

Sem vložte zadání Vaší práce.



ČESKÉ VYSOKÉ UČENÍ TECHNICKÉ V PRAZE  
FAKULTA INFORMAČNÍCH TECHNOLOGIÍ  
POČÍTAČOVÉ SYSTÉMY A SÍTĚ



Diplomová práce

## **GPU akcelerované vyhodnocení kvality vídea**

*Bc. Václav Fanfule*

Vedoucí práce: Ing. Ivan Šimeček, Ph.D.

9. ledna 2017



---

## **Poděkování**

Děkuji vedoucímu práce Ing. Ivanovi Šimečkovi, Ph.D. za vedení, rady a pomoc při tvorbě této práce. Dále také děkuji rodině a přítelkyni, Jitce Janíkové, za bezmeznou podporu v průběhu tvorby této práce a v průběhu celého studia a všem dalším, kteří jakkoli přispěli ke vzniku této práce.



---

## **Prohlášení**

Prohlašuji, že jsem předloženou práci vypracoval(a) samostatně a že jsem uvedl(a) veškeré použité informační zdroje v souladu s Metodickým pokynem o etické přípravě vysokoškolských závěrečných prací.

Beru na vědomí, že se na moji práci vztahují práva a povinnosti vyplývající ze zákona č. 121/2000 Sb., autorského zákona, ve znění pozdějších předpisů. V souladu s ust. § 46 odst. 6 tohoto zákona tímto uděluji nevýhradní oprávnění (licenci) k užití této mojí práce, a to včetně všech počítačových programů, jež jsou její součástí či přílohou, a veškeré jejich dokumentace (dále souhrnně jen „Dílo“), a to všem osobám, které si přejí Dílo užít. Tyto osoby jsou oprávněny Dílo užít jakýmkoli způsobem, který nesnižuje hodnotu Díla, a za jakýmkoli účelem (včetně užití k výdělečným účelům). Toto oprávnění je časově, teritoriálně i množstevně neomezené. Každá osoba, která využije výše uvedenou licenci, se však zavazuje udělit ke každému dílu, které vznikne (byť jen zčásti) na základě Díla, úpravou Díla, spojením Díla s jiným dílem, zařazením Díla do díla souborného či zpracováním Díla (včetně překladu), licenci alespoň ve výše uvedeném rozsahu a zároveň zpřístupnit zdrojový kód takového díla alespoň srovnatelným způsobem a ve srovnatelném rozsahu, jako je zpřístupněn zdrojový kód Díla.

V Praze dne 9. ledna 2017

.....

České vysoké učení technické v Praze  
Fakulta informačních technologií

© 2017 Václav Fanfule. Všechna práva vyhrazena.

*Tato práce vznikla jako školní dílo na Českém vysokém učení technickém v Praze, Fakultě informačních technologií. Práce je chráněna právními předpisy a mezinárodními úmluvami o právu autorském a právech souvisejících s právem autorským. K jejímu užití, s výjimkou bezúplatných zákonných licencí, je nezbytný souhlas autora.*

### Odkaz na tuto práci

Fanfule, Václav. *GPU akcelerované vyhodnocení kvality videa.* Diplomová práce. Praha: České vysoké učení technické v Praze, Fakulta informačních technologií, 2017.

---

# Abstrakt

Práce je zaměřena na objektivní vyhodnocení kvality videa. V rámci této práce jsou popsány metody jak porovnávat kvalitu různých verzí jednoho videa. Vybral jsem několik různých způsobů porovnávání kvality videa, tyto metody jsem následně implementoval. Pomocí mnou implementovaných metod jsem také změřil různé kodeky za cílem jejich vzájemného porovnání.

**Klíčová slova** Porovnávání videí, kvalita videí, PSNR, SSIM, stVSSIM, CUDA, H.264, H.265, VP9

---

# Abstract

The thesis focuses on an objective evaluation of video quality. It describes methods for comparison of different version of the same video. I have chosen several different ways to compare the quality of video and subsequently implemented these. Using these implemented methods I have also gauged various codecs with the aim of their relative comparison.

**Keywords** Video comparison, video quality, PSNR, SSIM, stVSSIM, CUDA, H.264, H.265, VP9



---

# Obsah

<b>Úvod</b>	<b>1</b>
<b>1 Technologie</b>	<b>3</b>
1.1 Metriky pro porovnávání videa . . . . .	3
1.2 Metrika PSNR . . . . .	6
1.3 Metrika SSIM . . . . .	7
1.4 Metrika stVSSIM . . . . .	10
1.5 Zpracování a uložení videa . . . . .	17
<b>2 Analýza</b>	<b>21</b>
2.1 OpenMP . . . . .	21
2.2 CUDA . . . . .	21
2.3 C++ . . . . .	22
2.4 Srovnání s konkurencí . . . . .	22
<b>3 Implementace</b>	<b>25</b>
3.1 Implementace PSNR . . . . .	26
3.2 Implementace SSIM . . . . .	27
3.3 Paralelizace PSNR a SSIM . . . . .	27
3.4 Implementace stVSSIM . . . . .	29
<b>4 Porovnání algoritmů</b>	<b>31</b>
4.1 Srovnání se subjektivním testováním . . . . .	31
4.2 Srovnání rychlosti algoritmů . . . . .	38
4.3 Srovnání rychlosti metrik . . . . .	41
<b>5 Porovnání kodeků</b>	<b>45</b>
5.1 Vyhodnocení měření . . . . .	50
<b>Závěr</b>	<b>61</b>

<b>Literatura</b>	<b>63</b>
<b>A Seznam použitých zkratek</b>	<b>67</b>
<b>B Obsah přiloženého CD</b>	<b>69</b>

---

# Seznam obrázků

1.1	Teoretický ideální Graf kvalit jednotlivých snímků ve videu . . . . .	5
1.2	Graf kvalit jednotlivých snímků ve videu . . . . .	5
1.3	Srovnání obrázku v různých hodnotách PSNR . . . . .	7
1.4	Diagram procesu tvorby SSIM . . . . .	9
1.5	Vertikální filtr . . . . .	12
1.6	Horizontální filtr . . . . .	13
1.7	Pravý filtr . . . . .	14
1.8	Vertikální filtr . . . . .	15
1.9	Typy podporujících sousedních bloků . . . . .	15
1.10	Podporující sousední bloky u okrajů . . . . .	16
1.11	Graf potenciálních kandidátů na vzor aktuálního bloku . . . . .	17
4.1	VQEG HDTV Sada 1, video snímek 1 až 9 . . . . .	32
4.2	VQEG HDTV, video snímek 11 až 14 . . . . .	33
4.3	VQEG HDTV snímek 2, subjektivně nízká kvalita . . . . .	35
4.4	VQEG HDTV snímek 2, subjektivně vysoká kvalita . . . . .	36
4.5	VQEG HDTV snímek 12, subjektivně nízká kvalita, kostičkování .	37
4.6	VQEG HDTV snímek 12, subjektivně vysoká kvalita . . . . .	38
5.1	Rozdíl h264 a h265, h264 jako referenční, metrika SSIM . . . . .	50
5.2	Rozdíl h264 a h265, průměry všech kodeků jako referenčních, me- trička SSIM . . . . .	51
5.3	Rozdíl h264 a vp9, h264 jako referenční, metrika SSIM . . . . .	52
5.4	Rozdíl h264 a vp9, vp9 jako referenční, metrika SSIM . . . . .	52
5.5	Rozdíl h265 a vp9, h265 jako referenční, metrika SSIM . . . . .	53
5.6	Rozdíl h265 a vp9, vp9 jako referenční, metrika SSIM . . . . .	54
5.7	Rozdíl h265 a vp9, h264 jako referenční, metrika SSIM . . . . .	54
5.8	Rozdíl h264 a h265, h264 jako referenční, metrika stVSSIM . . . .	56
5.9	Rozdíl h264 a vp9, vp9 jako referenční, metrika stVSSIM . . . .	57
5.10	Rozdíl h265 a vp9, vp9 jako referenční, metrika stVSSIM . . . .	58

