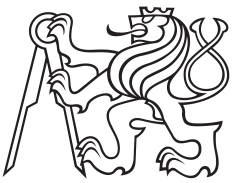
Bakalářská práce



České
vysoké
učení technické
v Praze

F3

Fakulta elektrotechnická
Katedra radioelektroniky

Technologie kontinuálního přenosu audiovizuálního obsahu

Rudolf Studený

Studijní program: Komunikace, multimédia a elektronika.
Obor: Multimediální technika.

Květen 2018

Vedoucí práce: Ing. Jan Bednář

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

I. OSOBNÍ A STUDIJNÍ ÚDAJE

Příjmení: **Studený** Jméno: **Rudolf** Osobní číslo: **434707**

Fakulta/ústav: **Fakulta elektrotechnická**

Zadávající katedra/ústav: **Katedra radioelektroniky**

Studijní program: **Komunikace, multimédia a elektronika**

Studijní obor: **Multimediální technika**

II. ÚDAJE K BAKALÁŘSKÉ PRÁCI

Název bakalářské práce:

Technologie kontinuálního přenosu audiovizuálního obsahu

Název bakalářské práce anglicky:

Technology of Continuous Transmission of Audiovisual Content

Pokyny pro vypracování:

Seznamte se s technologiemi kontinuálního přenosu audiovizuálního obsahu mezi zdrojem informace a koncovým uživatelem s ohledem na přenos a zpracování materiálu s ultra vysokým rozlišením UHD, s vysokým dynamickým rozsahem HDR a 360° videa.
Zaměřte se zejména na praktickou stránku věci: co je potřeba k uskutečnění přenosu, protokoly, webcast vs. video on demand, používané kodeky, typy serverů, poskytovatelé služeb a parametry těchto služeb.
S využitím těchto znalostí realizujte přenos audiovizuálního materiálu ve formátu UHD.

Seznam doporučené literatury:

[1] High-quality visual experience: creation, processing and interactivity of high-resolution and high-dimensional video signals. Editor Marta. MRAK, editor Mislav GRGIĆ, editor M. KUNT. Heidelberg [Germany]: Springer, c2010. Signals and communication technology. ISBN 978-3-642-12801-1.
[2] 2. Simpson, W.: Video Over IP: IPTV, Internet Video, H.264, P2P, Web TV, and Streaming: A Complete Guide to Understanding the Technology, Focal Press, ISBN-10: 0240810848, ISBN-13: 978-0240810843, 2008

Jméno a pracoviště vedoucí(ho) bakalářské práce:

Ing. Jan Bednář, katedra radioelektroniky FEL

Jméno a pracoviště druhé(ho) vedoucí(ho) nebo konzultanta(ky) bakalářské práce:

Datum zadání bakalářské práce: **17.02.2017** Termín odevzdání bakalářské práce: **25.05.2018**

Platnost zadání bakalářské práce: **31.08.2018**

Ing. Jan Bednář podpis vedoucí(ho) práce	podpis vedoucí(ho) ústavu/katedry	prof. Ing. Pavel Ripka, CSc. podpis děkana(ky)

III. PŘEVZETÍ ZADÁNÍ

Student bere na vědomí, že je povinen vypracovat bakalářskou práci samostatně, bez cizí pomoci, s výjimkou poskytnutých konzultací. Seznam použité literatury, jiných pramenů a jmen konzultantů je třeba uvést v bakalářské práci.

.	
Datum převzetí zadání	Podpis studenta

Poděkování / Prohlášení

Velký dík patří Ing. Janu Bednářovi, který to se mnou zkusil a v úzkých poradil, stejně tak za jeho ochotu a trpělivost při vedení mé bakalářské práce. Tato práce vznikla i jeho zásluhou.

Prohlašuji, že jsem předloženou práci vypracoval samostatně a že jsem uvedl veškeré použité informační zdroje v souladu s Metodickým pokynem o dodržování etických principů při přípravě vysokoškolských závěrečných prací.

V Praze dne

.....

Abstrakt / Abstract

Tato bakalářská práce se zabývá kontinuálním přenosem audiovizuálního obsahu, jak takovýto přenos uskutečnit, jakých prostředků je k tomu potřeba.

Seznamuje s moderními technologiemi v obrazové technice, jakými jsou např. UHD, HDR a sférická (360-stupňová) videa, snaží se je detailně popsat a dále se též zabývá jejich zpracováním. Popisuje a porovnává poslední kompresní metody, kterými jsou H.265/HEVC, VP9 či AV1.

Z praktického hlediska se pak zabývá přípravou obsahu a realizací přenosu (streamu) pomocí nástroje FFmpeg, spolu s tvorbou vlastního sférického videa.

This bachelor thesis deals with topic of continuous transmission of audiovisual content, how to realize such data stream and what resources are needed.

Further it presents modern technologies in image processing, such as UHD, HDR a spherical (360-degree) videos, aims on detail description of these terms and how to process such content. Thesis also describes latest compression (encoding) methods, such as H.265/HEVC, VP9 and AV1.

Practical part focuses on preparation of content for streaming and realization of transmission itself, using freeware tool FFmepg, as well as creation of custom spherical video content.

Obsah

Tabulky / Obrázky

Kapitola 1
Úvod

Podle posledních odhadů připadá na přenos multimediálního obsahu IP sítí kolem 70 procent celkového datového provozu na Internetu. Spolu s rostoucími požadavky na kvalitu konzumovaného obsahu a možností přehrávání i na mobilních zařízeních narážíme pomalu na zeď, kterou představuje datový tok, který jsme schopni sítí přenést za jednotku času.

Každý z nás se s technologií streamování (2.1) - kontinuálního přenosu obrazu anebo zvuku - setkává denně v podobě sledování zábavních videí, filmů, přednášek, poslouchání hudby, rádia. Jaké druhy streamů ale rozlišujeme (2.1.1), a také co je potřeba k vytvoření tohoto spojení mezi serverem a námi? (2.2).

Díky pokrokům zvláště v obrazové technice, které představují například ultra vysoká rozlišení video formátů (3.1), přichází také otázka, jak tato velká data zpracovávat (3.1.1).

V neposlední řadě se tato práce také zaměří na fenomén zvaný 360-stupňová videa, o co jde a jak s nimi zacházet (3.3) a v praxi si popíšeme, jak takový stream pomocí nástroje FFmpeg vytvořit (4).

Kapitola 2
Stream a streamovací řetězec

Streamování (angl. *streaming*) je pojem, popisující proces kontinuálního přenosu dat od zdroje, ke koncovému uživateli. Stream (z angl. *proud*) pak označuje tok dat jako takový. Jde o běžně rozšířený způsob, jakým dnes distribuujeme zvláště multimediální obsah - tedy jak zvukový (audio), tak obrazový (video).

2.1 Typy streamování

Protože je streamování pojem velmi obecný, zahrnuje různé technologie a způsoby přenosu multimediálního obsahu. Popišme si blíže ty nejběžnější:

– **True streaming** (česky doslova *skutečné streamování*) je princip, kdy jsou vysílaná data přijímána a sledována v reálném čase. Pokud tedy sledujeme dvouminutové video, tak nám vznikne dvouminutový stream, který přenese žádaný obsah ze serveru a tím skončí - tedy délka streamování se odvíjí od délky obsahu.

– **Download and play** (česky *stáhni a přehraj*) jak název napovídá, celý obsah určený ke sledování, je nejprve celý stažen do uživatelského zařízení, teprve poté přehrán. Omezení zde může představovat rychlost připojení uživatele, nicméně po stažení již není přenos nijak přerušován.

– **Progressive download and play** (česky *postupné stahování a přehrávání*) představuje kompromis předchozích dvou principů, snaží si vzít z obou to nejlepší. Nepřenese se obsah jako celek, ale kratší, samostatně přehratelné sekce - v jeden moment může sledovat uživatel žádaný obsah a v pozadí se již stahuje sekce následující.

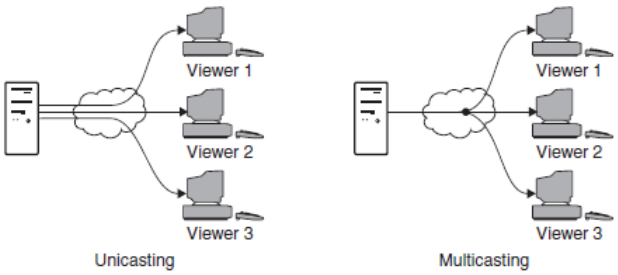
2.1.1 Webcast vs. VOD

– **(Live) webcast** – Jde o způsob přenosu, který se svým principem podobá klasickému televiznímu vysílání. Může jít o živé vysílání obsahu (např. streamovaná přednáška) nebo plánovaný přednahraný přenos, který může být vysílán ze serveru. [1] Je to jednosměrný tok dat, bez jakékoliv interakce na průběh streamování, a uživatel sleduje obsah od bodu, kdy se ke streamu připojí.

– **Video on Demand** (zkr. VOD, česky *video na vyžádání*) – Je způsob přenosu multimediálního obsahu, kdy je stream vytvořen až „na požádání“ uživatele. Jde o unicast stream (více o tomto v kapitole 2.1.2), který je vytvořen ve chvíli, kdy na daný server přijde žádost, jaký obsah má být vysílán. Velkým rozdílem od webcastingu je interaktivita - uživatel je schopen v reálnem čase ovlivnit průběh streamu běžnými funkcemi jako je: play (přehrát), pause (pozastavit), fast forward (přetáčení) a podobně. [1] To je možné díky faktu, že se takovéto unicast streamy tvoří pro každého uživatele zvlášť, kdežto při webcastu se uživatel připojí k vysílání, které běží i v případě, že ho nikdo nepřijímá. Jako typické představitele VOD streamů jsou dnes celosvětově rozšířené hostovací video služby jako YouTube, Metacafe, Vimeo a mnoho dalších.

2.1.2 Multicast

Když mluvíme o distribuci signálu v IP sítích, je důležité se zmínit o multicastu i přesto, že pro distribuci audiovisuálního obsahu je spíše výjimkou. Jde o proces, během kterého z jednoho zdroje signálu rozesíláme stream většímu množství uživatelů zároveň. Jednotlivé streamy ale nevznikají už u zdroje streamu, ale jednotlivé kopie se tvoří až v distribuční síti, resp. na routerech. Pro lepší porozumění tento pojem popíšeme v kontrastu s tzv. unicastem.



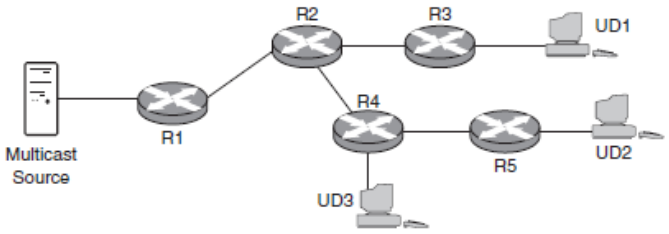
Obrázek 2.1. Rozdíl mezi unicast a multicast vysíláním. Převzato z [2]

– **Unicast** – (česky také nazýván *jednosměrové vysílání*) popisuje distribuci signálu právě mezi dvěma body sítě. Každý stream je tedy určen pro jednoho uživatele. V případě, že jiný uživatel chce sledovat stejný obsah, je pro něj vytvořen vlastní unicast stream. Značnou nevýhodou je zátěž na zdroj streamu - je nutný výkon na zpracování distribuovaného obsahu a síťové připojení o dostatečné šířce pro vytvoření streamu pro každého uživatele. [2]

Unicast vysílání je obvyklé v LAN a Internetu. Veškeré LAN a IP sítě podporují unicast přenos a mezi další standartní aplikace těchto připojení řadíme např. http, smtp, ftp, telnet; tyto služby používají TCP, pro naše aplikace též UDP protokoly (více o protokolech v kapitole 2.2.3) [3]

– **Multicast** – (česky *vícesměrové vysílání*) popisuje podobnou distribuci streamu jako unicast, s tím rozdílem, že zátěž s vytvářením jednotlivých streamů se přenáší na síť samotnou (pro unicast je tato zátěž na zdroji obsahu). Routery schopné vysílat v multicastu umí rozpoznat pakety, které mají být takto distribuovány, a ty pakety rozešle do několika lokací najednou, čehož docílí speciálními multicast adresami. Tato zátěž na síť nemusí být vždy výhodou, zvláště pokud je na jednom zařízení (routeru) větší počet účastníků, kteří sledují stejný stream - router má totiž za úkol vzít příchozí pakety a vytvořit kopii pro každého připojeného účastníka. Navíc ale musí i sledovat, zda se na další porty nechce k tomuto streamu připojit nový účastník, nebo naopak zda se účastník neodpojil a není tedy nutné vysílat na daný port. Multicast funguje jednosměrně a vyjma údajů jako například FLR, nemá zdroj obsahu feedback ze sítě. Interaktivita jakou máme v případě unicastu není možná.

