejprve transformován do barevného modelu YC_BC_R se vzorkováním typu 4:2:0. Základní jednotkou kodéru je makroblok, který se skládá ze 4 bloků jasových koeficientů a dvou bloků s chrominančními koeficienty. Velikost bloku je 8×8 bodů. Dále jsou jednotlivé bloky podrobovány dvourozměrné DCT transformaci a následně vyčítány způsobem cik-cak. Vyčítané koeficienty jsou kódovány s proměnnou délkou slova.

Neobvyklým blokem kodéru je vyhlazovací filtr tzv. *loop filter* [11]. Jedná se o dvourozměrnou dolní propust, která má za úkol vyhladit vysoké kmitočtové změny v prostorové oblasti snímku a odstranit tak projev blokového artefaktu. Vstupem filtru je predikovaný snímek, který je kompenzován vektory pohybu. Kodér umožňuje kompenzaci pohybu s jedním vektorem pohybu na makroblok s vyhledávací oblastí o velikosti 15 pixelů. Volitelným prvkem je také zabezpečení dopředným korekčním kódováním pomocí kódu BCH (Bose-Chaudhuri-Hocquenghem) [6]. Této vlastnosti se v praxi příliš nevyužívá, protože zabezpečení protichybovým kódem zvyšuje bitový tok a chybovost ISDN linky je nízká.

Další předností tohoto standardu jsou nízké požadavky na výpočetní výkon a snadná implementace. Tyto faktory hrály v době vzniku tohoto kodeku významnou roli. Navíc umožnil komunikovat napříč zeměmi, které měly odlišné televizní standardy. Proto našel uplatnění v mnoha konferenčních aplikacích.



Obr. 5.1: Blokové schéma kodéru H.261.

5.2 H.263

Doporučení H.263 popisuje kódování pro komprimování pohyblivých obrazů s nízkým bitovým tokem [11]. Základní konfigurace vychází z doporučení H.261. Oproti tomuto doporučení dosahuje podstatně lepšího kompresního poměru a poskytuje větší flexibilitu možností kódování [18]. Kodér může pracovat v jednom z pěti standardizovaných obrazových formátů: sub-QCIF, QCIF, CIF, 4CIF a 16CIF.

Největší rozdíl mezi doporučeními H.261 a H.263 je v implementaci kompenzace pohybu. Oproti doporučení H.261, které používá kompenzaci pohybu s přesností na jeden pixel a vyhlazovací filtr, dovoluje doporučení H.263 použít kompenzaci pohybu s půl-pixelovou přesností, ovšem vyhlazovací filtr nepodporuje. Kodér H.263 umožňuje použít pro kompenzaci pohybu jeden nebo čtyři vektory pohybu na makroblok. Použití 4 vektorů pohybu na makroblok má za následek přesnější predikování, čímž je snížen z predikovaného snímku počet informací. Tato technika tedy výrazně snižuje bitový tok. Obraz kódovaný pomocí H.263 obsahuje méně režijních dat. Tato vlastnost také přispívá ke snížení bitového toku při zachování stejné kvality jako při kódování podle H.261.

Původní doporučení je postupně rozšiřováno vydáváním nových verzí doporučení a pomocí *doložek* tvz. *Annex*. Verze 2 označovaná H.263+ a přinesla 12 nových doložek. Verze 3 označována jako H.263++ přinesla další doložky. Celkem jich v poslední verzi nalezneme 24, viz [11]. Doložky definují např. použití 4 vektorů pohybu na makroblok (Annex F), neomezené vektory pohybu (Annex D), škálovatelnost (Annex O) atd.

Doporučení H.263 je obsaženo ve standardu MPEG-4, který je popsán v kapitole 4.3 a je s tímto standardem kompatibilní. Využití nachází hlavně v oblasti videokonferencí, internetovém vysílání či přenosu videa přes bezdrátovou síť 3G.

5.3 H.264

Doporučení H.264 koresponduje se standardem MPEG-4 Part 10 popsaným v kapitole 4.3. Některé prameny jej označují jako H.264/MPEG4 Part 1

6 REALIZACE PROGRAMU

Pro realizaci programu byl zvolen programovací jazyk Java. Tento dnes velmi používaný univerzální programovací jazyk podporuje velké množství pokročilých objektových konceptů a patří mezi nejvyspělejší objektově orientované jazyky současnosti. Velkou výhodou jazyka Java je i přenositelnost programů mezi platformami (Linux/UNIX, Windows, atd.) bez nutnosti překompilování. Oproti jiným programovacím jazykům poskytuje jazyk Java programátorovi mnoho výhod např. v oblasti správy paměti.

Ke spuštění programu napsaného v jazyce Java stačí mít na jakékoli platformě k dispozici volně dostupný JVM (Java Virtual Machine – javový virtuální počítač), který zajišťuje běh přeložených programů. Po překladu programu napsaného v jazyce Java dojde k vytvoření tzv. *bajtového kódu*, který je později spouštěn pomocí JVM, jenž na konkrétní platformě program interpretuje [16].

Součástí aplikačního programového rozhraní jazyka Java jsou standardní knihovní třídy jazyka Java, ve kterých jsou implementovány funkce a třídy pro běžně se vyskytující operace. Tyto knihovny se musí nacházet v každém prostředí, kde je jazyk Java používán, tudíž je není nutno s programem přenášet.

6.1 Prostředky pro implementaci

Výběr prostředků pro implementaci a vývoj programu má velký význam na rychlost a kvalitu vývoje programu. V dnešní době je možné využít velkého množství ať už komerčních či volně dostupných vývojových prostředí a podpůrných prostředků pro vývoj programu.

6.1.1 Vývojové prostředí NetBeans

NetBeans je úspěšný Open Source projekt, založený firmou Sun Microsystems v roce 2000 [22]. Jedním z produktů tohoto projektu je vývojové prostředí NetBeans. Jedná se o bezplatně šířený produkt a jeho užívání není nijak omezeno.