5.11 Rozdíly mezi kodeky, metrika stVSSIM, 4K video . . . . .	59
---------------------------------------------------------------	----

---

# Seznam tabulek

4.1	VQEG HDTV video 1, mediány . . . . .	34
4.2	VQEG HDTV video 1, průměry . . . . .	35
4.3	VQEG HDTV video 2, průměry . . . . .	36
4.4	VQEG HDTV video 11 až 14, průměry . . . . .	39
4.5	Korelace mediánů pro jednotlivé sady videí . . . . .	40
4.6	Korelace průměrů pro jednotlivé sady videí . . . . .	40
4.7	Korelace průměrů pro videa 11 až 14 . . . . .	41
4.8	Průměrné relativní odchylky při čase výpočtu stVSSIM . . . . .	41
4.9	Souhrnná tabulka srovnání rychlosti metrik, Marvel's Avengers: Age of Ultron - Trailer 3 . . . . .	42
4.10	Nejlepší časy jednotlivých metrik a časy na snímek a film . . . . .	43
5.1	Vybraná část z video setu 1, bez úprav . . . . .	47
5.2	Data z video setu 1, srovnána podle dat vytvořených kodekem vp9	48
5.3	Relativní rozdíly ve velikosti při dané kvalitě pro jednotlivé kodeky a metriky, video set 1, h264 jako referenční kodek . . . . .	49
5.4	Průměr relativních rozdílů ve velikostech dat při stejně kvalitě, h264 jako referenční kodek . . . . .	49
5.5	Průměry relativních rozdílů mezi všemi kodeky . . . . .	55
5.6	Souhrné výsledky pro rozdíly mezi jednotlivými kodeky, metrika stVSSIM . . . . .	56
5.7	Výsledky při omezeném crf pro rozdíly mezi jednotlivými kodeky, 4K video . . . . .	57
5.8	Souhrné výsledky pro rozdíly mezi jednotlivými kodeky, full HD i 4K video . . . . .	59



---

# Úvod

Při práci s videem je často nutné zjistit jakou má toto video kvalitu, prostým subjektivním shlédnutím tohoto videa se ale často nepodaří dostatečně vyhodnotit jeho kvalitu, proto je tedy na místě použít nějaký analytický nástroj, který tuto kvalitu objektivně určí. Existuje mnoho způsobů jak tuto kvalitu určit, já ve své práci provedu analýzu některých metod, vyhodnotím jejich vhodnost, provedu implementaci a otestuji tyto metody.

Jedním z cílů této práce je analyzovat způsoby vyhodnocení kvality videa, konkrétně se zaměřím na objektivní, deterministické způsoby vyhodnocení kvality.

Dalším cílem této práce je porovnání způsobů ukládání videí a to především z hlediska jejich komprese, tato práce se snaží odpovědět na otázku jaký způsob ukládání a komprese videa je nevhodnější, k tomu budou použity mnou navržené metody vyhodnocování kvality videa.

Výstupem této práce bude mimo jiné program určený k vyhodnocení kvality videa, tento program bude schopen porovnat shodnost vzorového (originálního) videa a videí z něj odvozených a tím určit jejich kvalitu. Bude umožnovat i paralelní zpracování, stejně tak jako bude možné využívat grafickou kartu pro akceleraci tohoto programu.

V první kapitole představím nejdůležitější technologie použité dále v této práci, dojde zde k seznámení s pojmy jako kodek nebo formát, v této kapitole dojde i na popis způsobů porovnávání kvality videa. V neposlední řadě v této kapitole dojde na popis později implementovaných algoritmů.

V druhé kapitole provedu analýzu existujících nástrojů a programů, které bych mohl využít v mé práci. Vyhodnotím které z nich jsou nevhodnější a stručně je představím a popíši jejich možnosti.

V následující kapitole bude popis implementace jednotlivých algoritmů použitých v mému programu.

V posledních dvou kapitolách se bude nacházet měření algoritmů a kódů. U měření algoritmů se zaměřím především na to, jak dobře korelují se

## ÚVOD

---

subjektivním vnímáním, u měření kodeků se zaměřím na to, který kodek je nejlepší z hlediska ukládání a komprese videí.

Využitelnost této práce vidím v umožnění porovnávání kvalit videa při jeho různých parametrech, konkrétně například při kompresi videa mi můj program může pomoci s určením požadovaných parametrů pro kompresi.

# Technologie

Tato práce se zabývá objektivním porovnáváním kvality videí. V této kapitole představím technologie, které budu využívat, nejprve je ale nutné říct něco o samotném porovnávání kvality videí. Pod pojmem kvalita videa se v této práci rozumí množina všech vlastností a parametrů videa. Mezi tyto parametry například patří datový tok nebo rozlišení. Není ale jednoduché ze zadaných parametrů určit kvalitu tohoto videa, ani velmi dobré parametry videa (například vysoké rozlišení, velký datový tok) nezaručují dobrou kvalitu videa. V této práci se budu zabývat tím, jak určit, jakou má video kvalitu a především jak jednotlivá videa porovnat mezi sebou z hlediska jejich kvality.

V této práci se zejména zaměřím na porovnávání různých verzí jednoho videa. Protože mi jde mimojiné o porovnání různých forem uložení videa, je pro mě nejdůležitější vědět, o kolik nějaká komprese změnila video, a kvalitu tohoto videa budu posuzovat podle shodnosti s jeho vzorem.

## 1.1 Metriky pro porovnávání videa

V této sekci popíši způsoby srovnávání videa, jak je možné určit jejich shodnost. Pro toto určení využívám metriku. Metrika je číslo, které vyjadřuje vzdálenost dvou objektů. V této práci se pod pojmem metrika rozumí způsob, jak ze dvou objektů (v tomto případě například jednotlivých snímků nebo video záznamů) získat jedno číslo, které vyjadřuje vzdálenost těchto objektů. Z různých algoritmů však vycházejí různé metriky, a proto nejsou mezi sebou porovnatelné. Například u metriky PSNR (detailedy o PSNR jsou v sekci 1.2) platí, že čím jsou objekty shodnější, tím je vyšší metrika a naopak. Ale tato metrika nemá lineární průběh, průběh je spíše exponenciální, pro shodné objekty vychází plus nekonečno, pro zcela neshodné objekty vychází 0.

Obecně se metriky dělí na referenční a nereferenční metriky. U referenčních metrik je vždy k dispozici jedno vzorové video a k tomuto videu vztahujeme všechna srovnávaná videa. U nereferenčních metrik není k dispozici žádné ukázkové video a jsou porovnávána pouze videa mezi sebou, bez znalosti toho,

## 1. TECHNOLOGIE

---

které je výchozí a které je odvozené. U referenčních metrik je dáno, že čím je porovnávané video blíže k vzoru, tím je lepší. Naopak u nereferenčních metrik tento přístup není možný a je možné pouze určit, jak moc jsou si videa podobná. Pro odhad kvality u nereferenčních metrik je možné použít některé heuristiky [1]. Ovšem tyto heuristiky nejsou spolehlivé u všech typů obrázků, proto není vhodné jejich použití pro video, kde nemůžeme zaručit, že všechny snímky budou vhodné pro danou nereferenční metriku.