Řekněme si ještě něco o **připojování/odpojování účastníků**. Jak je zmíněno výše, každý multicast router zajišťuje kopii streamu na porty, které o to žádají, sleduje zda některý port nově nežádá o nové připojení a že z některého již aktivního portu se uživatel neodpojil. Ukažme si to názorně na příkladu.



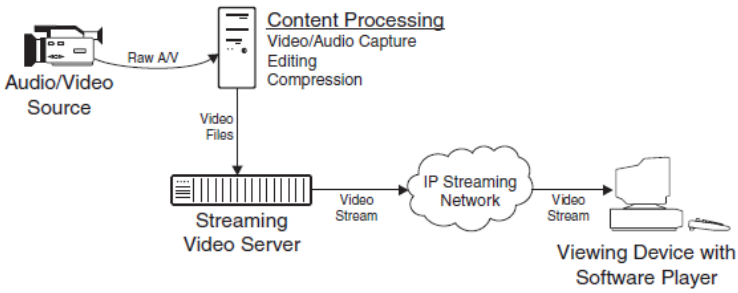
Obrázek 2.2. Připojení/odpojení uživatelů při multicast vysílání. Převzato z [2]

Vezměme topologii sítě tak, jak jí vidíme výše, na obrázku 2.2. Předpokládáme, že zatím není připojený žádný účastník (UD = user device, uživatelské zařízení). V případě, že se chce připojit uživatel UD1, vyšle request (žádost) o vytvoření streamu na router R3, ten ji předá R2, ten R1 a ten zdroji multicastu (Mulitcast source). Každý router nejprve ověří, zda už k tomuto streamu není připojený, a nestačí jen vytvořit kopii pro tohoto účastníka. Nyní máme uživatele UD1 připojeného a k našemu streamu se chce připojit uživatel UD3. Žádá o připojení nejbližší zařízení, router R4, ten připojen není a žádá u R2. Router R2 již stream přijímá a stačí mu tedy vytvořit novou kopii i na port pro R4 a ten připojí UD3.

Odpojování od streamu probíhá obdobně. V principu router zjišťuje, zda na jeho portech jsou stále účastníci, pro které je potřeba vysílat. Pokud by se tedy např. uživatel UD1 odpojil, router R3 zjistí, že na jeho portech už nikdo stream nepřijímá, přestane vysílat a předá informaci routeru R2. Ten přestane vysílat pro R3, ale na jiném portu vysílá pro účastníka UD3 a tedy nepřestává přijímat stream od R1.

2.2 Streamovací řetězec

Popišme si, co je k realizaci takového streamu potřeba - od zdroje obsahu, přes jeho zpracování a distribuci až po koncového uživatele, diváka.



Obrázek 2.3. Architektura typického streamovacího systému. [2]

Na obrázku 2.3 můžeme vidět základní prvky streamovacího řetězce. Na počátku je zdroj našeho signálu - v tomto případě nějaké záznamové zařízení, např. videokamera. Tento signál bude samozřejmě představovat velký datový tok, sám o sobě nevhodný pro streamování do sítě. Je třeba ho tedy zpracovat a připravit k distribuci, což představuje blok **Content processing** (česky *zpracování obsahu*). Může jít například o PC či konkrétně dedikovaný hardware, sloužící k ukládání zaznamenaných dat, editaci a zvláště pak kompresi dat. Takto předzpracovaná data již mohou být uploadována (nahrána) na **server**, speciálně určený právě ke streamování. Odtud vytvářené streamy již dopraví data po IP síti k jednotlivým uživatelům, pro které jsou streamy vytvářené.

Rozeberme si blíže tyto segmenty v dalších kapitolách.

2.2.1 Příprava obsahu, komprese

Zdrojem multimediálního obsahu může dnes být cokoliv - od profesionálních kamer, po webkameru v notebooku s integrovaným mikrofone. Takovýto (RAW) signál je vhodný ke zpracování, nikoliv ale k distribuci po síti. Prvním krokem by tedy měla být příprava ke streamingu, speciálně konverze formátu a komprese.

Časová náročnost tohoto procesu se může lišit na základě požadavků na výslednou kvalitu. Od kvality se dále odvíjí doba zpracování signálu, doba komprese i požadavky na síť, protože právě multimediální signály představují velké datové toky a tedy velké vytížení kapacity sítě.

Zdroj našeho signálu určuje, jak budeme signál zpracovávat před jeho distribucí. Je-li zdrojem videokamera nahrávající v analogovém formátu, jistě bude pro další zpracování nutná konverze na digitální signál. Signál může pocházet i ze záznamových medií, která (obzvláště ta stará, jako třeba magnetofonové pásky, videokazety apod.) můžou zanést do konvertovaného digitálního signálu artefakty či šumy, které je potřeba dále ošetřit, třeba pomocí filtrací. Podobně musíme myslet na zpracování audiosignálu, jeho ekvalizaci, případně synchronizaci s video signálem. Dnešní doba nám již umožňuje nahrávat samozřejmě i v digitální podobě, čímž se nám předzpracování signálu značně zjednodušuje, a je běžné, že se tak děje přímo během streamování.

V této kapitole se nyní zaměříme hlavně na **kompresní metody** pro audio i video signály. Z praktického hlediska totiž není možné přenášet audiovizuální obsah v RAW formátu - představoval by obrovské množství dat, což by přineslo zátěž jednak na straně uživatele a jeho síťové připojení, tak na straně serveru, který by vytvářel unicast streamy pro jednotlivé uživatele.

Jak tedy obecně kodeky [1] fungují? Mluvíme-li o **bezeztrátové kompresi** (angl. *lossless*), snaží se kodek zbavit redundantních (nadbytečných) informací ze zpracovávaného signálu. Výhodou těchto kompresí je možnost rekonstrukce do původní podoby bez nejmenší odchylky. Těchto metod se ale užívá speciálně při kompresi počítačových dat, textových souborů apod., nikoliv tolik pro multimediální soubory. Neposkytují totiž tak velký kompresní poměr [2] (typicky třeba 1/3).

Mnohem běžnější a výhodnější je proto použít kodeky s tzv. **ztrátovou kompresí** (angl. *lossy*). Ty oproti bezeztrátovým navíc využívají nedokonalosti lidských smyslů - sluchu a zraku, a kromě redundantní odstraňují také irelevantní informaci, tj. takovou, kterou na reprezentaci obrazu anebo zvuku v našem mozku nemá tak značný vliv. Mozek tyto chybějící informace buď nevnímá nebo je schopen si je domyslet. Díky tomu tyto komprese dosahují vyšších kompresních poměrů - 1/20 i vyšší, za cenu ztráty kvality obsahu.

— Komprese obrazu Abychom snáze porozuměli kompresím videa, začneme nejprve kompresí statického obrazu. Nakonec video je reprezentováno sekvencí obrazových statických snímků, promítaných dostatečně rychle (totiž takovou frekvencí, aby lidské oko nezaznamenalo tyto přechody, u televizního signálu typicky 50 Hz).

JPEG [3] je nejznámější kompresní metodou pro statický obraz, na které lze dobře demonstrovat základní principy zpracování obrazu obecně. Pochází z přelomu 80. a 90.

[1] Slovo kodek je složeninou počátečních slabik slov „komprese“ a „dekomprese“, může jít o SW či dedikované HW kompresní zařízení.

[2] Velikostní poměr mezi původním a komprimovaným souborem

[3] Zkratka je odvozena z názvu uskupení, které s tímto kodekem přišla: Joint Photographic Experts Group.

let 20. století a zohledňuje fyziologii lidského vizuálního systému a kolorimetrie. JPEG lze zhruba popsat v pěti krocích.

– *Transformace barevného prostoru* - obrazové snímače totiž z technologického a konstrukčního principu snímají obraz v RGB. Pro zpracování je ovšem výhodné tyto hodnoty přepočítat na prostor YC_BC_R, (jde o Y – jasovou složku; C_B,C_R – barevné komponenty, které obsahují větší množství irelevantní informace).

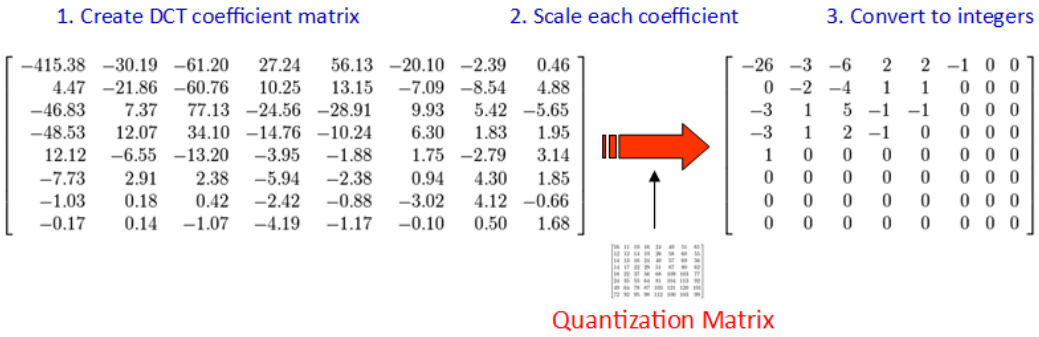
– *Barevné podvzorkování* - proces, kdy jsou všechny hodnoty barevných intenzit na vstupu postupně načítány, ale na výstupu jsou zapsány jen některé. Tím dochází ke snížení rozlišení barevných rovin a k nenávratné ztrátě informace.

– *Transformace obrazu po částech do kmitočtové oblasti* - po rozdělení obrazu na bloky (8x8 pixelů) dochází k diskrétní kosinové transformaci, jejíž výstupem je matice 8x8 obsahující koeficienty, které udávají míru zastoupení jednotlivých frekvencí v původním obrázku.

$$S(k/2) = \frac{1}{2}K(k)\sum_{u=0}^{N-1} s(u)\cos\frac{(2u+1)k\pi}{2N} \tag{1}$$

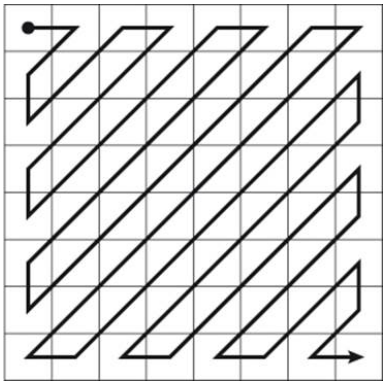
$$k = 0 : K(k) = \sqrt{\frac{1}{N}}, k \neq 0 : K(k) = \sqrt{\frac{2}{N}}$$

– *Kvantování* - udává z největší části kompresní poměr. Matice DCT koeficientů z předchozího bodu se vydělí koeficientem – tím se nám některé kmitočtové koeficienty vynulují, a dochází k další ztrátě informace.



Obrázek 2.4. Ukázka hodnot po přenásobení kvantizační maticí. Převzato z [4]

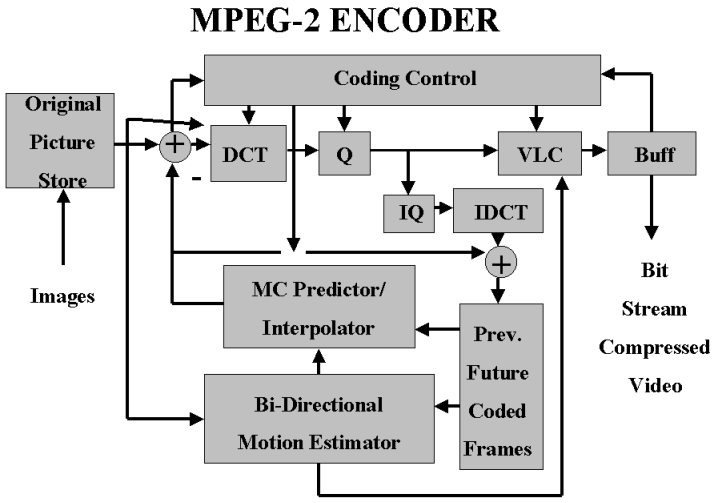
– *Kódování* - zde využíváme řady nulových hodnot za sebou, které nám vznikli během kvantizace. S využitím metod RLE – ZIG-ZAG, kde pomocí čtení hodnot ZIG-ZAG vznikne řada těchto hodnot, které můžeme komprimovat pomocí RLE kódování, které je výhodné kvůli nulovým hodnotám na vyšších obrazových kmitočtech. RLE (Run-lenght encoding) funguje na principu, řetězce symbolů zkracuje vždy na formát „symbol + počet symbolů“. Tím se efektivně zpracovávají právě řetězce s často se opakujícími symboly.



Obrázek 2.5. Vyčítání hodnot metodou ZIG-ZAG. Převzato z [5]

Nyní přejděme k obrazovým sekvencím (video, animace). Jak je již zmíněno výše, když mluvíme o videu, jde vlastně o jednotlivé statické snímky, které při rychlém přehrávání vnímáme jako spojité.