Samotné vývojové prostředí je vytvořeno pomocí jazyku Java. Dnes ovšem podporuje téměř jakýkoliv programovací jazyk. Pro jazyk Java nabízí NetBeans velmi kvalitní nástroje pro psaní programu, jeho překládání i ladění. Výhodou tohoto vývojového prostředí je, že má modulární architekturu, která umožňuje pomocí modulů rozšiřovat funkce vývojového prostředí. Příkladem může být modul pro verzování programu – Subversion [19].

Při práci v prostředí NetBeans je vhodné před vlastním psaním programu založit projekt. Tento projekt lze poté přeložit do spustitelné formy, přičemž se v projektu

vytvoří `.jar` soubor. Tento soubor je možné později distribuovat mezi platformami. V případě, kdy se překládá program, do kterého jsou vloženy externí knihovny ve formě `.jar` souborů, vytvoří se v přeloženém programu pouze odkaz na tyto knihovny. Při spouštění programu je poté nutné mít tyto knihovny – `.jar` soubory ve stejném adresáři jako spouštěný program.

Samotný překlad je tvořen několika kroky. Průběh překladu lze v prostředí NetBeans modifikovat, a to pomocí souboru `build.xml`, který se automaticky vytvoří v kořenovém adresáři projektu při jeho prvním překladu. Před a po vykonání každého kroku překladu je možné překlad v tomto souboru doplnit uživatelským kódem. Pokud např. chceme vytvořit `.jar` soubor, který by obsahoval všechny externí knihovny, stačí upravit sekci kódu, která je volána po vytvoření `.jar` souboru překládaného programu. V případě, kdy se projekt jmenuje Projekt a externí knihovny K1 a K2, může uživatelský kód vypadat např. takto:

```
<target name="-post-jar">
  <jar jarfile="dist/ProgramSKnihovnami.jar">
    <zipfileset src="${dist.jar}" excludes="META-INF/*"/>
    <zipfileset src="src/Projekt/Lib/K1.jar" excludes="META-INF/*"/>
    <zipfileset src="src/Projekt/Lib/K2.jar" excludes="META-INF/*"/>
    <manifest>
      <attribute name="Main-Class" value="${main.class}"/>
    </manifest>
  </jar>
</target>
```

6.1.2 Nástroj TortiseSVN

Při vývoji programu je důležité efektivně spravovat zdrojové kódy projektu. Pro tento účel existuje opět velké množství nástrojů, které se často označují zkratkou SCM (Source Code Management – správa zdrojových kódů). Volně dostupným nástrojem tohoto typu pro platformu Windows je program TortiseSVN. Tento program je dostupný jako grafická nástavba rozšíření shellu Windows [20]. Grafický klient pak využívá příkazy konzolového programu Subversion, označovaný také pouze zkratkou SVN.

TortiseSVN poskytuje jednoduchý nástroj pro správu verzí software. Často je využíván v případech, kdy na jednom projektu pracuje více lidí najednou. Dokáže totiž mimo jiné řešit konflikty, kdy více lidí edituje stejný soubor. Při používání verzovacího programu u projektu, na kterém pracuje pouze jeden programátor, je největší výhodou uchovávání historie jednotlivých změn v programu. Velice snadno tak lze vracet provedené změny v programu apod.

6.1.3 Volně dostupné knihovny

V programu je kromě standardních knihoven jazyka Java použita volně dostupná knihovna s názvem JAMA [12]. Knihovna byla vyvinuta v institutu NIST (National Institude of Standards and Technology – národní institut pro standardy a technologie v USA) společně s universitou v Marylandu a dodnes má dobrou podporu s dobře zpracovanou dokumentací.

Tato knihovna se zaměřuje na funkce pro výpočty v lineární algebře a obsahuje téměř všechny běžně používané funkce pro práci s maticemi. Programátor se tak může plně soustředit na řešení vlastní problematiky a nemusí se zabývat implementacemi operací s maticemi.

6.2 Struktura programu

Koncepce programu vychází ze zásad objektového programování. Seznam vytvořených tříd je vypsán v tabulce 6.1.

Jako první je spouštěna metoda `main()` ze třídy `Main`. Jejím úkolem je pouze inicializace programu a vytvoření instance třídy `GUI`.

Konstruktor třídy `GUI` vykreslí grafické rozhraní, a na základě uživatelově interakce volá patřičné funkce, které obsluhují požadavky uživatele.

Tab. 6.1: Seznam vytvořených tříd s krátkým popisem.

Třída	**Účel**
Debug	ladění, kontrolní výpisy
Frames	zobrazení detailu snímku
GUI	grafické rozhraní pro interakci s uživatelem
ImagePreview	zobrazení miniatury načítaného vstupního obrázku
Main	spuštění programu a jeho inicializace
MVectors	nalezení vektorů pohybu
MVectorsImage	vykreslení vektorů pohybu
MeasureQuality	výpočet objektivního hodnocení kvality obrázků
Messages	vytváření varovných hlášení
QMatrixes	generování kvantizačních matic
TransformMatrixes	generování matic pro výpočet DCT transformace
Transforms	transformace obrazu a operace s ním
ViewPanel	vykreslování snímků

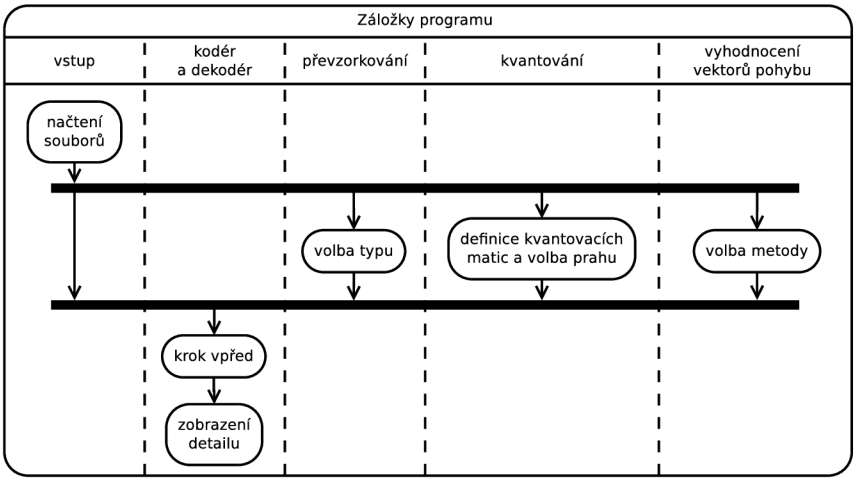
V dalších třídách jsou implementovány funkce pro transformace a úpravy obrazu, výpočty objektivního hodnocení kvality obrazu apod. Jedná ze zejména o třídy `MVectors`, `MeasureQuality` a `Transforms`. V poslední jmenované třídě je implementováno největší množství z potřebných transformací a úprav obrazu a je tedy základním prvkem celého programu. Příklady implementovaných funkcí:

- `createImageFromRGB()` – vytvoří Javový objekt `Image`,
- `dct()` – vypočte diskrétní kosinovu transformaci podle vzorce 2.1,
- `idct()` – vypočte zpětnou diskrétní kosinovu transformaci podle vzorce 2.2,
- `overSample()` – zajišťuje převzorkování obrazu,
- `quantization()` – zajišťuje kvantizaci obrazu,
- `RGBtoYCbCr()` – převede obraz z modelu RGB do modelu YC_BC_R,
- `YCbCrtoRGB()` – převede obraz z modelu YC_BC_R do modelu RGB.

Implementované funkce vycházejí z principů, popsaných v předchozích kapitolách. Příkladem může být převzorkovaní, proces kvantování ilustrovaný na obr. 2.2, výpočet dopředné a zpětné DCT pomocí vzorců 2.1 a 2.2, vyhledávání vektorů pohybu pomocí metod Full-Search a 3-Step-Search, jejichž algoritmy jsou popsány v kapitole 3.2 atd.

Hlavní myšlenkový tok práce s programem je zobrazen na obr. 6.1 pomocí diagramu aktivity jazyka UML (Unified Modeling Language – unifikovaný jazyk pro tvorbu diagramů). Diagramy aktivit mají podle specifikace UML 2 novou sémantiku založenou na Petriho sítích , dříve byly pouze speciálním případem stavových automatů. [1]. Z diagramu je možné vyčíst jaké nastavení lze provádět v dané záložce programu.

Mimo hlavní směr toku může uživatel vstoupit např. prostřednictvím akcí reset a restart. Více viz příloha A.



Obr. 6.1: Diagram aktivity.

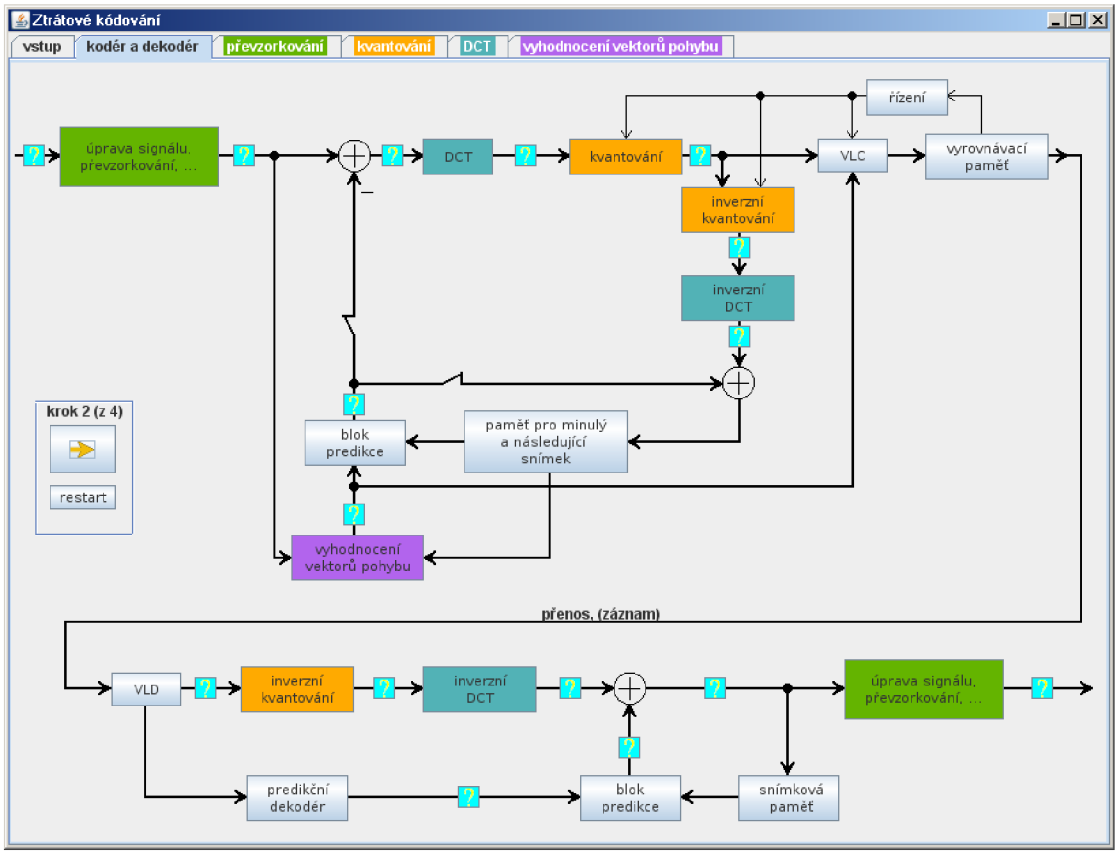
6.3 Uživatelské rozhraní

Základní uživatelské rozhraní je zobrazeno na obr. 6.2. Program se skládá z více částí, mezi kterými lze přepínat pomocí záložek. Podrobnější popis je uveden v příloze A.