Jedna z klíčových otázek této práce zní, jak poznat, které video je kvalitnější. Existuje mnoho přístupů k této problematice. Jedním z jednodušších je pouhé porovnání bitratu. Tato metrika je ale velmi nedostatečná, protože nezohledňuje mnoho faktorů, například různé kodeky pro enkódování. Další možností je porovnávání jednotlivých snímků videa a zkoumání jejich shodnosti. Tato metrika je znatelně spolehlivější a převádí problém na to, jak porovnávat jednotlivé snímky dvou videí mezi sebou. Na tuto problematiku existuje mnoho algoritmů od jednoduchého součtu odchylek barev jednotlivých pixelů až po komplexní algoritmy, které by měly zahrnovat i nedokonalé vnímání lidským okem. V této práci jsem zvolil z této oblasti algoritmy PSNR a SSIM (detailedy o SSIM jsou v sekci 1.3). Jedná se pravděpodobně o nejčastěji používané algoritmy pro porovnávání obrázků. Mezi další metriky se řadí například MSE nebo MSSIM, jedná se ale o algoritmy odvozené z PSNR nebo SSIM a nebo o algoritmy podobné[2].

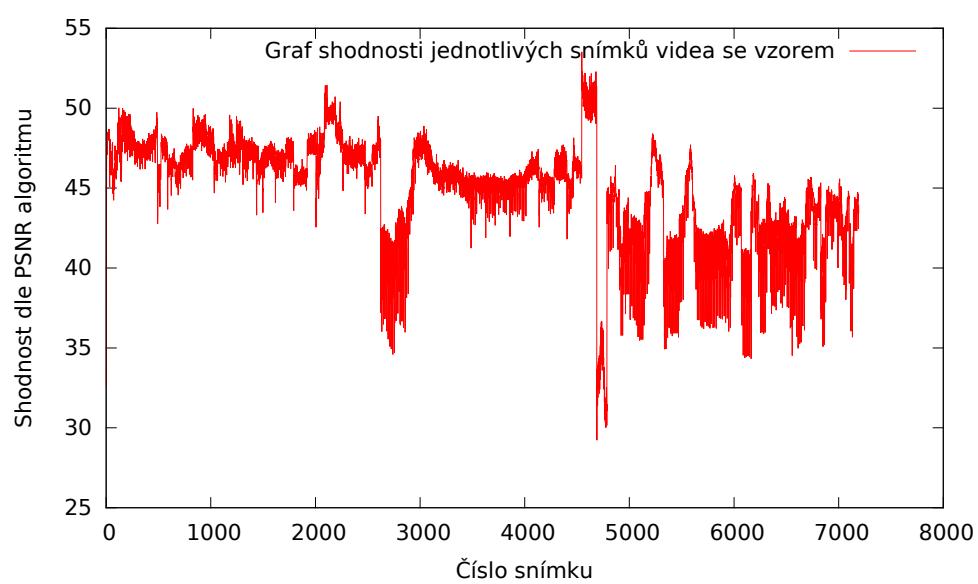
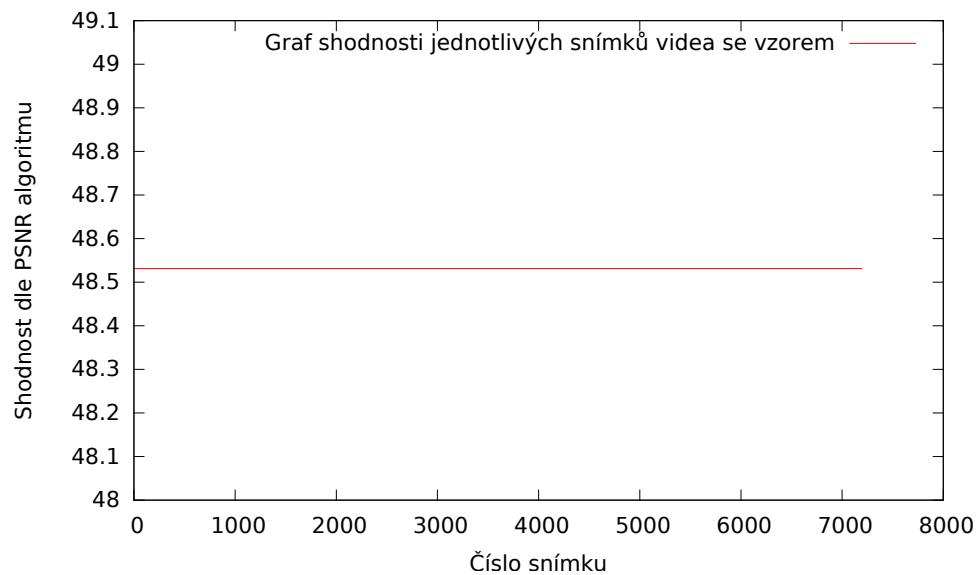
Při použití metrik určených pro porovnávání obrázků je ale potřeba výsledky z porovnávání jednotlivých snímků aplikovat na celé video. K tomuto problému je taktéž možno přistupovat několika způsoby, nejjednodušší je vzít průměr z metrik pro jednotlivé snímky a ten prohlásit za výsledek. Tento výsledek může být ale velmi nepřesný a nedostatečný, například nezohledňuje špičky a výkyvy v jednotlivých snímcích a dále také není dobře použitelný pro metriky, které mají nelineární průběh. O něco lepší je použít medián výsledných hodnot. Medián sice částečně eliminuje některé problémy, které má průměr, ale stále není dostatečný například proto, že nezohledňuje špičky a propady v grafu. Toto je možné pozorovat na obrázcích 1.1, 1.2. V ideálním případě by závislost shodnosti snímku na čase měla konstantní průběh viz 1.1, ale v praxi bude graf vypadat spíše jako na obrázku 1.2. Z tohoto důvodu je nutné přijít s vlastním algoritmem na vyhodnocování těchto dat. Je potřeba zohlednit špičky a propady v kvalitě snímků a zohlednit i (ne)linearity dané metriky. Také je žádoucí zohlednit množství informace na daném snímku, například čistě černé snímky budou většinou velmi podobné, pokud ne přímo shodné, ale nepřináší nám informace o kvalitě videosekvence.

Další možností porovnávání kvality videa je porovnávání videosekvencí jako celku, nikoliv pouze porovnávání jednotlivých snímků. Díky tomu je možné zachytit i přechody mezi snímky, například pomocí pohybového vektoru, a jejich kvalitu. Tyto algoritmy jsou podstatně náročnější, ale mají také přesnější výsledky. Příklad takového algoritmu je stVSSIM (detailedy o stVSSIM jsou v sekci 1.4), který budu v této práci také implementovat.

## 1.1. Metriky pro porovnávání videa

---

Obrázek 1.1: Teoretický ideální Graf kvalit jednotlivých snímků ve video



Obrázek 1.2: Graf kvalit jednotlivých snímků ve video

Tento algoritmus vychází z algoritmu SSIM, ale zohledňuje i předchozí a následující snímky při porovnávání. Taktéž zde částečně odpadá nutnost řešit vyhodnocení výsledků z jednotlivých snímků, protože tento algoritmus je již určen pro porovnávání kvality videa.