MPEG[1] je název pracovní skupiny vyvíjející standardy používané na kódování těchto signálů. Jde také o rodinu formátů, pokrývajících celou řadu aplikací. Popišme si blíže **MPEG-2**, jeden z nejrozšířenějších standardů zabývající se kódováním pohyblivého obrazu a přidruženého zvuku. Obecně jsou kompresní schémata pro video založena na identifikaci a odstraňování dvou typů redundance – prostorové (nadbytečná informace rozložená v prostoru popsaném souřadnicemi každého obrázku) a časové (díky podobnosti sousedních snímků).



Obrázek 2.6. Blokové schéma MPEG-2 kodéru. Převzato z [6]

První, jednodušší, způsob ukládání sekvence snímků spočívá v procesu, kde po sobě následující snímky I_1, I_2 nejsou zakódovány každý zvlášť, ale nejprve je vypočten rozdíl stejnolehlých pixelů v celém snímku: $P = I_1 - I_2$. Rozdíl P je pak zakódován na místo původního snímku, čímž dojde k dramatickému poklesu objemu dat, protože díky předpokládané podobnosti dvou sousedních snímků (snímky jsou pořízeny velmi krátce po sobě, např. při rychlosti 25 snímků za sekundu jde o 1/25 s) tento rozdíl bude obsahovat mnohem méně informace než původní snímek. Specifikace MPEG označuje

[1] Zkráceno z Moving Picture Experts Group

snímky s kompletní informací jako **I-snímky** (*Intra-frame*) a rozdílové jako **P-snímky** (Predictive-frame). Ty nesou informaci označovanou jako prediktivní chyba. Snímky I a P jsou pak kódovány v rámci procesu podobném kompresi JPEG založeném na DCT a tvoří sekvenci, kde je mezi dvěma I snímky uloženo několik P snímků:

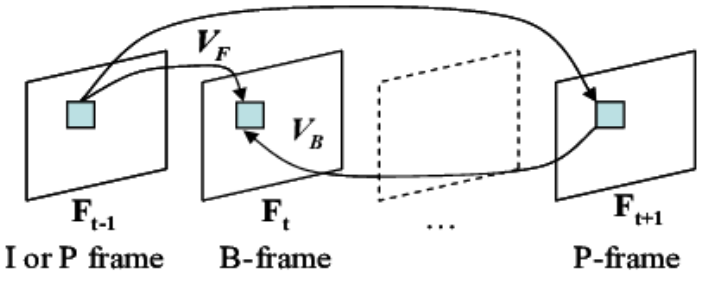
$$I\ P\ P\ P\ I\ P\ P\ P\ I\ P\ P\ P \tag{2}$$

Tato sekvence (2) odpovídá způsobu, kterým ukládá snímky MPEG-2 podle profilu *Simple profile*. Snímky P jsou zde dány rozdílem od předchozího I snímku. Perioda obsahující I snímek a všechny snímky předcházející dalšímu I snímku (v tomto případě $I\ P\ P\ P$) tvoří tzv. **skupinu snímků** (angl. *Group of pictures* neboli GOP). Video, u kterého se předpokládá další zpracování, se obvykle kóduje pouze s pomocí I snímků.

— *Kompenzace pohybu* - Na běžícím videu se typicky pohybují různé objekty, které se v různých časech vyskytují na snímku v různých místech, a to zvyšuje prediktivní chybu. Standard MPEG-2 proto za účelem dalšího snížení objemu dat používá další metodu nazývanou kompenzace pohybu (angl. *motion compensation*). Aktuální snímek I_2 je nejprve rozdělen na makrobloky o velikosti 16x16 pixelů. Každý makroblok je pak vyhledán v předchozím snímku I_1 pomocí prediktoru pohybu a je zaznamenána změna jeho pozice v podobě vektoru.

Pro celý snímek je takto získána celá množina vektorů V. Posuny makrobloků reprezentované množinou V jsou pak aplikovány na snímek I_1, čímž je získán prediktivní snímek P'. Během pohybu však objekty mohou měnit tvar, a proto je ještě vyčíslena prediktivní chyba rozdílem $P = P' - I_2$. Obsah P je pak zakódován místo původního snímku I_2 a spolu s ním je uložena i množina vektorů V.

V situacích, kdy na aktuálním snímku pohybující se objekt odkryje část pozadí, která byla na předchozím snímku zakryta, nastává problém, jak určit obsah odkryté neznámé oblasti. V tomto případě prediktivní kódování selhává, neboť pro zakódování chybí informace, která na předešlém snímku není k dispozici. Je ale možné ji získat na snímku následujícím (viz obr. 2.7).



Obrázek 2.7. Vztah snímků definovaných standardem MPEG. Převzato z [7]

MPEG specifikace počítá pro tyto situace s obousměrnými **B-snímky** (angl. *Bidirectional frame*). Tento typ snímků je zakódován pomocí rozdílů nejen vzhledem k předchozímu, ale i následujícímu snímku. Použití B-snímků radikálně snižuje nutnou přenosovou rychlost při zachování kvality videa. Vztah (3) zobrazuje typickou skupinu snímků obsahující I a P snímky, prokládané B-snímky.

$$I\ B\ B\ P\ B\ B\ P\ B\ B\ P\ B\ B\ I\ B\ B\ P \tag{3}$$

Při popisu kodéru se obvykle udává velikost skupiny GOP. Je to vzdálenost dvou následujících I snímků v sekvenci vyjádřená v počtu snímků. Toto číslo bývá doplněno dalším údajem vyjadřujícím vzdálenost následujících I a P snímků. Tyto hodnoty se

objevují jako dvojice $[M, N]$, kde N je velikost GOP. Typicky tato dvojice nabývá hodnot [3,12], což je i případ v sekvenci (3). Vyšší hodnoty M a N znamenají značné snížení objemu ukládaných dat, ale vedou k postupné ztrátě přesnosti a degradaci obsahu videa, a vice versa. [8]

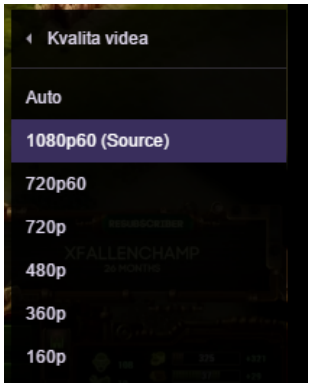
Takový je stručný náhled na zpracování videa. Pro naše účely je kompresní metoda popsaná standardem MPEG-2 poměrně zastaralá, a na obsah v UHD již nestačí. V kapitole 3.1.1 si ale kompresní metody, které to umí, ještě popíšeme.

2.2.2 Servery

Servery jsou zodpovědné za distribuci multimediálního obsahu ke koncovému uživateli. Můžou sloužit jako úložiště tohoto obsahu, který je předem nahrán (resp. při živém vysílání kontinuálně nahráván). Při obdržení žádosti ze strany uživatele, vytváří jednotlivé streamy, kterými dopraví žádaný obsah.

V praxi pak můžeme rozlišit několik typů jejich serverů, podle jejich konkrétní specifické funkce:

– *Streaming servery* - Hlavní funkcí těchto serverů je uskladnění a distribuce multimediálního obsahu. V praxi je běžné, že servery neobsahují pouze jednu verzi požadovaného obsahu, ale rovnou několik. Důvodem je náročnost na síťové připojení uživatele. Proto je ten samý obsah komprimován několikrát s různými kompresními poměry, které sice budou mít v extrémních případech značný vliv na kvalitu, ale i uživatel připojený nízkokapacitní přípojkou může konzumovat stejný obsah, jako uživatel připojený přes optickou síť.



Obrázek 2.8. Možnosti kvality streamu (služba Twitch).

Server někdy může dostat také zodpovědnost za upravování a průběžnou změnu kvality streamu v závislosti na měnících se síťových podmínkách - např. degradace bezdrátového spojení díky meteorologickým vlivům, vytížení lokální sítě apod. Server automaticky pak vybere nižší kvalitu uploadovaného streamu jako kompenzaci těchto vlivů. Tato funkce je na obrázku 2.8 reprezentována funkcí *auto*.

Větší množství verzí jednoho obsahu se ještě dále komplikuje ve chvíli, kdy server podporuje několik přehrávačů (tj. software, který na straně uživatele reprezentuje tok paketů jako obraz anebo zvuk) jako například QuickTime, Windows Media Player aj. Je tedy běžné, že je obsah kódován kompatibilně hned s několika přehrávači pro zahrnutí co nejvíce uživatelů. Tím se nám počet verzí jednoho obsahu znovu násobí.

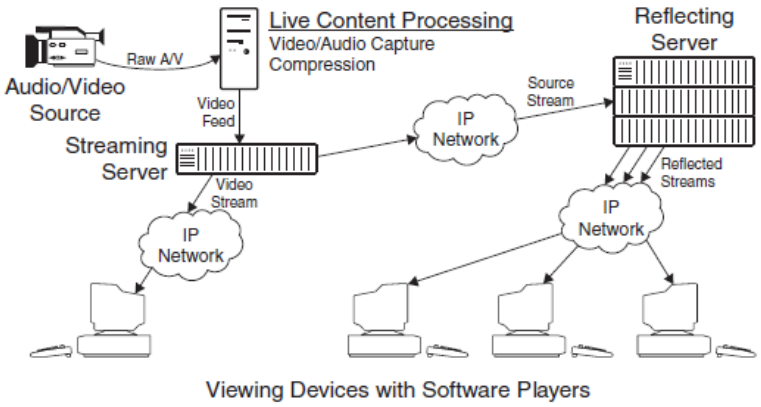
Dále tedy server produkuje pakety pro každý odchozí stream v reálném čase. To zahrnuje tvorbu hlaviček (angl. *header*) IP paketům, obsahující správnou IP adresu destinace, na kterou mají dorazit. V případě že server posílá standardní RTP pakety

(více o protokolech v kapitole 2.2.3), video a audio stream musí být posílán separátně, každý na jiný port uživatelského zařízení.

Šifrování paketu do odchozího streamu může být v režii serveru, pokud je to požadováno - např. užitím moderního šifrování tzv. **Public-key cryptography** (Kryptografie s veřejným klíčem) nebo-li asymetrickou kryptografii, kdy šifrování probíhá s veřejným klíčem a dešifrování s jiným, privátním klíčem (který je ale s tím veřejným matematicky svázán). Každý stream tedy musí být šifrován unikátním klíčem pro každého uživatele.

Z důvodů výše zmíněných, se při větším množství uživatelů konzumujících stream, může server velmi vytížit. Proto se jako kompenzace obsah může zkopírovat na několik serverů, nacházejících se na fyzicky rozdílných lokacích, a tím mohou podpořit tvorbu streamů bez dopad na uživatele.

– *Reflecting servery* - právě z důvodů posledně zmíněných, existují tzv. reflecting servery. Jejich funkcí je podpořit vzdálený server posílající data do určité lokality, a to zvlášť v případech, kdy dochází k vysoké špičce nad běžný provoz. Reflecting servery jsou nejčastěji instalovány soukromými společnostmi pro podpoření streamů do lokální sítě. Vzdálenému serveru tedy stačí poslat pouze jeden datový stream do lokality, a reflecting server se postará o distribuce jednotlivým uživatelům. [2]



Obrázek 2.9. Ukázka distribuce pomocí reflecting serveru. Převzato z [2]

2.2.3 Přenos IP sítí

Serverem vytvořené streamy jsou dopravovány přes IP síť podobně, jako kterákoli jiná data. Tuto komunikaci a přenos dat zajišťují protokoly. Problémem je, že kombinace protokolů TCP/IP, kterou využívá většina webových aplikací, je pro streamování multimediálního obsahu naprosto nevhodná. Rozeberme si tyto protokoly blíže, abychom zjistili proč.

TCP/IP (neboli *Transmision Control Protocol/Internet Protocol*) - Internet Protocol je hlavním komunikačním protokolem pracujícím na síťové vrstvě[1]. IP má ale samo o sobě nevýhodu. Není spolehlivým systémem k doručovaní - má proměnlivou latenci (zpoždění), pakety mohou dorazit v různém pořadí, než byli vyslány, a může dojít ke ztrátě paketů při přenosu.

Tyto nedostatky jsou kompenzovány protokoly vyšších vrstvy. Transport Protocol je představitelem čtvrté, transportní vrstvy. Jeho největší výhodou je spolehlivost. Jeho protichybová ochrana z něj dělá ideálního prostředníka k doručení běžných dat jako e-mail, textové soubory, zálohovaná data apod., ale její implementace je nevhodná právě

[1] Tedy třetí vrstva dle referenčního modelu ISO/OSI

ke streamování. TCP posílá informací jaký byte má cílová destinace očekávat. Pokud byte nedorazí v daném časové periodě, je byte poslán znovu (tzv. retransmise). Tato vlastnost protokolu TCP umožňuje zařízením detekovat ztracené pakety a zažádat si tedy o nové zaslání. To ovšem přidává k latenci celkové komunikace a přenosu dat jako celku. Při konzumaci multimediálního obsahu ale uživatel předpokládá příchod dat v reálném čase. Tato přerušení a zpoždění v přenosu paketů tvoří zpoždění, která zapříčiní vyprázdnění bufferu přehrávače a tedy pozastavení v kontinuálním zobrazení obsahu, než se buffer znovu naplní a přehrávání pokračuje (v případě živého vysílání by uživatel přišel o celé části vysílaného obsahu). Alternativou k této ochraně proti chybnému přenesení je ignorace těchto ztracených paketů. To může vést ke zkreslení nebo ztrátě některých snímků, což je ale pouze přechodným jevem, který nebude nutně uživatelem zaznamenán. V konečném důsledku je tedy pro real-time aplikace (česky *aplikace v reálném čase*) přednější včasné doručení než bezchybný přenos.