6.4 Ukázka výstupu programu

Pomocí tlačítek pro zobrazení detailu, které jsou v programu umístěny vždy na vstupu a výstupu důležitých sekcí kodéru a dekodéru, je možno v programu zobrazit tyto snímky:

- vstupní a výstupní snímek,
- převzorkovaný snímek,
- snímek před a po DCT transformaci,
- snímek po dopředné a zpětné kvantizaci,
- snímek po zpětné DCT transformaci,
- predikovaný snímek a snímek s mapou vektorů pohybu.



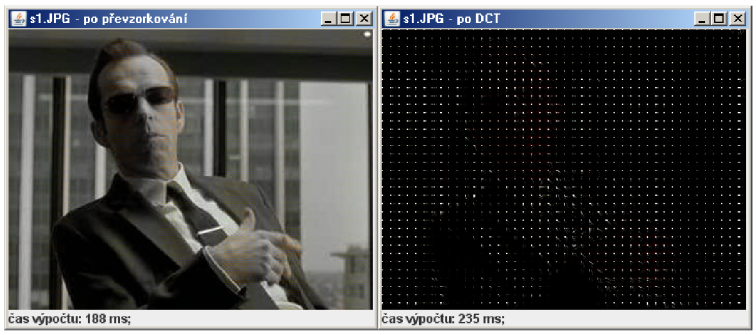
Obr. 6.2: Grafické rozhraní programu.

Ve výchozím nastavení je zvolen typ vzorkování 4:2:0, Full-Search jako metoda pro nalezení vektorů pohybu a není použito prahování. Časy výpočtu uvedené ve stavovém řádku jsou pouze orientační.

Vstupní snímek není nijak upravován. Po jeho zobrazení je možné zobrazit i jednotlivé složky obrazu pomocí volby `Zobrazit`. Tímto způsobem vznikl např. obr. 1.2. Pro názornost byla použita sekvence snímků, která je uvedena na obr. 6.3.

Obr. 6.3: Vstupní sekvence snímků.

Dvojice snímků na obr. 6.4 zobrazuje vstup a výstup bloku realizující dopřednou dvourozměrnou diskrétní kosinovu transformaci. Vstupem je pouze převzorkovaný vstupní snímek. Výstupní snímek vizualizuje kmitočtové koeficienty, jenž jsou výstupem DCT transformace. Na tomto snímku lze dobře pozorovat, jak DCT koncentruje energii transformovaného bloku směrem k DC koeficientu.

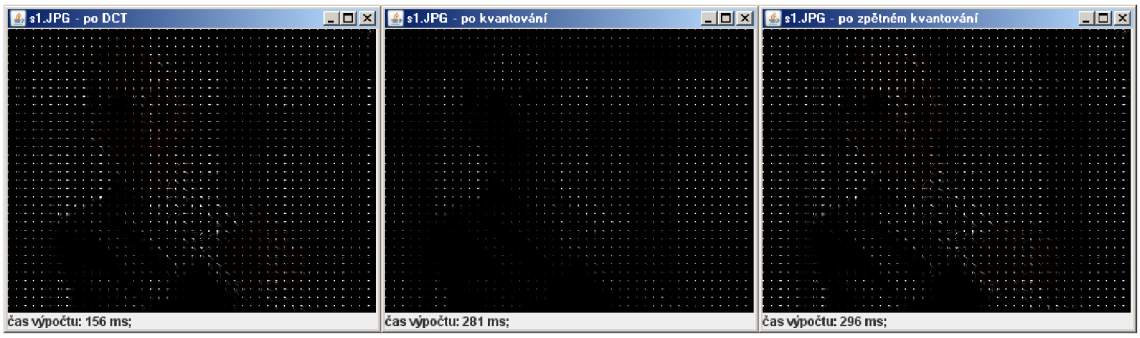


Obr. 6.4: Vstup a výstup bloku realizující dopřednou dvourozměrnou diskrétní kosinovu transformaci.

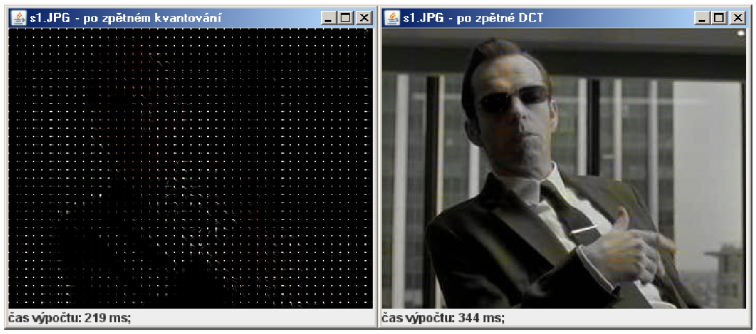
Výstup bloku realizující transformaci DCT je vstupem pro kvantování. Tento proces ovlivňují kvantovací matice, které lze definovat na zvláštní záložce programu. Na obr. 6.5 je zobrazen vstup bloku kvantování, jeho výstup a zároveň i výstup bloku zpětného kvantování. Z obrázku je patrné, že pokud není uplatněno prahování, jedná se o bezeztrátový proces.

Dalším krokem je zpětná DCT transformace. Její vstup a výstup je zobrazen na obr. 6.6. Při porovnání s obr. 6.4 je patrné, že se opět jedná o bezeztrátový proces.

Pokud do procesu kódování vstoupí další snímek, je možné provést kompenzaci pohybu mezi snímkem minulým a právě kódovaným. Mapa vektorů pohybu je zobrazena na obr. 6.7. Díky vektorům pohybu je možné sestavit predikovaný snímek, který zefektivňuje další proces kódování. V mapě vektorů lze pozorovat, že největší změny se v sekvenci snímků odehrávají v pravé dolní oblasti.