V neposlední řadě je možné kvalitu videa porovnávat na základě subjektivních sledování videosnímků lidmi a následně statistického zpracování těchto výsledků. Tato problematika však nespadá pod téma této práce, a tudíž se v této práci nebudu zabývat subjektivním porovnáváním.

## 1.2 Metrika PSNR

Peak signal-to-noise ratio (PSNR), do češtiny přeloženo jako špičkový poměr signálu k šumu[12]. Jedná se o poměr maximálního možného množství informace obsažené v signálu (v našem případě obrázku) k množství šumu, který ovlivňuje kvalitu tohoto signálu. Díky typicky velkému dynamickému rozsahu množství informace v signálu se PSNR udává v logaritmické škále a má jednotku dB.

Předpoklad pro PSNR je, že máme dva obrázky, které mají stejná rozlišení a jsou uloženy ve 2D matici, indexované od 0,0, po jednotlivých pixelech.

Poté platí matematický zápis:

$$PSNR = 10 \cdot \log_{10} \left( \frac{MAX_I^2}{MSE} \right) \quad (1.1)$$

kde platí, že *MSE* (Mean Squared Error) - průměrná čtvercová odchylka je definována jako:

$$MSE = \frac{1}{mn} \sum_{i=0}^{m-1} \sum_{j=0}^{n-1} [I(i,j) - K(i,j)]^2 \quad (1.2)$$

*m, n* reprezentují šířku a výšku daného obrázku

*I(j, k)* reprezentuje daný pixel v originálním obrázku

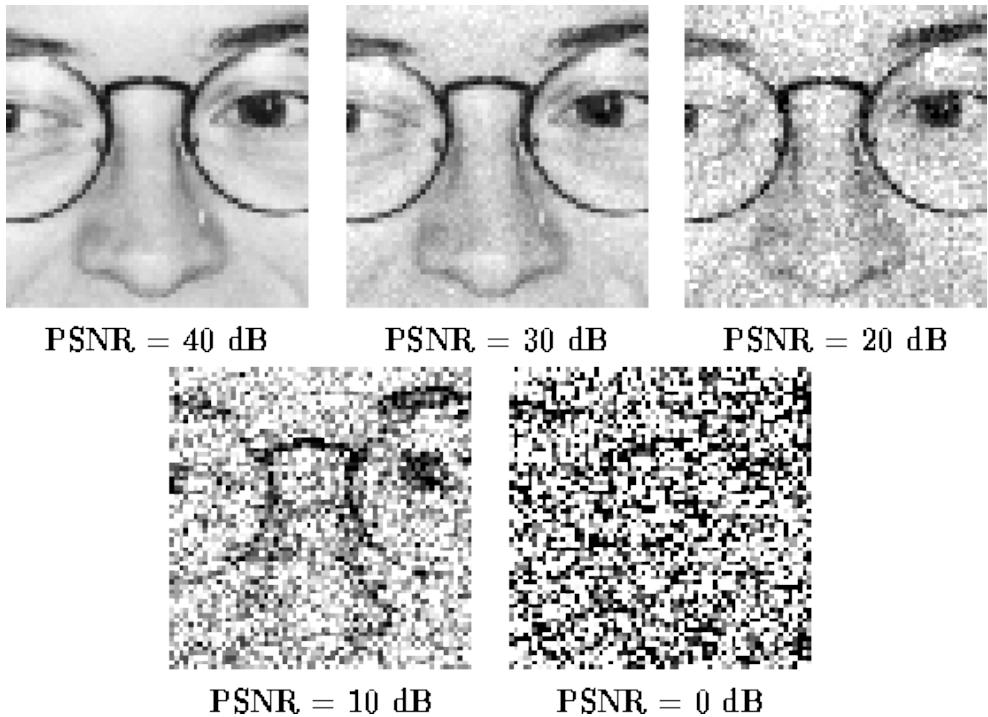
*K(j, k)* reprezentuje daný pixel v testovaném obrázku

*MAX<sub>I</sub><sup>2</sup>* reprezentuje maximální hodnotu jednoho kanálu jednoho pixelu

Pro obrázky s více než jednou barevnou složkou je možné PSNR počítat pouze na jasové části a nebo může být použit stejný vzorec s úpravou výpočtu *MSE* tak, že se postupně sečtou všechny barevné složky a následně se výsledek vydělí počtem složek[13].

Průměrná čtvercová odchylka udává odlišnost originálního a testovaného obrázku. U shodných obrázků vyjde 0, a tudíž PSNR není možné matematicky vyjádřit. Z definice také plyne, že čím nižší je MSE, tím vyšší je PSNR a tudíž je testovaný obrázek blíže k originálu. Minimální hodnota PSNR je 0dB, velmi dobrá hodnota je 50dB[14] ovšem tato hodnota závisí na obrázku a není možné takto porovnávat různé obrázky, je možné porovnávat pouze stejný obrázek

v různých kvalitách[15]. Na obrázku 1.3 je vidět ilustrace různých hodnot PSNR pro daný obrázek.



Obrázek 1.3: Srovnání obrázku v různých hodnotách PSNR

Mezi hlavní výhody algoritmu patří jeho implementační jednoduchost při uspokojivém výsledku. Hlavní slabinou tohoto algoritmu je, že je založen pouze na numerické odlišnosti jednotlivých pixelů a zcela ignoruje všechny biologické faktory vnímání obrázku lidským okem. Z tohoto důvodu jsou pro lepší přesnost doporučovány jiné algoritmy (například SSIM). Pro vylepšení přesnosti je možné počítat PSNR pouze na jasové složce z důvodu vyšší citlivosti lidského oka na jasovou složku oproti ostatním složkám.

### 1.3 Metrika SSIM

Metrika SSIM - structural similarity vyjadřuje podobnost dvou obrazů. Mezi její hlavní výhody patří zaměrní tohoto algoritmu na způsob vnímání reality lidským okem [16]. Výpočetní náročnost je přibližně stejná jako u algoritmu PSNR, stejně jako PSNR i SSIM je tzv. full reference (plně referenční) metrika (pro svojí fukčnost potřebuje mít celý referenční obraz) [16]. Dle autorů této metriky lépe koreluje se subjektivním hodnocením než metrika PSNR, toto tvrzení budu ověřovat v kap. 4. Výpočet SSIM je prováděn nad jasovou složkou

## 1. TECHNOLOGIE

---

obrazu. Základní vzorec pro výpočet SSIM je 1.3.

$$SSIM(x, y) = [l(x, y)]^\alpha \cdot [c(x, y)]^\beta \cdot [s(x, y)]^\gamma \quad (1.3)$$

Kdy musí platit, že  $\alpha > 0$ ,  $\beta > 0$  a  $\lambda > 0$ , v referenční verzi se používá  $\alpha = \beta = \gamma = 1$  [16]. Takto je výpočet SSIM rozdělen do tří částí, srovnává se postupně jas, kontrast a struktura. Tyto tři veličiny jsou na sobě nezávislé. Nejprve je spočítán jas. Pro průměrnou hodnotu jasu platí vzorec 1.4

$$\mu_x = \frac{1}{N} \sum_{i=1}^N x_i \quad (1.4)$$

Pro rozdíl mezi jasovými složkami dvou snímků je použit vzorec 1.5.

$$l(x, y) = \frac{2\mu_x\mu_y + C_1}{\mu_x^2 + \mu_y^2 + C_1} \quad (1.5)$$

Následně je odečtena průměrná jasová složka z obrazu a za pomocí vzorce 1.6 je vypočítána standardní odchylka jednotlivých snímků.