Aplikace	Protokol aplikační vrstvy	Typický transportní protokol
E-mail	SMTP	TCP
Vzdálený přístup k terminálu	Telnet	TCP
Web	HTTP	TCP
Přenos souborů	FTP	TCP
Vzdálený souborový server	NFS	UDP
Streamování	RTSP nebo proprietární	UDP
Voice-over IP	proprietární	UDP
Managment sítě	SNMP	UDP
Routovací protokol	RIP	UDP
Překlad doménových jmen	DNS	UDP

Tabulka 2.1. Běžné internetové aplikace a jejich transportní protokoly. Převzato z [1]

Výše zmíněné problémy kompenzujeme řadou jiných protokolů, které mají pro naše aplikace řadu výhod. Jedním z těchto protokolů je **UDP** (*User Datagram Protocol*).

UDP na rozdíl od TCP nijak nezaručuje správný přenos paketu (datagramu) - zda bude vůbec doručen, nebo nebude doručen několikrát. Z hlediska přenesených dat ho lze považovat za *nespolehlivý*. UDP pouze přidává kontrolní součty a schopnost roztřiďovat UDP pakety mezi více aplikací běžících na stejném počítači.[9]

Je využíván tedy jako transportní protokol v kombinaci s řadou protokolů aplikační vrstvy: např. Network File System (NFS), Simple Network Management Protocol (SNMP), a Domain Name System (DNS).

+	bity 0 - 15	16 - 31
0	zdrojový port	cílový port
32	délka	kontrolní součet
64	data	

Obrázek 2.10. Formát UDP paketu. Převzato z [9].

Rozeberme si tedy další protokoly, které využíváme spolu s UDP při přenosu multimediálního obsahu, a k čemu slouží.

Zkratka	Celý název protokolu	Pozn.	Číslo RFC dok.
RSVP	Resource Reservation Protocol	Specifikace protokolu	2205
	RSVP applicability statement	Nasazení protokolu	2208
	Message processing rules		2209
RTCP	Real-Time Control Protocol	Součást RTP	3550
RTSP	Real-Time Streaming Protocol		2326
RTP	Real-Time Protocol		3550
SDP	Session Description Protocol		2327
UDP	User Datagram Protocol		768

Tabulka 2.2. Souhrn protokolů pro přenos multimediálního obsahu. Převzato z [1]

RTP (*Real-time transport protocol* nebo také *Transport Protocol for Real-Time Applications*) je transportní protokol poprvé uvedený v roce 1996 popsaný RFC 1889, později v roce 2003 nahrazen RFC 3550. Jde o protokol určený k datovému přenosu v reálném čase, jako je interaktivní obrazový a zvukový obsah. Je obvykle používám v kombinaci s výše zmíněným UDP.

Sám o sobě neobsahuje mechanismy k zaručení včasného přenesení dat, či jiné quality-of-service garance, ale spoléhá na služby nižších vrstev. [10]

RTCP (*RTP Control Protocol*) je doplňující řídící protokol k RTP, specifikován v témže dokumentu RFC 3550. Slouží jako zpětná vazba na kvalitu služeb (QoS - Quality of Service) poskytovanou RTP, ve formě paketů od účastníků streamovací relace. Sám ovšem data nepřenáší. Přenáší informace jako např. počet odeslaných bajtů, počet odeslaných paketů, počet ztracených paketů, jitter, zpětnou vazbu a dobu odezvy. [10]

RSVP (*Resource ReSerVation Protocol*) je signalizačním protokolem transportní vrstvy specifikovaným v dokumentu RFC 2205, původně navržen pro multicast (podporuje ale i unicast). Je určen k reservaci síťových prostředků - používán hostitelem služby, který tímto způsobem žádá o daný QoS v síti. Dále je také využíván routery, které takto zajišťují určitý datový tok v každém uzlu na cestě k uživateli.

Jde o simplex, tzn. zajišťuje jednosměrný provoz; a také reciever-orinented, tedy příjemce zahajuje a udržuje rezervaci síťových prostředků. Je dynamický z hlediska změn v routování. Podporuje IPv4 i IPv6. [11]

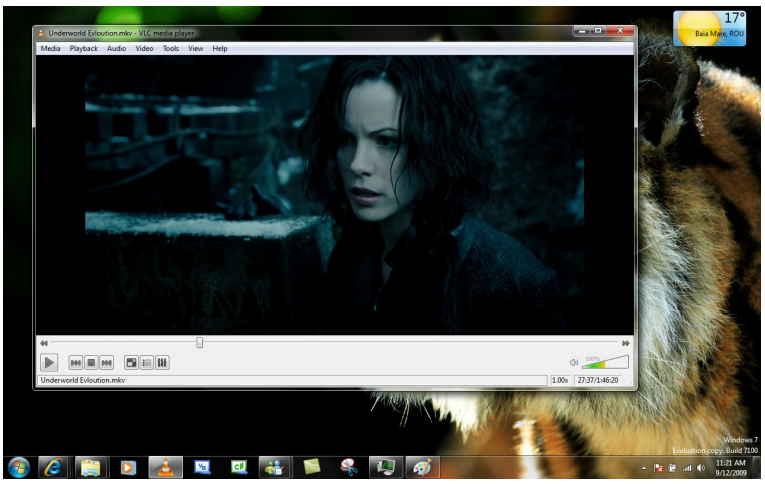
RTSP (*Real Time Streaming Protocol*) je protokol aplikační vrstvy specifikovaný v RFC 2326. Je navržen k řízení mediálních streamů v reálném čase, pomocí implementace příkazů jako *play, pause, fast forward* v případě streamu od serveru k uživateli (např. On Demand obsah) nebo také *record* při streamování od uživatele na server (nahrávání vlastního obsahu). [12]

SDP (*Session Description Protocol*) je internetový protokol specifikovaný v RFC 2327, určený k popisu vlastností relace multimediálního streamu. Vyjednává parametry jako typ média (obraz, zvuk,...), transportní protokol, typ kodeku, přenosová rychlost. [13]

Tímto je shrnuta většina protokolů pro účely streamování a dostáváme se přes přenos IP sítí k uživateli, a reprezentaci dat na jeho straně.

2.2.4 Mediální přehrávač

Mediálním přehrávačem (angl. *media player* nebo *player software*) rozumíme software běžící na PC uživatele, který přijímá stream. Je odpovědný za příjem streamu a reprezentaci příchozích paketů ve formě obrazu anebo zvuku. Pokud je přijímaný mediální obsah šifrován, je úkolem přehrávače ho dešifrovat pomocí klíče, jež obdrží přímo ze streamovacího serveru nebo pomocí ověřovací služby třetí strany.



Obrázek 2.11. Mediální přehrávač VLC. Převzato z *https://www.videolan.org/*.

Streamovací protokoly, jako RTP, oddělují obraz a zvuk do samostatných datových toků. Přehrávač je pak také zodpovědný za zpětnou synchronizaci příchozích streamů, a to pomocí tzv. *time-stamps* (časových značek) obsažených v obou streamech, které porovnává s přiřazenými RTCP pakety.

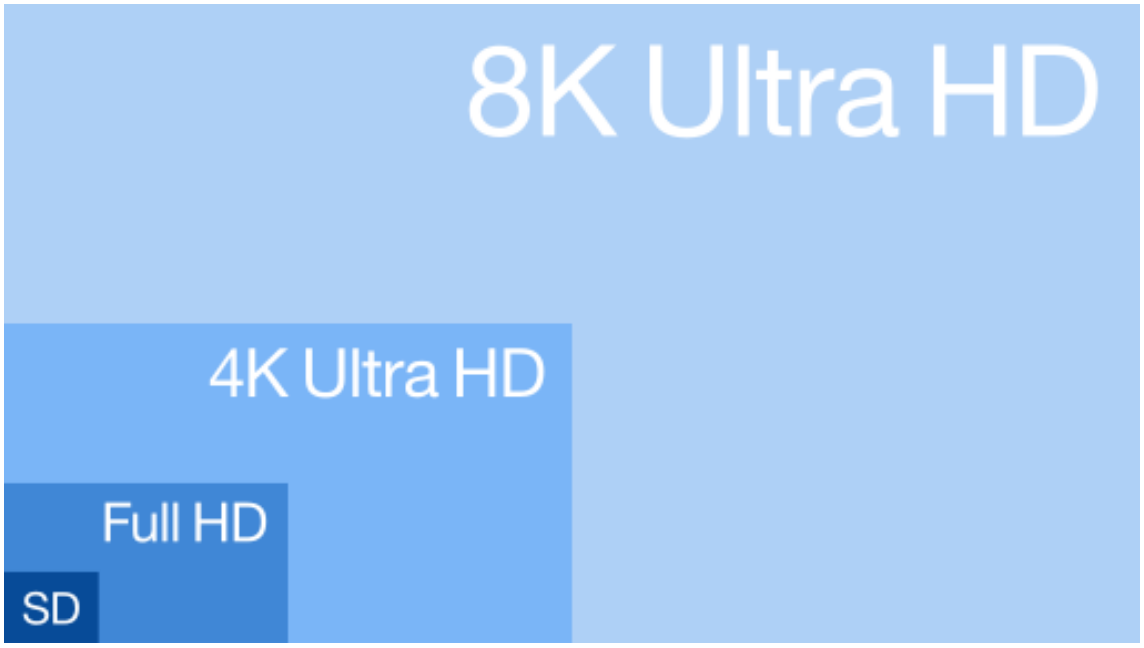
Díky enormním datovým tokům je streamovaný obsah nutně komprimován (pomocí kompresních metod, např. MPEG-4 aj.). Mediální přehrávač tedy přijatý obsah také zpětně dekóduje, což představuje nutný výpočetní výkon (větší datové toky, složitější kompresní metody, představují větší zátěž). Toto se dá obejít dedikovaným zařízením, hardwarem, který se stará pouze o dekódování přijímaných streamů - jde např. o televizní set-top boxy, hardwarové dekodéry apod. [2]

Kapitola 3
Současné technologie

Předchozí kapitola probírala základní terminologii, obecné technologie a metody realizace multimediálních streamů. Nyní se podívejme na současný stav obrazové technologie, zpracování obrazu, a také co lze výhledově očekávat v nejbližší budoucnosti.

3.1 UHD

Jedním ze zásadních faktorů kvality obrazu je **rozlišení** (angl. *resolution*), které u zobrazovací techniky jako je monitor nebo display, udává maximální počet pixelů, které lze zobrazit. Stejně tak u obrazového materiálu (obrázek, video) mluvíme o jeho rozlišení, tedy skutečné velikosti v jaké je zaznamenán.



Obrázek 3.1. Běžná rozlišení obrazu. Převzato z [14].

Zkratkou **UHD** (*Ultra High Definition*, česky *Ultra vysoké rozlišení*) potom souhrnně označujeme digitální video formáty **4K UHD** (3840 x 2160 pixelů, vyjma filmové produkce) a **8K UHD** (7680 x 4320 pixelů). Je zřejmé, že tato rozlišení přináší nárůst v objemu dat, který chceme přenášet, a také proto klademe vysoké nároky na zpracování, kompresi takovýchto datových toků.

Na obrázku 3.1 pak můžeme vidět srovnání s široce rozšířeným Full HD (*High Definition*) rozlišením, případně SD (*Standart definition*).

Podívejme se, pro představu, na současné nároky na bitrate, který je nutný přenést, pro streamovací aplikace.