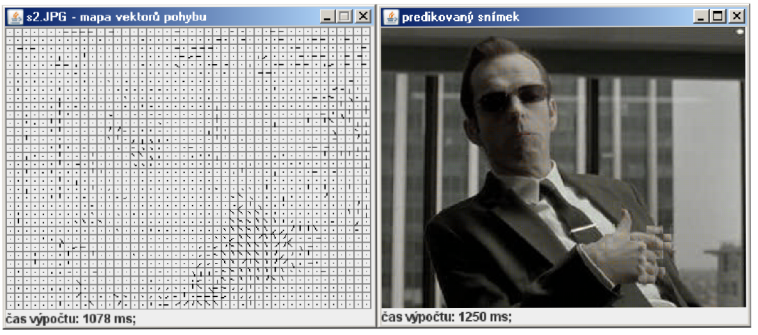
Obr. 6.5: Snímky s DCT koeficienty před kvantováním, po kvantování a po zpětném kvantování.



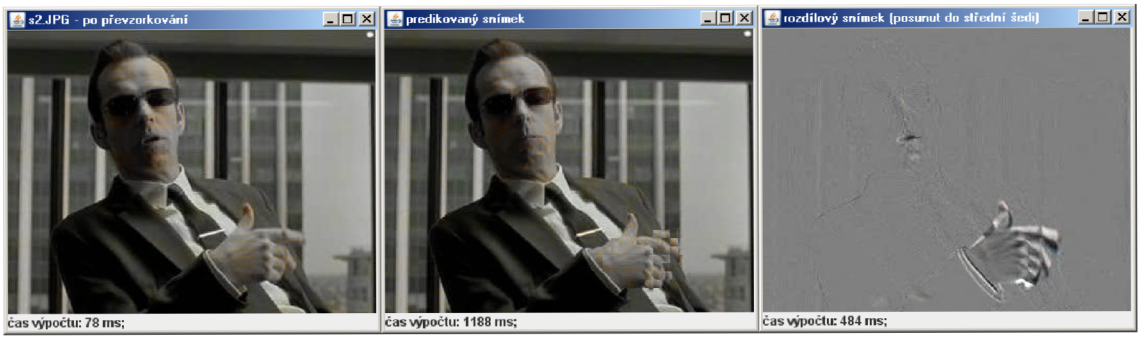
Obr. 6.6: Snímek s DCT koeficienty před zpětnou DCT transformací a po ní.

Na základě predikovaného snímku je možné určit rozdílový snímek, který vznikne odečtením predikovaného a právě kódovaného snímku. Tyto snímky – právě kódovaný, predikovaný a rozdílový jsou zobrazeny na obr. 6.8. Aby byly rozdíly, které jsou přenášeny v predikovaném snímku lépe viditelné, jsou hodnoty posunuty do střední šedi.

Srovnání vstupního a výstupního snímku je zobrazeno na obr. 6.9. Je patrné, že lidským okem není téměř možné nalézt rozdíly. Svědčí o tom i malé hodnoty chyb MAE a MSE a velká hodnota PSNR. Tyto hodnoty jsou uvedeny ve stavovém řádku výstupního snímku.



Obr. 6.7: Mapa vektorů pohybu, pomocí které je sestaven predikovaný snímek.



Obr. 6.8: Právě kódovaný, predikovaný a rozdílový snímek.

Obr. 6.9: Srovnání vstupního a výstupního snímku.

7 ZÁVĚR

Cílem této práce bylo popsat techniky pro komprimaci obrazových dat a videosekvencí. V teoretické části práce jsou popsány obecné techniky jak pro ztrátové, tak hlavně bezeztrátové kódování videosekvencí.

Pro demonstraci kódovacích technik byl vytvořen program v jazyce Java, jenž zobrazuje, jak zpracovávaný snímek vypadá v jednotlivých krocích kódování. Program dovoluje měnit některé parametry kódování a na reálných snímcích videosekvence je možné pozorovat jejich vliv.

V programu jsou naprogramovány obecné metody pro zpracování obrazu a díky nim je možné program lehce upravovat pro případnou demonstraci funkce jiných standardů pro zpracování obrazu.

LITERATURA

[1] Arlow, Neustadt: *UML 2 a unifikovaný proces vývoje aplikací: Průvodce analýzou a návrhem objektově orientovaného softwaru.* 2. vydání. Brno: Computer Press, 2008. 568 s. ISBN 978-80-251-1503-9.

[2] Fraser, Murphy, Bunting: *Správa barev: průvodce profesionála v grafice a prepressu.* Přeložil Milan Daněk. 1. vydání. Brno: Computer Press, 2003. 521 s. ISBN 80-7226-943-7.

[3] Ghanbari, M.: *The cross-search algorithm for motion estimation* [online]. 1990, [cit. 24. 11. 2010]. Dostupné z URL: <http://www.di.unipi.it/ lonetti/PhDCourse/ME/CrossSearch.pdf>.

[4] Gibson, Jerry D.: *Digital compression for multimedia: principles and standards.* 1st ed. San Francisco: Morgan Kaufmann Publishers. 1998. 476 s. ISBN 1-55860-369-7.

[5] Gonzalez, Woods: *Digital image processing.* 3rd ed. Upper Saddle River: Prentice-Hall, Inc. 2008. 954 s. ISBN 0-13-168728-X.

[6] Hanzo, Cherriman, Streit: *Video Compression and Communications: From Basics to H.261, H.263, H.264, MPEG4 for DVB and HSDPA-Style Adaptive Turbo-Transceivers.* 2nd ed. Chichester: John Wiley & Sons, Ltd., 2007. 677 s. ISBN 978-0-470-51849-6.

[7] Haskell, Puri, Netravali: *Digital Video: An Introduction to MPEG-2.* 1st ed. Kluwer Academic Publishers, 2003. 441 s. ISBN 0-412-08411-2.

[8] *International Organization for Standardization* [online]. 2010, [cit. 22. 11. 2010]. Dostupné z URL: <http://www.iso.org>.

[9] ITU-T Recommendation H.261: Video CODEC for audiovisual services at $p \times 64$ kbit/s, 1993.

[10] ITU-T Recommendation H.262: Information Ttechnology – Generic coding of moving pictures and associated audio information: Video, 1995.

[11] ITU-T Recommendation H.263: Video coding for low bit rate communication, Version 2, 2005.