$$\sigma_x = \left( \frac{1}{N-1} \sum_{i=1}^N (x_i - \mu_x)^2 \right)^{1/2} \quad (1.6)$$

Vzorec 1.7 vyjadřuje rozdíl kontrastu dvou snímků.

$$c(x, y) = \frac{2\sigma_x\sigma_y + C_2}{\sigma_x^2 + \sigma_y^2 + C_2} \quad (1.7)$$

Následně je signál (snímek) znárován (vydělen) jeho vlastní standardní odchylkou. Díky tomu mají oba snímky stejnou standardní odchylku. Porovnávání struktury signálů je tedy prováděna na znárovovaném signálu  $(x - \mu_x)/\sigma_x$ . Korelace těchto signálů je vyjádřena pomocí vzorce 1.8

$$s(x, y) = \frac{\sigma_{xy} + C_3}{\sigma_x\sigma_y + C_3} \quad (1.8)$$

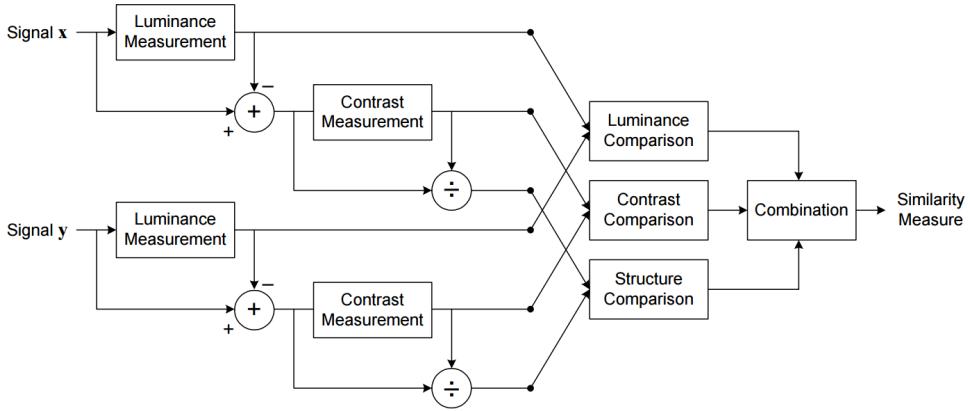
Z toho  $\sigma_{xy}$  se vypočítá dle 1.9.

$$\sigma_{xy} = \frac{1}{N-1} \sum_{i=1}^N (x_i - \mu_x)(y_i - \mu_y) \quad (1.9)$$

Kde  $\mu_x$  and  $\mu_y$  představují průměr z originálního a testovaného obrázku.  $\sigma_x$  a  $\sigma_y$  představují standardní odchylku.  $\sigma_x^2$  a  $\sigma_y^2$  představují varianci.  $\sigma_{xy}$  představuje kovariance těchto dvou obrázků. Na obrázku 1.4 je možné vidět jednotlivé kroky procesu tvorby SSIM indexu.

Při použití hodnot  $\alpha = \beta = \gamma = 1$  a  $C_3 = C_2/2$  [16] vyjde zjednodušený vzorec 1.10.

$$SSIM(x, y) = \frac{(2\mu_x\mu_y + C_1)(2\sigma_{xy} + C_2)}{(\mu_x^2 + \mu_y^2 + C_1)(\sigma_x^2 + \sigma_y^2 + C_2)} \quad (1.10)$$



Obrázek 1.4: Diagram procesu tvorby SSIM

$C_1, C_2$  a  $C_3$  jsou použity pro eliminaci nestability výsledků, když některá z hodnot  $\mu_x^2 + \mu_y^2$  nebo  $\sigma_x^2 + \sigma_y^2$  bude blízká k nule. Pro zjednodušení je možné uvažovat, že  $C_1 = C_2 = C_3 = 0$ . Po tomto zjednodušení by vzorec korespondoval s  $UQI$ (universal quality index). V SSIM metrice jsou  $C_1, C_2$  definovány jako  $C_1 = (K_1 L)^2$  respektive  $C_2 = (K_2 L)^2$ , kde  $L$  je dynamický rozsah hodnoty pixelu (255 pro jasovou složku 8 bit obrázku) a  $K$  je konstanta výrazně menší než 1. Běžně se používají hodnoty  $K_1 = 0.01$  a  $K_2 = 0.03$  [17].

Hodnoty  $\mu_x$ ,  $\sigma_x$  a  $\sigma_{xy}$  se počítají na blocích  $8 \times 8$  pixelů, toto okno se postupně posouvá po celém obrazu, je možné posouvat po jednom nebo více pixelech pro zvýšení rychlosti výpočtu za cenu snížení přesnosti výpočtu. Pro vypočítání SSIM pro celý snímek se poté použije průměr z výsledků SSIM pro  $8 \times 8$  okna. Pro zvýšení přesnosti tohoto algoritmu je možné použít vážený průměr nebo heuristiku zohledňující bloky s horším výsledkem[18].

Metrika SSIM má oproti metrice PSNR výhodu v tom, že zohledňuje vnímání obrazu lidským okem a nepočítá pouze absolutní odchylku od výchozího zdroje, například zohledňuje citlivost lidského oka na jasovou složku obrázku. V metrice SSIM je také využito maskování postupně průměrného jasu a kontrastu pro získání nezávislých výsledků při počítání kontrastu a struktury. Metrika SSIM se opírá o myšlenku, že pixely, zejména ty které jsou blízko sebe, mají silnou závislost. Tato závislost obsahuje důležité množství informace ohledně struktury objektů v obrázku. Je ovšem výpočetně náročnější oproti PSNR. Stejně jako PSNR je určena pro porovnávání snímků a nikoliv videí, tudíž pro mé využití je potřeba tento algoritmus rozšířit, aby byl použitelný i na video.

## 1.4 Metrika stVSSIM

Metrika spatio-temporal video SSIM[18] je další z algoritmů na porovnávání kvality videí. Narození od algoritmů SSIM a PSNR, které porovnávají pouze jednotlivé snímky mezi sebou, stVSSIM algoritmus zohledňuje i okolní snímky videa. Jedná se o plně referenční metriku, stejně jako PSNR a SSIM. Inspirací pro stVSSIM byl Motion-based Video Integrity Evaluation(MOVIE) index[19]. MOVIE index se sice ukázal být poměrně přesný, ale výpočetně velmi náročný. Proto byl navržen spatio-temporal video SSIM algoritmus.

Spatio-temporal video SSIM využívá pohybové informace získané z blokově založeného algoritmu na odhad pohybu, při použití sady časoprostorových filtrů. Algoritmus stVSSIM se snaží o zachování kvality algoritmu MOVIE při snížení potřebného výpočetního výkonu.

Algoritmus stVSSIM využívá metriku SSIM popsanou v předchozí kapitole na měření kvality jednotlivých snímků. Tato metrika velmi dobře koreluje se subjektivním pozorováním při porovnávání kvality snímků[20]. Vyhodnocování kvality v časové rovině je dosaženo rozšířením SSIM o časoprostorovou doménu, kterou autoři této metriky nazývají SSIM-3D.

### 1.4.1 Algoritmus SSIM-3D

SSIM je počítána z jednoho snímku po blocích dané velikosti(typicky  $8 \times 8$  pixelů), při SSIM-3D je přidána ještě časová osa.