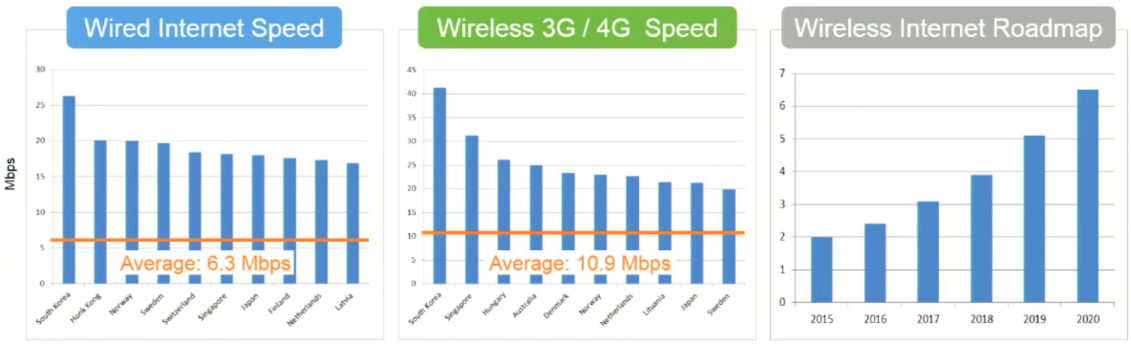
Format	Application	Transmitted Resolution	Codec	Bitrate (Mbps)
SD	Streaming	720 x 480 x 30	AVC	1.5
HDp30	Streaming	1280 x 720 x 30	AVC	3
1080p60	Streaming/Broadcast	1920 x 1080 x 60	HEVC	4
VR Legacy	Streaming	2560 x 1440 x 30	HEVC	15
UHD-1	Streaming/Broadcast	3840 x 2160 x 60	HEVC	20+

Obrázek 3.2. Současné nároky streamovacích aplikací na bitrate. Převzato z [15].

Z obrázku 3.2 je zřejmé, že s nástupem vyšších obrazových rozlišení, nám nároky značně stoupají, i za použití moderních kompresních metod, o kterých si více povíme v kapitole 3.1.1.

Porovnejme si dále tyto údaje s analýzou datového připojení v IP síti běžného uživatele z dané země, jak demonstruje obrázek 3.3.

A nakonec také predikce budoucnosti, kterou lze vidět na obrázku 3.4, kdy již potenciálně budou UHD formáty běžné.



Obrázek 3.3. Analýza uživatelského připojení v IP sítí. Převzato z [15].

Format	Application	Transmitted Resolution	Codec	Bitrate* (Mbps)
VR Tiling HD	Streaming	1920 x 1080 x 60	HEVC	2.6
VR Tiling UHD-1	Streaming	2560 x 1440 x 60	HEVC	9.8
VR Tiling UHD-1 p120	Streaming	3840 x 2160 x 120	HEVC	13.1
UHD-1 p120	Streaming/Broadcast	3840 x 2160 x 120	HEVC	19.7
UHD-2 (8K)	Broadcast	7680 x 4320 x 120	HEVC	65.6
VR 6DoF	Download	undefined	HEVC	>1000

Obrázek 3.4. Odhad nároků na bitrate v budoucnu. Převzato z [15].

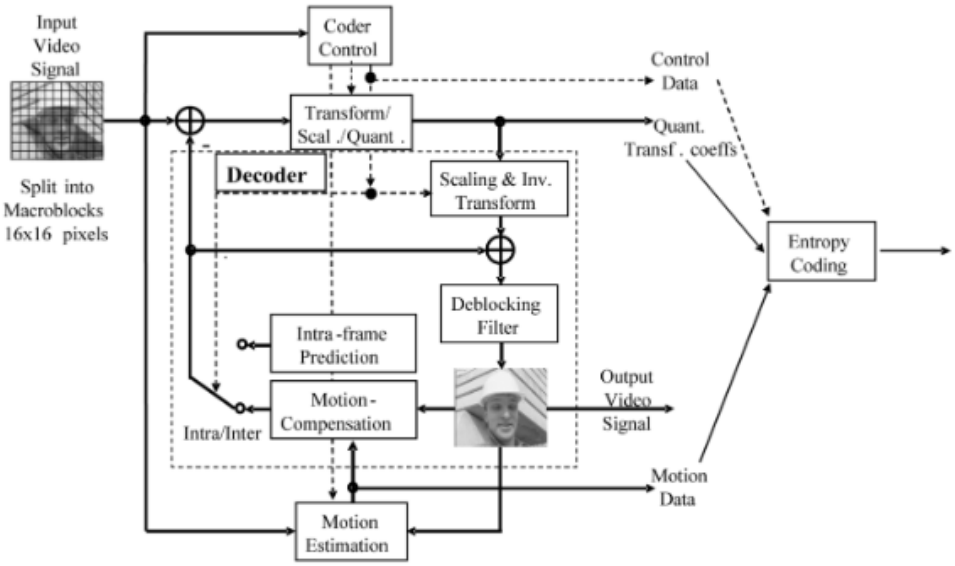
Z těchto analýz lze snadno vyvodit, že nároky na přenesená data za jednotku času nám porostou úměrně se zvyšujícím se rozlišením, které brzy bude běžné pro všední média běžného uživatele.

V další kapitole se podíváme, jak se s tímto problémem vypořádat, neboť kompresní metody jako MPEG-2, popisované v sekci 2.2.1, již na to nestačí, a potřebujeme nástroje nové.

3.1.1 Kompresní metody pro UHD

Jako fundament kompresních metod jsme uvedli MPEG-2, v sekci 2.2.1, který je vhodný pro demonstraci funkce kodeků. Metody, které si popíšeme, jsou si často dost podobné, a rozebereme tedy, co přináší nového, jaké nové funkce implementují, a díky kterým dosahují lepších výsledků.[16]

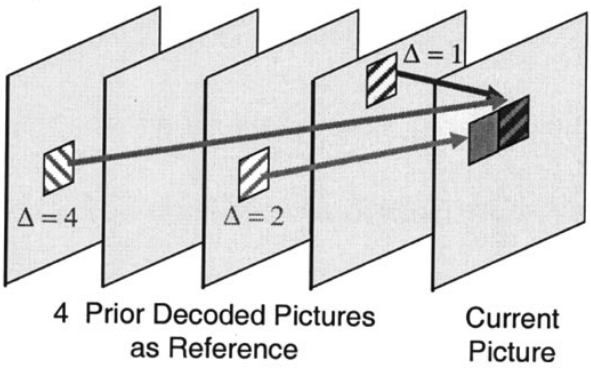
– **H.264/AVC** (MPEG-4 Part 10, *Advanced Video Coding*) – Standard pro kódování videa, jenž vznikl ve spolupráci ITU-T (International Telecommunication Union) a ISO/IEC (International Organization for Standardization/International Electrotechnical Commission); finální verze tohoto standartu je již z roku 2003 a je dodnes silným nástrojem pro zpracování obrazu.[16]



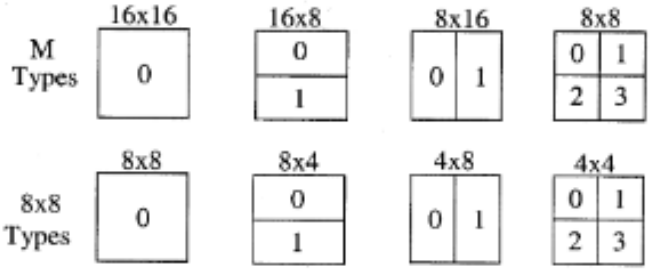
Obrázek 3.5. Blokové schéma kodéru H.264/AVC. Převzato z [16].

H.264 rozvíjí predikci pohybu, jak ji známe z MPEG-2. Je flexibilnější ve využívání již referenčních snímků; pohybové vektory směřovali pouze do oblastí již dekódovaných snímků, s H.264 lze již extrapolovat i z aktuálního snímku. Obrázek 3.6 pak demonstruje vícesnímkovou kompenzaci pohybu. Tento koncept je aplikován i na B-snímky.[16]

Segmentace na makrobloky pixelů (16x16 a 8x8), které se dále dělí a zpracovávají jako tzv. transformační jednotky, je v H.264 detailnější, a umožňuje práci (primárně) s bloky 4x4, výjimečně i 8x8. Obrázek 3.7 ukazuje možnosti segmentace. [16]

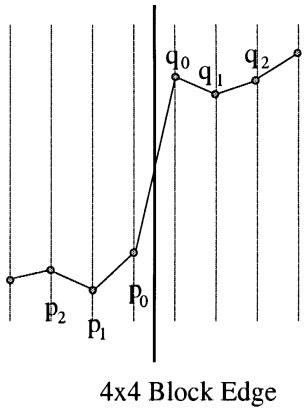


Obrázek 3.6. Vícesnímková kompenzace pohybu v H.264/AVC. Převzato z [16].



Obrázek 3.7. Segmentace makrobloků v kodéru H.264/AVC. Převzato z [16].

Blokové zpracování obrazu s sebou přináší také nežádoucí obrazové artefakty při dekódování, které vznikají predikční i residuální diferencí. Jde o zpracování makrobloků, zvláště pak při jeho hranách. Proto H.264 zavádí tzv. In-loop deblocking filtery, které (pokud jsou správně navrženy) potlačují tyto artefakty a zvyšují tak objektivní i subjektivní kvalitu obrazu.[16]



Obrázek 3.8. Princip deblocking filteru. Převzato z [16].

V neposlední řadě využívá H.264 vylepšené aritmetické entropické kódování. Jde o metody **CABAC** a **CAVLC** (Context-adaptive binary arithmetic coding a Context-adaptive variable-length coding); bezeztrátové kompresní metody, použití záleží na vybraném profilu komprese H.264. CAVLC používá Baseline profile, díky jeho menší náročnosti a procesnímu výkonu potřebnému k dekódování. [16]

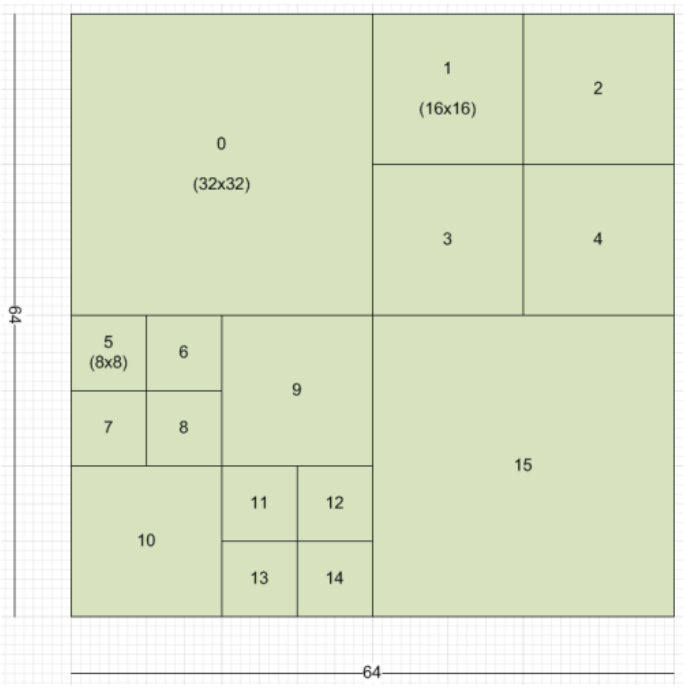
Jde tedy již o překonanou, nicméně stále velice rozšířenou kompresní metodu, která je ovšem již schopna zpracovávat obraz v 8K UHD rozlišení. Bitové toky jsou ale stále vysoké, a tedy zcela nevhodné pro streamovací aplikace.

– H.265/HEVC (*High Efficiency Video Coding*) – Nástupce standartu H.264, nadále vyvíjen specializovanými skupinami z ITU-T a ISO/IEC, z roku 2013. Jde aktuálně o nejmodernější metodu z těchto dílen, která je široce implementována v aplikacích pracujících s UHD, jako jsou UHDTV (Ultra High Definition Television), v Česku nyní aktuální přechod na pozemní vysílaní digitální televize DVB-T2, a samozřejmě i streamovací služby.

Tak jako H.264, i H.265 se inspiruje svými předchůdci a přebírá spoustu jejich funkcí a implementuje nové; rozeberme je.

V první řadě H.265 implementuje novou segmentaci na makrobloky. Analýzy ukazuji, že práce s většími makrobloky vykazuje v jistých případech lepší, efektivnější výsledky

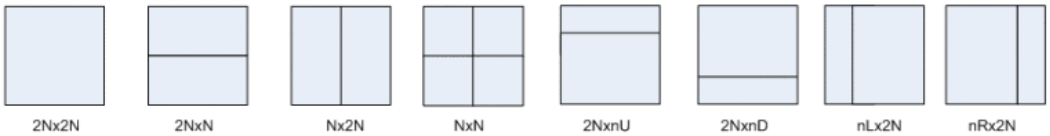
komprese (ale také delší kódování obrazu). Proto H.265 implementuje práci s bloky 64x64, segmentovatelné až na bloky 8x8, jak lze vidět na obrázku 3.9. Tyto nejmenší, tzv. kódovací jednoty, jsou pak používány zvláště na hranách, nebo obecně místech s jemnými detaily. Větší, tzv. kódovací stromové bloky, jsou pak pro vetší plochy s minimem přechodů. [17]



Obrázek 3.9. Segmentace makrobloků v kodéru H.265/HEVC. Převzato z [17].

Další segmentace na transformační jednotky, které už prochází DCT, se také liší. Zatímco H.264 pracoval hlavně s jednotkami o velikost 4x4, v jistých případech i 8x8, tak H.265 jich využívá hned několik 32x32, 16x16, 8x8 i 4x4. Z matematického hlediska jsou větší jednotky pro stacionární signál, menší jednotky pak pro menší „impulsní“ signály.