[12] *JAMA: Java Matrix Package* [online]. 2005, [cit. 7. 5. 2011]. Dostupné z URL: <http://math.nist.gov/javanumerics/jama>.

[13] *JPEG resources* [online]. 2007, poslední aktualizace 2007 [cit. 9. 11. 2010]. Dostupné z URL: <http://www.jpeg.org>.

[14] *MPEG resources* [online]. 2010, poslední aktualizace 2010 [cit. 9. 11. 2010]. Dostupné z URL: <http://www.mpeg.org>.

[15] *Overview of the MPEG-4 Standard* [online]. 2002, poslední aktualizace 2002 [cit. 16. 3. 2011]. Dostupné z URL: <http://mpeg.chiariglione.org/standards/mpeg-4/mpeg-4.htm>.

[16] PITNER, T: *Java-začínáme programovat,podrobný průvodce začínajícího uživatele.* 1. vydání. Praha: Grada Publishing. 2002. 222 s. ISBN 80-247-0295-9.

[17] Richardson, Iain E. G.: *H.264 and MPEG-4 Video Compression: Video Coding for Next-generation Multimedia.* 1st ed. Chichester: John Wiley & Sons, Ltd., 2003. 281 s. ISBN 0-470-84837-5.

[18] Richardson, Iain E. G.: *Video Codec Design: Developing Image and Video Compression Systems.* 1st ed. Chichester: John Wiley & Sons, Ltd., 2002. 303 s. ISBN 0-471-48553-5.

[19] *Subversion - NetBeans Plugin detail* [online]. 2010, [cit. 7. 5. 2011]. Dostupné z URL: <http://plugins.netbeans.org/plugin/11492/subversion>.

[20] *TortoiseSVN* [online]. 2011, [cit. 7. 5. 2011]. Dostupné z URL: <http://tortoisesvn.net>.

[21] Vít, V.: *Televizní technika: přenosové barevné soustavy.* 1. vydání. Praha: BEN - technická literatura, 1997. 719 s. ISBN 80-86056-04-X.

[22] *Vítejte u NetBeans* [online]. 2010, [cit. 7. 5. 2011]. Dostupné z URL: <http://netbeans.org/index_cs.html>.

[23] Watkinson, John: *The MPEG Handbook.* 2nd ed. Amsterdam: Focal Press, 2001. 435 s. ISBN 0-240-80578-X.

SEZNAM SYMBOLŮ, VELIČIN A ZKRATEK

BCH	Bose-Chaudhuri-Hocquenghem
CD	Compact Disc
CIF	Common Intermediate Format
DCT	Discrete Cosine Transform – diskrétní kosinova transformace
DPCM	Differential Pulse Code Modulation - rozdílová pulsní kódová modulace
DVB	Digital Video Broadcasting
DWT	Discrete Wavelet Transform – diskrétní waveletova transformace
GOP	Group Of Picturec – skupina snímků
HDTV	High Definition Television – televizní systém s vysokým rozlišením
HVS	Human Visual System – systém lidského vizuálního vnímání
ISDN	Integrated Services Digital Network – digitální síť integrovaných služeb
ISO	International Organization for Standardization
ITU	International Telecommunication Union – mezinárodní telekomunikační unie
JPEG	Joint Photographic Experts Group
JVM	Java Virtual Machine – javový virtuální počítač
MAE	Mean Absolute Error – střední absolutní chyba
MOS	Mean Opinion Score
MPEG	Motion Picture Experts Group
MSE	Mean Square Error – střední kvadratická chyba
NIST	National Institude of Standards and Technology – národní institut pro standardy a technologie v USA
NMSE	Normalized Mean Square Error – normalizovaná střední kvadratická chyba
PSNR	Peak Signal to Noise Ratio – špičkový poměr signálu k šumu
ROI	Region of Interest – oblast zájmu

SAE Sum of Absolute Errors – součet absolutních chyb

SCM Source Code Management – správa zdrojových kódů

SDTV Standard-definition television – televizní systém se standardním rozlišením

SIF Source Input Format

SNR Signal to Noise Ratio – poměr signálu k šumu

UML Unified Modeling Language – unifikovaný jazyk pro tvorbu diagramů

VLBV Very Low Bit-rate Video – video s velmi nízkým bitovým tokem

VLC Variable Lenght Coding – kódování s proměnnou délkou slova

VOP Video Object Plane

SEZNAM PŘÍLOH

A NÁVOD PRO PRÁCI S PROGRAMEM

Grafické rozhraní programu se skládá z několika sekcí, mezi kterými se lze pohybovat pomocí záložek viz obr. A.1.

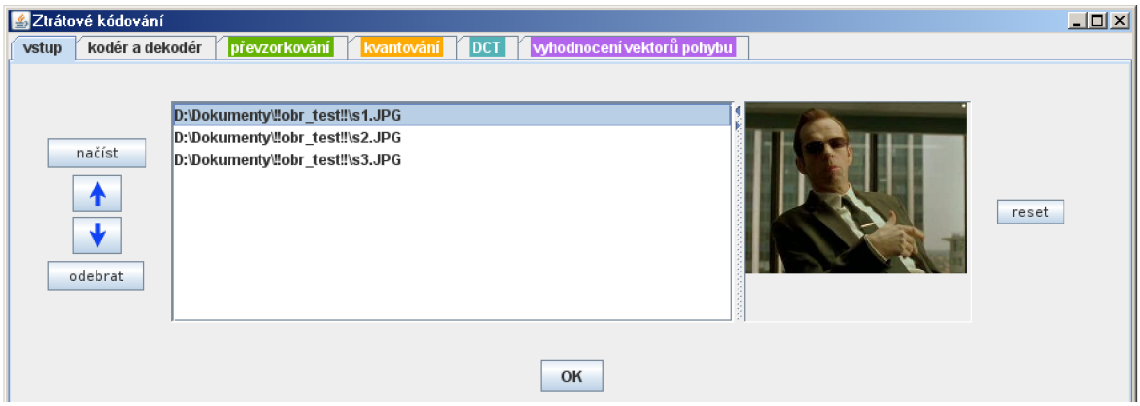
Obr. A.1: Záložky programu.