Vezměme si pixel na souřadnicích  $(i, j, k)$ , kde  $(i, j)$  jsou prostorové souřadnice daného pixelu v rámci daného obrazu a  $k$  značí číslo snímku. V tomto případě nás při každém výpočtu zajímá blok pixelů v okolí pixelu  $(i, j, k)$ . Definujme  $\alpha, \beta$  jako rozměry v prostoru a  $\gamma$  jako počet snímků. V případě real-time implementace tohoto algoritmu by bylo využito  $\gamma - 1$  předchozích snímků a současný. V této práci ale nepotřebujeme vyhodnocovat kvalitu v real-time, a proto budou brány i následující snímky. Pro každý vyhodnocovaný snímek tedy budou brány snímky  $k - \gamma/2$  až  $k + \gamma/2$ . Pro pixel  $(i, j, k)$  je tedy možné určit blok jako všechny pixely mezi:

$$\begin{aligned}
& (i - \left\lfloor \frac{\alpha}{2} \right\rfloor, j - \left\lfloor \frac{\beta}{2} \right\rfloor, k - \left\lfloor \frac{\gamma}{2} \right\rfloor) \\
& (i - \left\lfloor \frac{\alpha}{2} \right\rfloor, j + \left\lfloor \frac{\beta}{2} \right\rfloor, k - \left\lfloor \frac{\gamma}{2} \right\rfloor) \\
& (i + \left\lfloor \frac{\alpha}{2} \right\rfloor, j - \left\lfloor \frac{\beta}{2} \right\rfloor, k - \left\lfloor \frac{\gamma}{2} \right\rfloor) \\
& (i + \left\lfloor \frac{\alpha}{2} \right\rfloor, j + \left\lfloor \frac{\beta}{2} \right\rfloor, k - \left\lfloor \frac{\gamma}{2} \right\rfloor) \\
& \text{a} \\
& (i - \left\lfloor \frac{\alpha}{2} \right\rfloor, j - \left\lfloor \frac{\beta}{2} \right\rfloor, k + \left\lfloor \frac{\gamma}{2} \right\rfloor)
\end{aligned}$$

$$(i - \left\lfloor \frac{\alpha}{2} \right\rfloor, j + \left\lfloor \frac{\beta}{2} \right\rfloor, k + \left\lfloor \frac{\gamma}{2} \right\rfloor)$$

$$(i + \left\lfloor \frac{\alpha}{2} \right\rfloor, j - \left\lfloor \frac{\beta}{2} \right\rfloor, k + \left\lfloor \frac{\gamma}{2} \right\rfloor)$$

$$(i + \left\lfloor \frac{\alpha}{2} \right\rfloor, j + \left\lfloor \frac{\beta}{2} \right\rfloor, k + \left\lfloor \frac{\gamma}{2} \right\rfloor)$$

Referenční hodnoty jsou  $\alpha = \beta = 11$  a  $\gamma = 33$ . Tyto hodnoty vychází z algoritmu SSIM a MOVIE, kterým je metrika stVSSIM inspirována. Obdobně jako u metriky SSIM jsou použity následující vzorce.

$$\mu_{x(i,j,k)} = \sum_{m=1}^{\alpha} \sum_{n=1}^{\beta} \sum_{o=1}^{\gamma} w(m, n, o) x(m, n, o) \quad (1.11)$$

$$\mu_{y(i,j,k)} = \sum_{m=1}^{\alpha} \sum_{n=1}^{\beta} \sum_{o=1}^{\gamma} w(m, n, o) y(m, n, o) \quad (1.12)$$

$$\sigma_{x(i,j,k)}^2 = \sum_{m=1}^{\alpha} \sum_{n=1}^{\beta} \sum_{o=1}^{\gamma} w(m, n, o) (x(m, n, o) - \mu_{x(i,j,k)})^2 \quad (1.13)$$

$$\sigma_{y(i,j,k)}^2 = \sum_{m=1}^{\alpha} \sum_{n=1}^{\beta} \sum_{o=1}^{\gamma} w(m, n, o) (y(m, n, o) - \mu_{y(i,j,k)})^2 \quad (1.14)$$

$$\sigma_{x(i,j,k)y(i,j,k)} = \sum_{m=1}^{\alpha} \sum_{n=1}^{\beta} \sum_{o=1}^{\gamma} w(m, n, o) (x(m, n, o) - \mu_{x(i,j,k)}) (y(m, n, o) - \mu_{y(i,j,k)}) \quad (1.15)$$

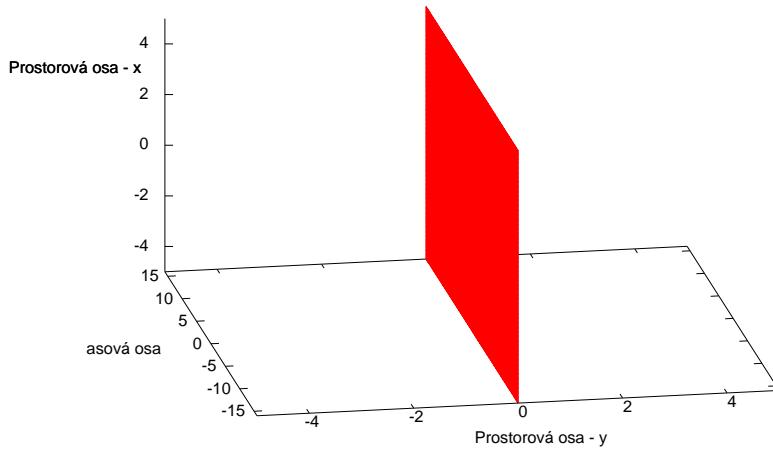
Výpočet celého SSIM-3D pro pixel  $(i, j, k)$  je zobrazen v rovnici 1.16.

$$\text{SSIM-3D} = \frac{(2\mu_{x(i,j,k)}\mu_{y(i,j,k)} + C1)(2\sigma_{x(i,j,k)y(i,j,k)} + C2)}{(\mu_{x(i,j,k)}^2 + \mu_{y(i,j,k)}^2 + C1)(\sigma_{x(i,j,k)}^2 + \sigma_{y(i,j,k)}^2 + C2)} \quad (1.16)$$

Konstanty  $C1$  a  $C2$  jsou stejné jako v případě metriky SSIM. Váhová funkce  $w$  záleží na typu filtru, který je použit. V případě algoritmu SSIM-3D jsou použity 4 různé filtry, viz obrázky 1.5, 1.6, 1.8, 1.7. Váhová funkce  $w(m, n, o)$  nabývá hodnoty 1, pokud bod  $(m, n, o)$  leží v rovině daného filtru, v opačném případě nabývá hodnoty 0. Souřadnice  $m$  a  $n$  značí horizontální a vertikální pozici od výchozího pixelu z bloku pixelů. Záporné hodnoty značí pixely vlevo od tohoto pixelu a naopak.

Souřadnice  $o$  značí číslo snímku od výchozího snímku. Záporné hodnoty  $o$  značí předchozí snímky a naopak. V referenční publikaci jsou spočítány SSIM-3D hodnoty pro všechny 4 filtry pro každý  $(i, j, k)$  pixel a následně je jich při výpočtu vybrána pouze část. Který filtr bude použit se rozhoduje podle odhadu pohybového vektoru. Tento odhad pohybového vektoru je vypočítán

z aktuálního a předchozího snímku při použití bloku  $8 \times 8$ . Použití bloku  $8 \times 8$  je zapříčiněno velmi četným použitím právě tohoto bloku při kompresích videa. Moderní kompresní algoritmy ale používají proměnnou velikost bloku a z toho důvodu je vhodné zkoušet změnit velikost bloku. Pro výpočet pohybového vektoru je použit algoritmus Adaptive Rood Pattern Search (ARPS)[21].