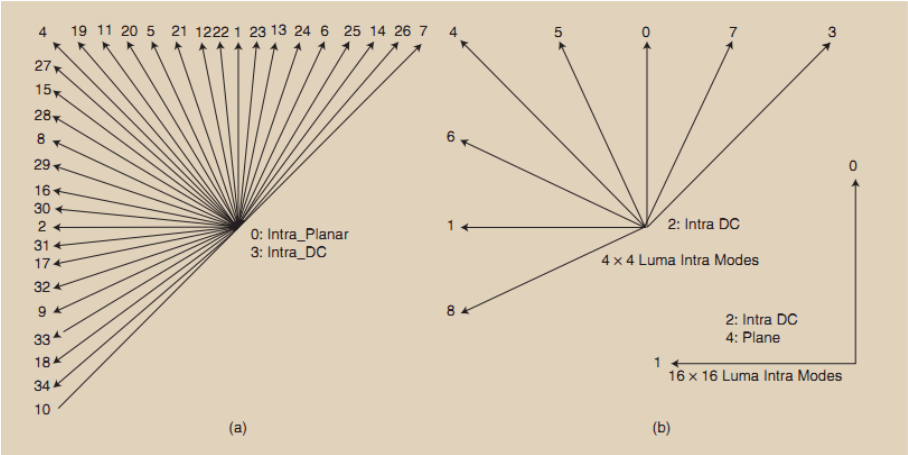
Před transformací a kvantizací dochází ještě k další dělení (viz 3.10) na tzv. predikční jednotky, využíváme exkluzivně k **inter-frame** (mezi snímky) a **intra-frame** (uvnitř snímku) predikci. [17]



Obrázek 3.10. Segmentace na predikční jednotky v kodéru H.265/HEVC. Převzato z [17].

H.265 dále implementuje detailnější vektorizaci v intra predikci, [1] o 35 módech (oproti 9 módům v H.264): DC (stejnosměrná složka), planární a 33 směrových vektorů. [17]

[1] Jde o predikci uvnitř snímky z okolních bloků.



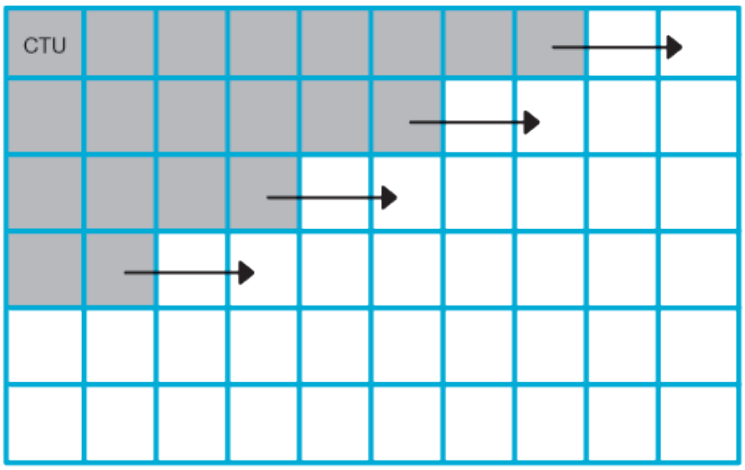
Obrázek 3.11. Vektory intra predikce v kodéru H.265/HEVC. Převzato z [17].

Deblocking filtr je podobný jako u H.264, je ovšem aplikovaný pouze na bloky 8x8 (H.264 aplikovala na 4x4), umožňující tak paralelní zpracování (nedochází k překrývání filtrů.

Zároveň H.265 implementuje další filtr **SAO** (Sample Adaptive Offset) aplikovaný taktéž do predikční smyčky, a jeho úkolem je korigovat chybné predikce a barevné artefakty.

Zakódování do podoby toků bitů probíhá v H.265 pouze pomocí **CABAC** ve všech profilech, pouze s malou zjednodušující úpravou umožňující **paralelní dekódování**.

Poslední zásadní změnou je tedy paralelní kódování a dekódování. Zakódované bloky jsou vyčítány po řádcích, nikoliv však jeden po druhém, ale způsobem připomínající pipelining, jak znázorňuje obrázek 3.12. [17]



Obrázek 3.12. Paralelní zpracování kódovaných bloků v kodéru H.265/HEVC. Převzato z [17].

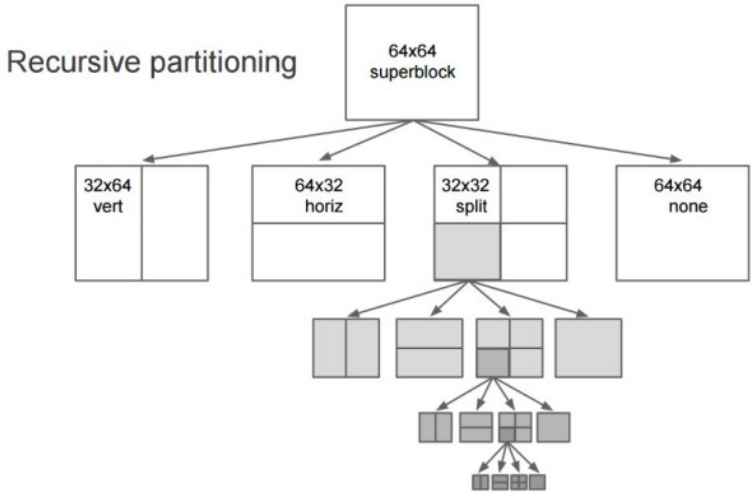
H.265/HEVC je v současnosti předním představitelem moderních kompresních metod, běžně aplikovaných u služeb pracujících s UHD, a tak jako jeho předchůdci patří k těm nejvíc rozšířeným.

Podívejme se dále také na konkurenty, nabízející alternativu k H.265.

– VP9 – VP9 je kompresní metoda vyvinuta společností Google, která byla původně implementována na video platformě YouTube a poprvé byla vydána v prosinci roku

2012 (zhruba 5 měsíců před uvedením H.265). Je nutné podotknout, že VP9 je také zcela zdarma, nezatížený patenty, což z něj taky děla zajímavou alternativu k H.265. Nyní k jeho vlastnostem.

Obdobně jako H.265, VP9 dělí obraz do tzv. super bloků (64x64), které mohou být rekurzivně děleny až na rozměr 4x4. Dovoluje taky dělení na „nečtvercové“ bloky (32x16, 8x16 apod.).[18]



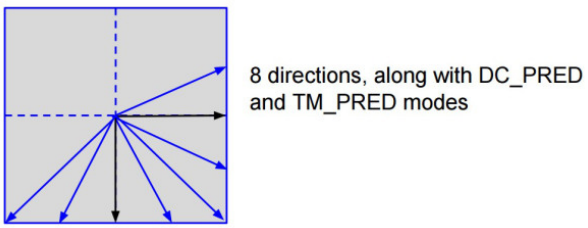
Obrázek 3.13. Rekurzivní dělení na bloku kodéru VP9. Převzato z [18].

VP9 využívá 8-bitový aritmetický kódovací engine známý jako bool-coder. Používá statický per-frame (snímek po snímku) statistický model oproti adaptivnímu CABAC používaného H.264/H.265. Pro každý snímek se použije jeden ze 4 statistických modelů, který je pro daný snímek vhodný.[18]

Podobně jako H.265 používá VP9 4 transformační velikosti bloků: 32x32, 16x16, 8x8 a 4x4; transformované na koeficienty DCT nebo DST [1], podle typu a rozměru snímku. Koeficienty jsou vyčítány podobně jako u H.265 - nikoli zig-zag, ale podobnou logikou.

Pro kvantizaci využívá 4 škálovací faktory: Luma DC a AC koeficienty (jasové složky) a Chroma DC a AC koeficienty (barevné složky), použité na snímkové velikosti, ne adaptivní na velikosti bloků jako H.264/H.265. Obsahuje také speciální bezeztrátový mód, při kterém se využívá Walshovi transformace na blocích 4x4.

Intra predikce aplikovaná na transformační jednotky je jednoduší než u H.265, využívá 8 směrových predikčních vektorů, 2 nesměrové.[18]



Obrázek 3.14. Intra predikce kodéru VP9. Převzato z [18].

Pokud jde o inter predikci, používá VP9 citlivější kompenzaci pohybu. Neumí používat obousměrnou predikci a kompenzaci, a každý blok má pouze jeden vektor. Nicméně

[1] Diskrétní sinová transformace.

používá systém zvaný **compound prediction** (složená predikce), která využívá dvou vektorů, jako průměr ze dvou predikcí. Toho se využívá pouze na neviditelných snímcích (označovaných jako „AltRef“). Takovýto snímek vzniká během dekódování, není vidět, ale je využitelný pro predikci - v compound módu. VP9 tedy oficiálně nemá B-snímky, ale v praxi jde o něco podobného.

Komplexností jsou pohybové predikční vektory podobné H.265.

Posledním zvláštním prvek je možnost **segmentace** - totiž seskupování dohromady bloky, s podobnými charakteristickými vlastnostmi, čímž dochází k optimalizaci při kódování informace (a také psycho-vizuální optimalizaci).[18]

Může tedy VP obstát v porovnání s H.265? VP9 skrývá určitý potenciál, který z části závisí na reálném kodéru. Nutné je však podotknout, že z pohledu objektivních testů, se VP9 drží někde mezi H.264 a H.265, v jistých případech dosahuje i lepších výsledků.

Nejvíc bude ztrácet v intra snímkové predikci (malý počet módů) a entropickém kódování (statické tabulky oproti adaptivnímu kódování. Kladně se pak jeví z hlediska psycho-vizuální optimalizace a řízení datové toku díky segmentaci a adaptivnímu rozlišení snímku.

Obrázek 3.15. Oficiální logo AOMedia Video formátu.

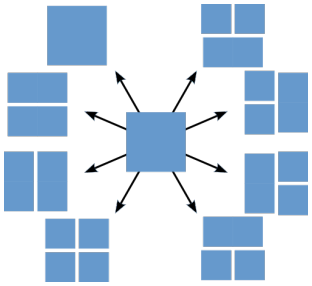
– **AV1** (AOMedia Video 1) – Jde kompresní metodu z dílem AOMedia (Alliance for Open Media), což je neziskové konsorcium vytvořené právě za účelem vytvoření technologie pro šíření mediálního obsahu, bez licenčních poplatků. Zakládajícími členy jsou firmy jako Amazon, Cisco, Google, Intel, Microsoft, Mozilla či Netflix, a mnoho dalších.

S technického hlediska jde o nástupce VP9, a zatím nejslibnější bezplatnou alternativu ke komerčně rozšířenému H.265/HEVC. Navazuje a přejímá prvky kompresních metod Daala (Xiph.Org), Thor (Cisco Systems) a VP10 (Google).

AOMedia již publikovala i bezplatnou implementaci kodéru psanou v programovacím jazyce C, a nabízí také možnost podílet se na vývoji nehledě na členství v AOM. Vedle toho také existuje open source kodér *rav1e*, zaměřený na snadnou a rychlou implementaci AV1, za cenu dosažitelné účinnosti.

Nyní tedy ke zpracování obrazu - transformace dat do frekvenční domény využívá již známé čtvercové a obdélníkové DCT i nesymetrické DST na bloky na jejichž vrchní anebo levé hraně očekáváme nižší chybovost díky predikci z blízkých pixelů. Může kombinovat dvě jednodimenzionální transformace za účel použití jiných transformací na horizontální a vertikální kmitočty.

Predikovat můžeme z větších bloků (až **128x128**), dále segmentovatelné na menší bloky - implementuje nové **dělení „ve tvaru T“** (viz obrázek 3.16), vyvíjené původně pro VP10.



Obrázek 3.16. Segmentace kodéru AV1. Převzato z [19].

Dvě různé predikce lze použít na prostorově odlišné části bloku použitím hladké přechodové čáry tvaru klínu, eliminující tak typický schodovitý přechod na hranách čtvercových bloků.

AV1 provádí také vnitřní zpracování s vysokou přesností (10 nebo 12 bitů na vzorek), což vede ke zlepšení kvality komprese, díky menším zaokrouhlovacím chybám v referenci. Lze kombinovat predikce v jednom bloku (díky compound prediction), včetně jemných či ostrých přechodů hran, díky výše zmíněné segmentaci ve tvaru klínu (wedge-partitioned prediction); dále také díky implicitním maskám, vytvořených na základě rozdílu dvou předchozích prediktorů. To umožňuje kombinaci dvou inter predikcí či jedné inter a jedné intra predikce v témže bloku.

Nástroje *Warped Motion* a *Global Motion* slouží k redukci redundantní informace v pohybových vektorech pomocí rozpoznání vzorových pohybů v pohybech kamery.