Záložka vstup: Po spuštění programu je aktivována záložka `vstup` viz obr. A.2. Zde je možno načíst sekvenci snímků se kterou se bude později pracovat. Je dovoleno načítat soubory s příponami `.jpg`, `.jpeg`, `.gif` a `.png`. Pro další práci s programem je potřeba načíst minimálně 2 soubory. Snímky musí mít pochopitelně stejné rozměry, které navíc musí být dělitelné 8 beze zbytku. Validace těchto omezení je kontrolována při spuštění krokování, viz níže.

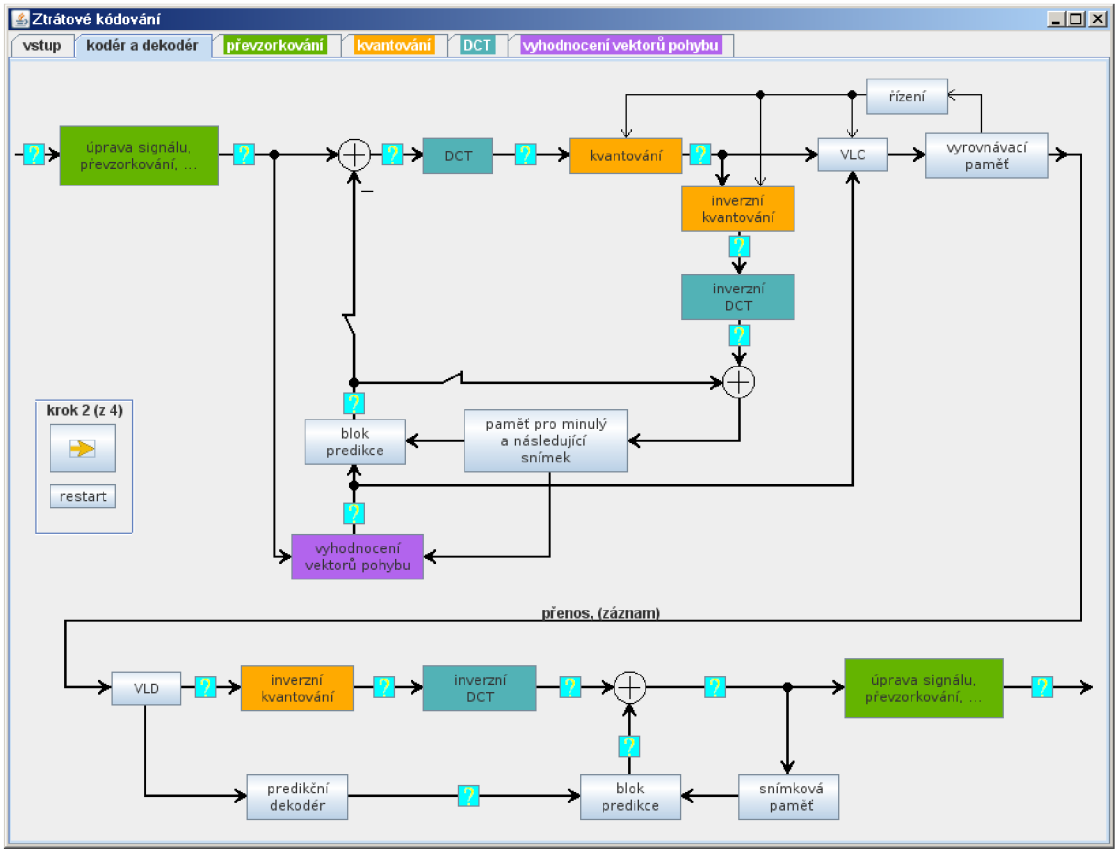
Po načtení souborů dojde k vypsání jejich absolutních cest umístění. Po kliknutí na položku seznamu se v jeho pravé části zobrazí miniatura snímku. Dále je možné přeuspořádat pořadí snímků, popř. některé snímky odebrat.

Jakmile je spuštěno krokování kodéru a dekodéru, jsou tlačítka pro práci se soubory deaktivovány. Aktivaci těchto tlačítek je možné provést pomocí tlačítka `reset`. Tím dojde k resetování celého programu do výchozího nastavení – mj. budou i odstraněny všechny načtené soubory.

Stiskem tlačítka `OK` dojde k přepnutí na záložku `kodér a dekodér`.



Obr. A.2: Prostředí záložky `vstup`.



Obr. A.3: Prostředí záložky `kodér a dekodér`.

Záložka kodér a dekodér: Na této záložce, která je zobrazena na obr. A.3, je možno provádět vlastní krokování kodéru, resp. dekodéru. Záložka se skládá ze tří částí. První je kodér, druhou dekodér a poslední částí jsou ovládací prvky.

Ovládací část je velice jednoduchá, viz obr. A.4. Skládá se ze dvou tlačítek. Pomocí horního tlačítka lze provést jeden krok kódování. Jeden krok odpovídá vpuštění jednoho snímku do kodéru. Při prvním kroku dojde k deaktivaci všech tlačítek pro práci se soubory a také je znemožněno měnit jakékoli nastavení kódování, viz dále. Při tomto kroku jsou také kontrolovány rozměry vstupních snímků, které musí být u všech snímků stejné a musí být dělitelné 8 beze zbytku. Tlačítko `restart` restartuje krokování a vrátí program do stavu před provedením prvního kroku.

Vpuštění snímku do kodéru zpřístupní prohlížení detailů snímků před a po jednotlivých blocích kodéru, resp. dekodéru.

Kodér a dekodér jsou vertikálně odděleny. Kodér se nachází v horní polovině, dekodér ve spodní. Bloky kodéru a dekodéru jsou reprezentovány tlačítky. V podbarvených blocích je možno měnit nastavení kódování. Barva bloku vždy koresponduje s některou ze záložek, na které se dané nastavení provádí.

Obr. A.4: Ovládací prvky záložky `kodér a dekodér`.