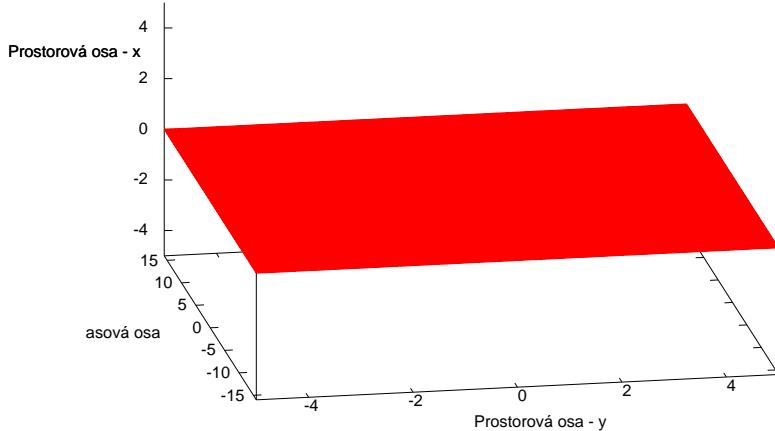
Obrázek 1.5: Vertikální filtr

### 1.4.2 Algoritmus ARPS

#### 1.4.2.1 Popis algoritmu

Adaptive Rood Pattern Search (ARPS)[21] je algoritmus, který se zabývá hledáním pohybových vektorů ve videu. Pohybový vektor (motion vector) je způsob reprezentace bloku v daném snímku na základě stejného (nebo podobného) bloku v předchozím snímku. Pro nalezení pohybového vektoru je potřeba nalézt v předchozím snímku blok co nejpodobnější aktuálnímu bloku. Při videokomprese se často používají pohybové vektory, protože následující snímek je často alespoň částečně tvořen pouze posunutím předchozího snímku, nebo jeho části. Tohoto algoritmu je využíváno zejména při komprese videí. Hledání pohybových vektorů je časově nejnáročnější akcí při komprese videa, a proto může tvořit velké omezení při komprese/enkódování videa. Algoritmus ARPS se snaží přinést dostatečně efektivní způsob, který bude ale také mnohem méně výpočetně náročný než například celé prohledávání hrubou silou.

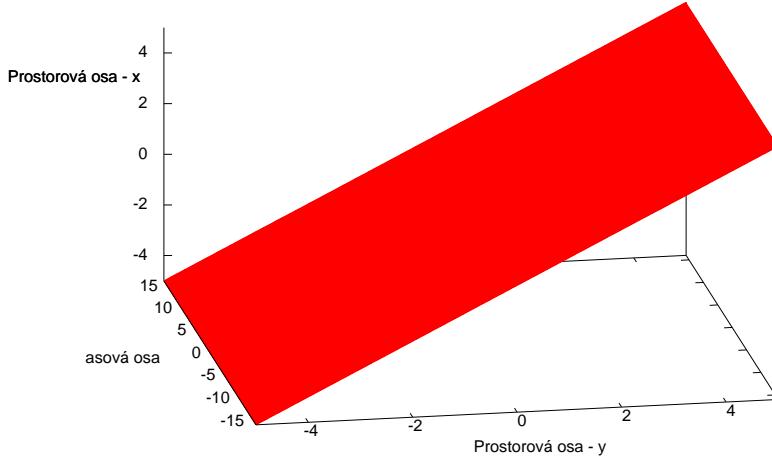
Jednou z důležitých vlastností tohoto algoritmu je předpoklad, že okolní bloky mají podobný pohybový vektor. Pohybový vektor získaný z okolních



Obrázek 1.6: Horizontální filtr

bloků budu nazývat předpovídáný pohybový vektor. Okolní bloky mohou být buď časové, nebo prostorové, intuitivně by blok na stejné prostorové pozici v předchozím snímku mohl mít podobný pohybový vektor, ale z důvodu vysoké paměťové náročnosti tohoto řešení bylo v referenčním řešení zvoleno využití sousedních prostorových bloků. Protože každý snímek je zpracováván postupně po blocích v jednotlivých řadách, jsou k dispozici pohybové vektory okolních bloků viz obrázek 1.9. Na uvedeném obrázku je červeně znázorněn právě ověřovaný blok a šedě bloky, které jsou použity pro odhad okolních bloků. Typ B je často používán při video kompresi, například i u populárního formátu H263[22]. Předpovídáný pohybový vektor se spočítá jako medián z okolních pohybových vektorů. V případě bloku na okraji snímků jsou použity bloky jako na obrázku 1.10. Místo mediánu je možné použít průměr nebo jiné složitější metody. Ty ale vzhledem k zanedbatelnému přínosu v přesnosnosti a rychlosti nemá smysl používat. Dle [21] je velmi malý rozdíl v přesnosti a výkonu při používání různých podporujících sousedních bloků, z toho důvodu v implementaci použije typ D.

Algoritmus ARPS se snaží najít vzor aktuálního bloku dle kříže, jak je znázorněno na obrázku 1.11. Na tomto obrázku je vidět jak využití pohybového vektoru z předchozího snímku, který byl  $(2, -1)$ , tak využití 4 okolních bodů a středu. Vzdálenost okolních bodů od středu ( $\Gamma$ ) může být dána například rovnicí 1.17. Alternativně je možné použít rovnici 1.18. Protože druhá zmíněná rovnice nevyžaduje ani zaokrouhllování, ani umocňování a odmocňování, je mnohem méně výpočetně náročná. Protože při mé implementaci nepřinášela



Obrázek 1.7: Pravý filtr

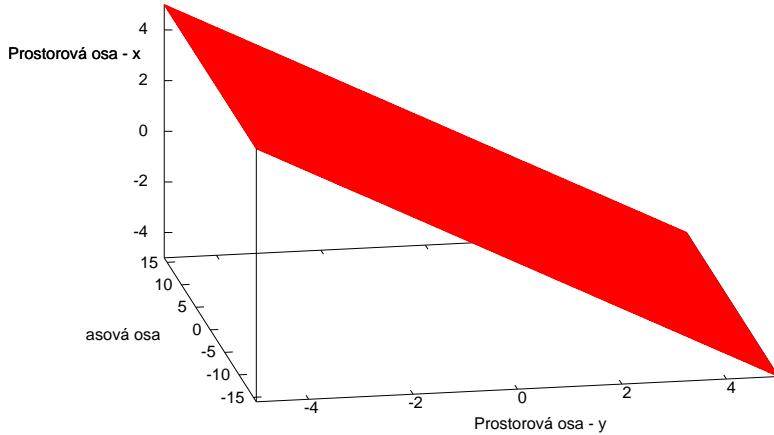
první rovnice jiné výsledky, budu používat rovnici 1.18. Pokud není dostupný předchozí blok, bude použita vzdálenost 2 pixely, tedy  $\Gamma = 2$ .

$$\Gamma = \text{Round} \left| \overrightarrow{MV}_{predicted} \right| = \text{Round} \left| \sqrt{MV_{predicted}(x)^2 + MV_{predicted}(y)^2} \right| \quad (1.17)$$

$$\Gamma = \text{Max}(|MV_{predicted}(x)|, |MV_{predicted}(y)|) \quad (1.18)$$

Pro zjištění shodnosti daného bloku se vzorem je použita suma absolutních rozdílů pixelů mezi vzorem a daným blokem.