Intra predikce využívá **56 směrů** vektorizace pro směrovou predikci a váhových filtrů na pixelovou extrapolaci. AV1 také implementuje **Paeth** prediktor, který porovnává hodnoty pixelů přímo nad a přímo vlevo od levého vrchního pixelu daného bloku a vybere ten s menší změnou za prediktor. Za účelem minimalizace diskontinuit (nesourodosti) na hranách inter-predikovaných bloků, lze prediktory překrývat a mísit se sousedními bloky (overlapped block motion compensation). [20]

Jako in-loop filtr (v predikční smyčce) se využívá tzv. **Constrained Directional Enhancement Filter** - je navržený po vzoru omezeného filtru dolní propust z projektu Thor a směrový deringing filtr [1] z projektu Daala. Jeho účelem je vyhlazovat hrany signálu, a korigovat tzv. ringing artefakty. Dále obsahuje také tzv. loop restoration filtr odstraňující artefakty rozmazávající obraz v důsledku zpracovávání obrazového bloku.

Pokud jde o entropického kódování, AV1 implementuje nebinární aritmetický kodér z projektu Daala (namísto binárního z VP9). Nebinární aritmetický kodér umožňuje se vyhnout patentům (a zůstat bezplatný) a zároveň paralelní bitové zpracování jinak sériového procesu, čímž se redukuje clock rate (hodinový takt) při hardwarové implementaci. [21]

AV1 má potenciál být obrovským přínosem pro oblast moderního zpracování obrazu s ultra vysokým rozlišením. Již nyní objektivní testy ukazují, že se tento volně dostupný kompresní formát dostává do popředí a možná budeme svědky v dalších měsících velkého rozmachu.

[1] Ringing artifacts - jde o artefakty při zpracování digitálního obrazu, vznikajícího při ostrých přechodech signálu; jde o „duchy“ poblíž hran. V analogové formě jde o překmit signálu při impulsní, příliš rychlé změně.

3.1.2 Budoucnost kompresních metod a JVET

Ve zvláštní sekci zmíníme možnou budoucnost kompresních metod. Skupina JVET (Joint Video Experts Team) sestávající se z týmů VCEG (Video Coding Experts Group) z ITU-T a specializovaná skupina z ISO/IEC znovu spojila síly, a přichází myšlenkou, jak překonat stávající „kompresní zeď“ před kterou nyní stojí a to bez nové kompresní metody.

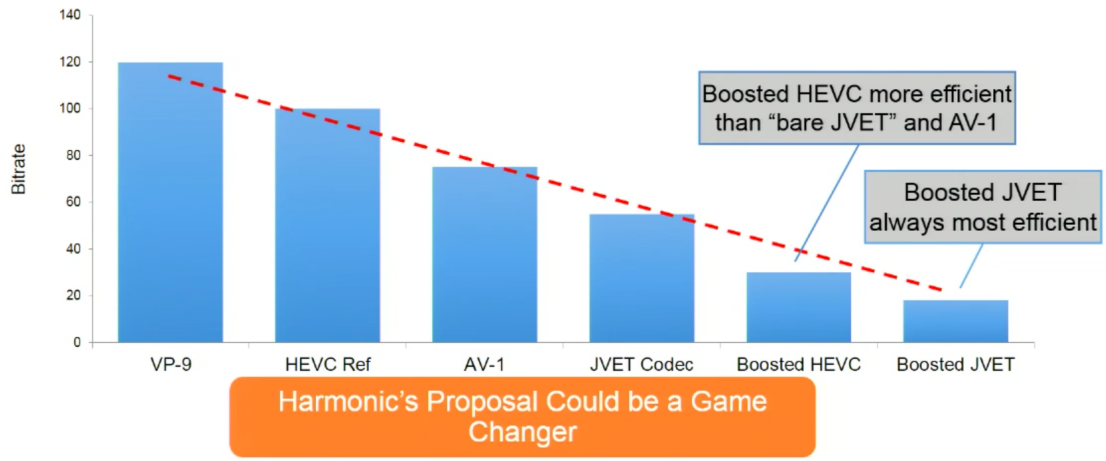
Nástroje, které by JVET chtěl implementovat jsou takovéto:

– **Elastické kódování** (angl. *Elastic encoding*) – Jde o metodu kódování, které adaptabilně navýší procesní výkon, pokud je kódovaný obsah složitější, a vice versa.

– **Strojové učení** – Naučit kodér jak zakódovat informaci s pomocí technik umělé inteligence.

– **Pre/post processing** – Pomocí standardních metadat spárovat funkčnost pre processor a post processor.

– **Content aware** (česky *Povědomí o obsahu*) – Bitová distribuce založená na psychovizuálním modelu, jenž by tak měl povědomí o zpracovávaném (kódovaném) obsahu.



Obrázek 3.17. Analýza vývoje H.265 podle plánu JVET. Převzato z [15].

Zda se tato představa stane skutečností ještě nevíme, graf na obrázku 3.17 vypadá slibně, na reálné výsledku si zatím musíme počkat. V každém případě závod v nejúčinnějším zpracování obrazu zdaleka nekončí.

3.2 HDR

HDR neboli **High-dynamic-range** (česky *Vysoký dynamický rozsah*) video – je metodou zpracování obrazu, která umožňuje zachycení světla v obrazu v jeho plném rozsahu, přenést ho a také znovu zobrazit.

Princip zachycení spočívá v kombinaci dvou obrazů, zachycených s různou světelnou expozicí. Tento princip je znám a zkoumán již od 90. let minulého století, nicméně v kombinaci s ultra vysokým rozlišením se opět dostáváme k vysokým datovým tokům, představující asi čtyřnásobek běžného SDR (Standard dynamic range).

Není tedy překvapením, že zpracování takového obrazu je poněkud výzvou. Jediný nekomprimovaný snímek ve Full HD rozlišení (1920x1080) představuje 24 MB dat, minuta záznamu při snímkové frekvenci 30 fps pak 42 GB. Jedním ze způsobů jak se s takovýmto datovým tokem vypořádat je komprese implementující tzv. **PTF** (Power transfer

function) - lidský vizuální systém zpracovává jas nikoli lineárně, ale logaritmicky; to v praxi znamená, že vaše oko je citlivější na relativní změnu v tmavých oblastech spíš, než v těch světlých. PTF se pak snaží dávat větší váhu HDR obrazu v těch místech, kde je náš vizuální systém nejcitlivější. [22]

3.3 360-stupňové video

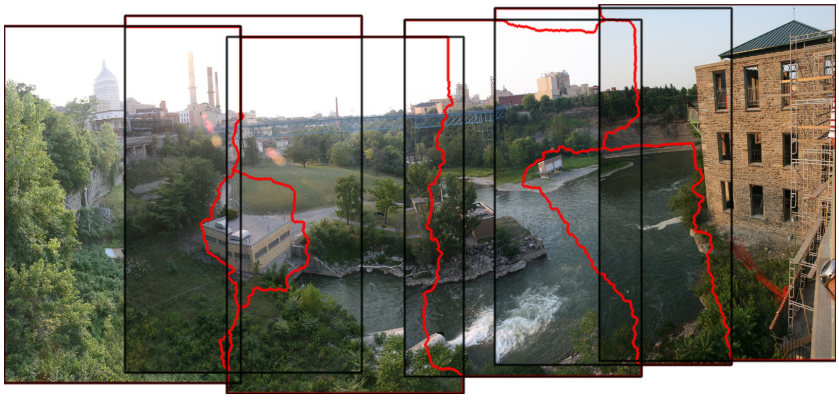
Tato videa se stala zajímavým trendem posledních let. Jde o speciálně snímaná panoramatická videa, která v jeden moment snímají všechny směry, a díky uživatelskému rozhraní si divák sám v reálném čase vybírá, kterou část panoramata chce v daný moment sledovat.

3.3.1 Tvorba a reprezentace

Existují minimálně dvě možnosti, jak takováto videa snímat - buďte všesměrovou kamerou a nebo aparát, vytvořený z několika samostatných kamer (např. firma GoPro přímo takovéto sety vyrábí, viz obrázek 3.18).

Obraz panoramata je pak sestaven z jednotlivých snímků metodou zvanou **stitching** (česky *sešívání*). Principem je najít překryv jednotlivých snímků a postupně vytvořit jediné sférické obraz, video, viz 3.19. Pro co nejlepší výsledek, je nutné mít celý set zkalibrovaný - od kvality nahrávání, kontrastu po barevného vyvážení.

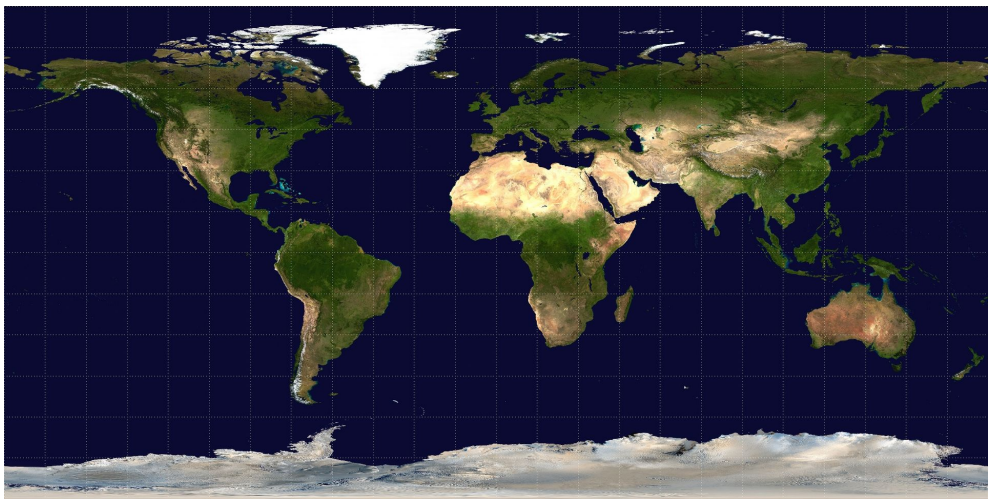
Obrázek 3.18. Kamerový set H3PRO – 7HD firmy GoPro pro 360-stupňové video. Převzato z [23].



Obrázek 3.19. Princip metody stitching. Převzato z [24].

Výsledkem ovšem bude plocha - totiž dvoudimenzionální panoramatický snímek. Sférický obraz nám následně vznikne pomocí tzv. **ekvidistantní válcové projekce**, což je speciální válcová projekce, využívající se zejména v kartografii. Nejde o plochojevná

projekci ani konformní zobrazení. Zjednodušeně si tento proces lze představit jako rozvinutí glóbusu do plochy (viz obrázek 3.20). V případě vytvoření sférického videa jde pouze o proces opačný. [25]



Obrázek 3.20. Ekvidistantní válcová projekce planety Země. Převzato z [26].

Ke zpětnému přehrání těchto videí je zapotřebí služba, mediální přehrávač, která tato videa umí interpretovat. Před přehráním často dochází k injekci metadat, který říkají, že daný obsah má být zobrazen jako sférické video.

Např. příkaz pro upload na službu YouTube pomocí softwaru FFmpeg:

```
ffmpeg -i input_file.mkv -c copy \
-metadata:s:v:0 stereo_mode=1 output_file.mkv
```

Momentálně existuje několik služeb, které podporují interpretaci 360-stupňových videí. Mezi přední průkopníky patří Google (s jejich službou YouTube), jejichž projekt běží od roku 2015 a spolupracují například také s výrobci kamer. Mezi další se pak řadí Facebook (září 2015), Vimeo (březen 2017) nebo například VideoLAN s poslední verzí jejich mediálního přehrávače VLC 3.0.

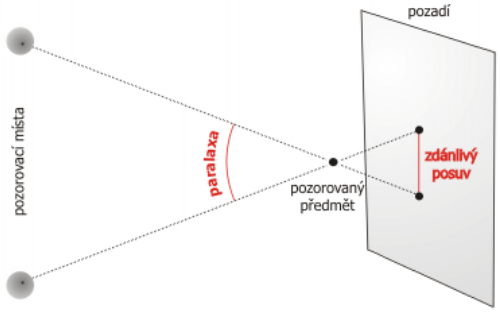
360-stupňová videa se také pojí s rozmachem VR technologie [1] a za zmínku stojí i podpora Androidu a smartphonů, využívajících gyroskopů k ovládání a prohlížení 360-stupňového obsahu.

3.3.2 Stitch lines a paralaxa

Spolu se stitchingem se pojí také nepříznivé jevy, v podobě tzv. stitch lines („stehů“), 3.21 . Jde o nedokonalost obrazu, která vzniká díky nepřesnému pojení videí nebo také vlivem paralaxy. Pokud pozorujeme ze dvou různých míst - v našem případě jde o dvojici čoček sousedících kamer; předmět se nám oproti pozadí pohybuje. Paralaxou pak označujeme úhel mezi spojnicemi od bodů pozorování, k pozorovanému předmětu, někdy také zdánlivou změnu polohy pozorovaného předmětu vůči pozadí. Čím blíže je předmět, tím větší je paralaxa a vice versa.[25]

[1] Virtuální realita

Obrázek 3.21. Názorná ukázka tzv. stitch lines. Převzato z [27].