Obr. A.5: Tlačítko pro zobrazení detailu snímku.

Záložka převzorkování: Na záložce `převzorkování` je možno před provedením prvního kroku krokování nastavit jaký typ vzorkování bude při kódování použit. Jsou možné varianty 4:4:4, 4:2:2 a 4:2:0. Tlačítkem `OK` dojde k přepnutí na záložku `kodér a dekodér`.

Záložka kvantování: Zde je možné definovat kvantovací matice, které budou použity při procesu kvantování. Pro kódování jsou použity dvě kvantovací matice. Jedna pro jasovou složku a druhá pro složky chrominanční.

Dalším nastavovacím prvkem je prahování. Pokud je nastavené na 0, prahování není uplatněno.

Jak kvantování s prahováním pracuje je možné vypozorovat na demonstračním příkladu, který je ve spodní části záložky. Vlevo dole je vstupní matice, která reprezentuje blok s jasovou složkou. Tuto matici není možné modifikovat. Po stisknutí tlačítka `výpočet` je tato matice kvantována, tj. dělena prvek po prvku maticí pro kvantování jasové složky, která je umístěna vlevo nahoře. Do procesu vstupuje ještě prahování, které vynuluje položky, které jsou po předchozím dělení menší než nastavený práh. Příklad společně s celým prostředím je znázorněn na obr. A.6.

Tlačítkem `OK` dojde opět k přepnutí na záložku `kodér a dekodér`.

Záložka DCT: Na záložce `DCT` jsou pouze zobrazeny rovnice pro výpočet dopředné i zpětné dvourozměrné kosinové transformace. Tlačítko `OK` opět přepne na záložku `kodér a dekodér`.

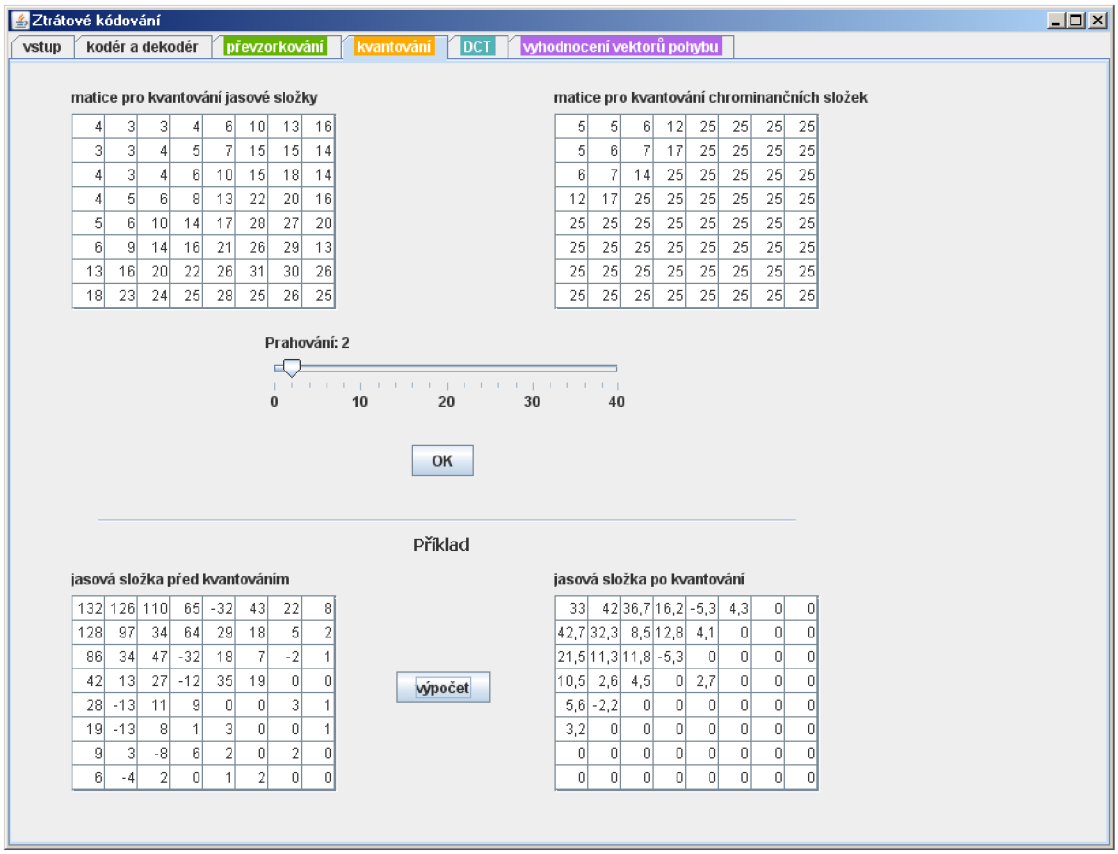
Záložka vyhodnocení vektorů pohybu: Nastavení, které lze provádět na této záložce, ovlivňuje způsob vyhledávání vektorů pohybu. V programu jsou realizovány

metody Full-Search a 3-Step-Search, které jsou popsány v kapitole 3.2. Přepnutí na záložku `kodér a dekodér` je možné provést pomocí tlačítka `OK`.

Možnosti detailu vstupního snímku: Při zobrazení vstupního snímku je možné pomocí volby `Zobrazit` zobrazit různé složky obrazu. Zobrazit lze:

- složku Y (Ctrl+Y),
- složku Cb posunutou do středního jasu (Ctrl+B),
- složku Cr posunutou do středního jasu (Ctrl+R),
- složku Cb posunutou do střední hodnoty (Ctrl+V),
- složku Cr posunutou do střední hodnoty (Ctrl+E),
- snímek v RGB (Ctrl+G).

V závorce jsou uvedeny klávesové zkratky, díky kterým je možno mezi volbami přepínat.



Obr. A.6: Prostředí záložky `kvantování`.

B OBSAH PŘILOŽENÉHO CD

- Text diplomové práce.
- Program – spustitelný `.jar` soubor.
- Zdrojové kódy programu.
- Projekt programu NetBeans.