Obrázek 3.22. Paralaxa. Převzato z [28].

Kapitola 4
Praktická část - vlastní stream

Praktická část této práce je věnována vytvoření streamovacího řetězce a odstreamování vlastního obsahu na distribuční server.

K tvorbě obsahu v reálném čase jsem využil kamery **Blackmagic Pocket Cinema Camera**, nastavenou k záznamu v rozlišení Full HD (1920x1080) při snímkové frekvenci 25 fps (po dohodě s vedoucím práce, jsem realizoval stream pouze ve Full HD rozlišení.). Tato kamera byla připojena do PC přes rozhraní HDMI do karty pro záznam Blackmagic Design Intensity Pro 4K, umožňující zpracování obrazu až po 4K UHD rozlišení. Jako softwarové zpracování obsahu - kodér a odeslání dat posloužil freeware pro zpracování obrazového a zvukového materiálu FFmpeg.

FFmpeg byl verze N-80117-gdac030d (z roku 2016) s konfigurací `--enable-decklink`, implementující knihovnu s příkazy pro karty Blackmagic.

Obrázek 4.1. Verze a konfigurace FFmpeg.

FFmpeg je ovládaný z Příkazové řádky. Níže popisuji potřebné příkazy k realizaci samotného streamu:

```
ffmpeg -f decklink -list_devices 1 -i dummy
```

Příkaz, který vyčítá seznam připojených zařízení typu Blackmagic (`-f decklink`).

```
ffmpeg -f decklink -list_formats 1 -i "Intensity Pro 4K"
```

Tento příkaz poskytuje výčet formátů, které dané zařízení schopné zpracovat. [1]

Každý formát (určený rozlišením a snímkovou frekvencí) je **indexovaný** - tento údaj je pro nás podstatný.

Nyní když víme, jak zpřístupnit naše zařízení, můžeme přejít ke streamu:

```
ffplay -f decklink -i "Intensity Pro 4K@5"
```

Protože FFmpeg má implementovaný i vlastní mediální přehrávač ffplay.exe, sloužil mi tento příkaz k testu, že zařízení správně funguje a FFmpeg přijímá obraz ke zpracování. `@5` je zmiňovaný index, určující používaný formát.

[1] V oficiální dokumentaci jsou uvedeny pouze apostrofy, které v mé práci nikdy nefungovaly. Proto vždy používám klasické uvozovky.

Příkaz pro vytvoření samotného streamu vypadá asi takto:

```
ffmpeg -f decklink -i "Intensity Pro 4K@5" -af 44100 -f flv - |
ffmpeg -f flv -i - -c:v copy -copyts -c:a copy \
-f flv "rtmp://a.rtmp.youtube.com/live2/STREAM_KEY"
```

V tuto chvíli již stream běží. Struktura je trochu složitější, rozeberme si tedy příkaz podrobněji: ve skutečnosti jde o dvě instance FFmpegu a využitím metody *pipování*, si data předávají.

První instance je určená vstupem, kterým je naše karta přijímající obraz z kamery, parametr `-af 44100` nastavuje vzorkovací kmitočet zvuku na 44,1 kHz. `-f flv` určuje výstupní formát a na místo výstupu je příkaz zakončen pouze znakem `-`, tedy pipe.

Druhá instance instance takto připravený obsah vezme ze vstupu - pipy (`-i -`), nemění jeho formát (`-f flv`) ani jeho parametry (`-c:v copy -c:a copy -copyts`, tzn. přebírá použitý audio a video kodek, a dále také time-stampy) a se stejným formátem ho posílá na výstup, kterým je pro nás streamovací server služby YouTube. Za hodnotu `STREAM_KEY` se pouze doplní unikátně generovaný klíč pro daný stream, poskytnutý také YouTube.

Jak by se proces změnil pro UHD obsah? Protože karta Intensity Pro 4K podporuje UHD, byl by prvním krokem získání indexu korelujícího s požadovaným formátem. A dále bychom pomocí správných parametrů pozměnili první z příkazů.

Parametr `-c:v libx265` využívá knihoven implementující kodéru H.265/HEVC. Dalším parametrem který můžeme přidat by mohl být CRF (Constant Rate Factor) umožňující zachovat vysokou kvalitu, za cenu většího datového toku.

Dalších úprav můžeme dosáhnout zavedení zvukových (`-b:a`) nebo obrazových (`-b:v`) bufferů, kodeků (`-c:v, -c:a`) a dalších, pro přizpůsobení konkrétním požadavkům.

V poslední řadě bych rád zmínil pokus o vlastní **360-stupňové video**, realizovaný také čistě v FFmpeg. S využitím druhé kamery Blackmagic Pocket Cinema Camera připojené pomocí HDMI do externí karty Blackmagic Intensity Shuttle, a přes rozhraní USB do PC.

```
ffmpeg -f decklink -i "Intensity Pro 4K@5" -f decklink \
-i "Intensity Shuttle@8" -hstack -af 44100 -f flv |
ffmpeg -f flv -i - -c:v copy -copyts -c:a copy \
-f flv "rtmp://a.rtmp.youtube.com/live2/STREAM_KEY"
```

Úpravou prvního příkazu jsem tedy ze dvou vstupů, horizontálně „sešitých“ k sobě vytvořil jeden, který je odstreamován na YouTube. Protože YouTube podporuje sféric-kou reprezentaci videí, vše funguje bez problémů.

Kapitola 5
Závěr

Cílem této práce bylo seznámení se s technologií streamování, popsání streamovacího řetězce a se zaměřením na obsah v UHD takovýto stream vytvořit.

V jednotlivých kapitolách je tedy zaměřena nejprve na obecné vlastnosti streamovacího řetězce - popisuje o co jde, jak vypadá a z čeho se sestává. Jakých technologií je potřeba pro dopravu multimediálního obsahu od jeho tvůrce až k samotnému uživateli.

V další kapitole pak práce rozebírá současné moderní metody zpracování obrazu. Popisuje způsoby jak se vypořádat s enormními datovými toky, které představují ultra vysoká rozlišení a technologie jako například HDR. Podává také náhled na možnou budoucnost a vývoj, kterému čelí technici z oboru zpracování obrazu. A nakonec také popisuje tvorbu tzv. 360-stupňových, nebo sférických videí, které jsou zajímavým trendem posledních let.

V poslední řadě pak tato práce popisuje samotnou tvorbu streamu pomocí freewaru FFmpeg.

Za zmínku jistě stojí, že FFmpeg je velice silný nástroj, a představuje výbornou bezplatnou alternativu v oblasti zpracování obrazových obsahů i pro běžného uživatele. Jeho vývojáři již spolupracují i s většími celky, jako je třeba AOMedia, stojící za kompresní metodou AV1.

Po dostatečném seznámení jde o silný, rychlý ale i jednoduchý nástroj s velkými možnostmi. Při správném nastavení parametrů nebyl problém vytvořit stream a prakticky si otestovat jeho funkčnost. Pokus s 360-stupňové video se dá považovat za úspěch, byť použité prostředky nebyli nejvhodnější: je zřejmé, že každá z použitých kamer jistě nebude mít 180-stupňový záběr; Full HD rozlišení pak představuje asi poloviční, než které by bylo potřeba pro tvorbu kvalitního obsahu. Přes tato fakta je výsledkem práce poukazující možnosti implementaci FFmpeg (nejen) v oblasti moderního zpracování obrazu.

Mé náměty na zdokonalení představují rozšíření počtu kamer, a tím lepší kvalitu a pokrytí všesměrového záběru. Dále pak implementovat další software zaměřený na stitching obrazu, a FFmpegu tak nechat na starost pouze zpracování obrazu jako celku spolu s transportem dat na server.

Literatura

[1] David Austerberry. *The Technology of Video and Audio Streaming*. Oxford: Focal Press, 2005. ISBN 978-0240805801.

[2] Wes Simpson. *Video Over IP*. Oxford: Focal Press, 2008. ISBN 978-0-240-81084-3.

[3] Fairhurst Gorry. *Unicast, Broadcast, and Multicast*. http://www.erg.abdn.ac.uk/users/gorry/course/intro-pages/uni-b-mcast.html.

[4] DataGenetics. *Discrete Cosine Transformations*. http://datagenetics.com/blog/november32012/index.html.

[5] Jean-Guillaume Dumas, Jean-Louis Roch, Eric Tannier a Sebastien Varrette. *Foundations of Coding: Compression, Encryption, Error Correction*. New Jersey: Wiley, 2015. ISBN 978-1-118-88144-6.

[6] Daniel Mlynek a Yusuf Leblebici. *Design of VLSI Systems, Chapter 13 - Architectures for video processing*. http://book.huihoo.com/design-of-vlsi-systems/ch13/ArchiMultimed.htm.

[7] Vasily Moshnyaga a Shigeaki Yamaoka. *MPEG complexity reduction by scene adaptive motion estimation*. 2006. https://www.researchgate.net/publication.

[8] Roman Berka, František Rund, Libor Husník a Adam Sporka. *Multimédia I*. Praha: České vysoké učení technické v Praze, 2016. ISBN 978-80-01-05859-6.

[9] J. Postel. *Dokument IETF RFC 768*. https://tools.ietf.org/html/rfc768.

[10] H. Schulzrinne, S. Casner, R. Frederick a V. Jacobson. *Dokument IETF RFC 3550*. https://tools.ietf.org/html/rfc3550.

[11] R. Braden, L. Zhang, S. Berson, S. Herzog a S. Jamin. *Dokument IETF RFC 2205*. https://tools.ietf.org/html/rfc2205.

[12] H. Schulzrinne, A. Rao a R. Lanphier. *Dokument IETF RFC 2326*. https://tools.ietf.org/html/rfc2326.

[13] M. Handley a V. Jacobson. *Dokument IETF RFC 2327*. https://tools.ietf.org/html/rfc2327.

[14] Libron. *Resolution of SD, Full HD, 4K Ultra HD and 8K Ultra HD*. https://upload.wikimedia.org/wikipedia/commons/c/cc/Resolution_of_SD%2C_Full_HD%2C_4K_Ultra_ svg.

[15] T. Fautier. *Next-Generation Video Compression Techniques*. In: *SMPTE 2017 Annual Technical Conference and Exhibition*. 2017. 1-12.

[16] T. Wiegand, G. J. Sullivan, G. Bjontegaard a A. Luthra. Overview of the H.264/AVC video coding standard. *IEEE Transactions on Circuits and Systems for Video Technology*. 2003, 13 (7), 560-576. DOI 10.1109/TCSVT.2003.815165.

[17] Fabio Sonnati. *H265 – part I : Technical Overview*. https://sonnati.wordpress.com/2014/06/20/h265-part-i-technical-overview/.

[18] Fabio Sonnati. *Does VP9 deserve attention – Part I*.
https://sonnati.wordpress.com/2016/06/03/does-vp9-deserve-attention-part-i/.

[19] Dr. Sriram Sethuraman a Cherma Rajan. *Decoding the Buzz over AV1 Codec*.
https://www.ittiam.com/decoding-buzz-av1-codec/.

[20] Urvang Joshi, Debargha Mukherjee, Jingning Han, Yue Chen, Sarah Parker, Hui Su, Angie Chiang, Yaowu Xu, Zoe Liu, Yunqing Wang, Jim Bankoski, Chen Wang a Emil Keyder. *Novel inter and intra prediction tools under consideration for the emerging AV1 video codec*.
https://www.spiedigitallibrary.org/conference-proceedings-of-spie/10396/2274022/Novel-inter-and-intra-prediction-tools-under-consideration-for-the/10.1117/12.2274022.short?SSO=1.

[21] Timothy B. Terriberry. *Progress in the Alliance for Open Media*.
https://people.xiph.org/%7Etterribe/pubs/lca2017/aom.pdf.

[22] "Hatchett. *"The Visual Computer"*. "2018", "34" ("2"), "167–176". DOI "10.1007/s00371-016-1322-0".

[23] Liza Brown. *Review the exclusive 360 video rigs for GoPro*.
https://filmora.wondershare.com/video-editing-tips/360-video-rigs-for-gopro.html.

[24] Noso1. *Example for geometrical registration and stitch line in panorama creation*.
https://upload.wikimedia.org/wikipedia/commons/a/a0/Rochester_NY.jpg.

[25] Michael Morgenstern. *A Step-By-Step Guide to Creating 360-degree VR, from the Makers of the 1 Trillion Pixel Cat Video*.
https://nofilmschool.com/2016/03/360-degree-video-cat-cafe.

[26] NASA. *Equirectangular projection of Visible Earth*.
https://visibleearth.nasa.gov/.

[27] fusion-of-horizons. *Stitch lines*.
https://www.flickr.com/photos/fusion_of_horizons/.

[28] Tlusťa. *Paralaxa*.
https://upload.wikimedia.org/wikipedia/commons/b/b1/Paralaxa.png.