Další vlastností algoritmu ARPS je takzvaná premisa nulového pohybu (zero-motion prejudgment). Tento předsudek očekává, že ve většině bloků nedochází k žádnému pohybu. Jako klasický příklad může být uvedena telekonference, kde po většinu doby nedochází k žádnému pohybu, nebo jen ve velmi omezené části obrazu. Algoritmus tedy v prvním kroku spočítá sumu absolutních rozdílů oproti stejně umístěnému bloku v předchozích snímcích, a pokud je tato suma nižší než stanovená hranice, tak již výpočet nepokračuje a je bráno, že zde nedošlo k žádnému pohybu. Tato hranice je v referenční verzi algoritmu stanovena na odchylku 2 na každém pixelu, tedy při velikosti  $8 \times 8$  je možné stanovit hranici na  $T = 128$ . Tuto hranici je možné posunout, čím výše je tato hranice nasazena, tím lepší dosáhneme zrychlení tohoto algoritmu, ale ztrácíme přesnost výpočtu.



Obrázek 1.8: Vertikální filtr



Obrázek 1.9: Typy podporujících sousedních bloků

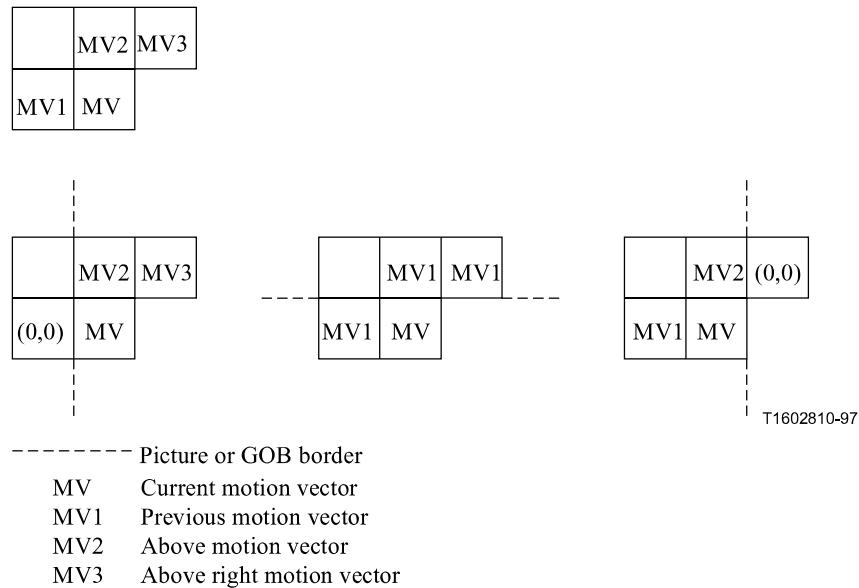
#### 1.4.2.2 Výpočet ARPS

Jak bylo již zmíněno dříve, budu používat D typ podporujícího sousedního bloku dle obrázku 1.9. Pro všechny nelevější bloky použijeme  $\Gamma = 2$ . Hranice pro nulový pohyb bude v našem případě  $T = 128$ .

1. Spočítáme sumu absolutních rozdílů oproti stejně umístěnému bloku v předchozím snímku. Pokud je tato suma menší nebo rovna 128, pak ukončíme algoritmus a vrátíme pohybový vektor o hodnotě  $(0, 0)$  předpokládáme tedy nulový posuv.
2. Pokud je aktuální blok nelevější v daném snímku, nastavíme  $\Gamma = 2$ , v opačném případě nastavíme  $\Gamma = \text{Max}(|MV_{predicted}(x)|, |MV_{predicted}(x)|)$

## 1. TECHNOLOGIE

---



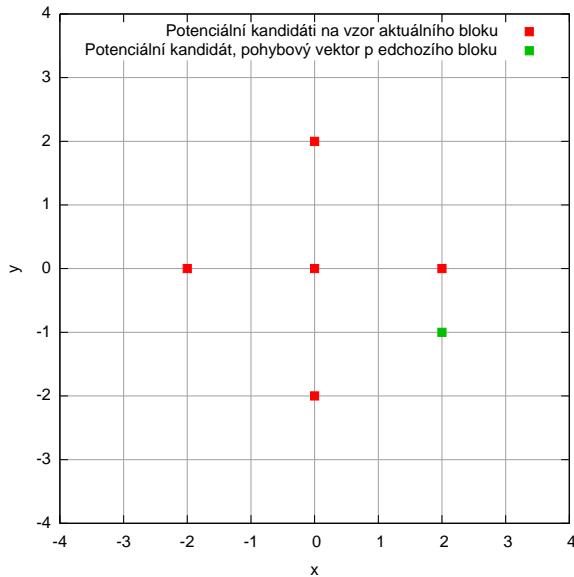
Obrázek 1.10: Podporující sousední bloky u okrajů

3. Nastavíme střed kříže, který je znázorněn na obrázku 1.11, na stejné souřadnice, jako je aktuální blok.
4. Spočítáme sumy absolutních rozdílů pro bloky se středy danými těmito body. Pokud vyjde střed jako bod s nejnižší sumou rozdílů, vracíme rozdíl souřadnic, který reprezentuje pohybový vektor.
5. V opačném případě nastavíme střed kříže na bod s nejnižším absolutním rozdílem a opakujeme bod 3.

Pro snížení výpočetní náročnosti je vhodné implementovat nějakým způsobem identifikování, zdali už byla spočítána odchylka daného bodu, například pomocí bitmapy. Takto se postupně dostaneme do bodu, který budu používat jako vzor pro aktuální blok.

### 1.4.3 Využití ARPS

Poté, co je spočítán pohybový vektor daného bloku, je potřeba rozhodnout, který filtr bude použit. Je možné použít vážené průměry těchto filtrov dle směrového vektoru, pro snížení výpočetní náročnosti je ale také možné použít hladový algoritmus a zvolit filtr, který je nejbližší vzniklému pohybovému vektoru. Tím se vyhneme například výpočtům s desetinnými čísly. Pokud je vzniklý pohybový vektor přesně mezi dvěma filtrov, je použit průměr z těchto



Obrázek 1.11: Graf potenciálních kandidátů na vzor aktuálního bloku

dvou filtrů. Pokud je pohybový vektor roven  $(0, 0)$ , pak použijeme průměr ze všech čtyř filtrů. V nejhorším případě bude výpočetně nejnáročnější operací dělení čtyřmi, což lze ale implementovat pomocí bitového posunu, proto lze označit tuto část výpočtu za výpočetně nenáročnou.

#### 1.4.4 Výpočet stVSSIM

Na každý pixel z daného snímku aplikujeme algoritmus SSIM-3D. V referenční verzi je použit nejhorších 6 % z výsledků SSIM-3D pro daný snímek. Je ale také možné použít průměr všech výsledků SSIM-3D. Výsledek pro časovou doménu celého video snímku ( $T_{video}$ ) je spočítán jako průměr výsledků jednotlivých snímků, stejně jako je výsledek pro prostorovou rovinu ( $S_{video}$ ) spočítán jako průměr výsledků SSIM pro jednotlivé snímky. V referenční implementaci [18] se počítá SSIM i SSIM-3D pouze z každého šestnáctého snímku. Výsledný index stVSSIM je následně spočítán jako 1.19.

$$stVSSIM = T_{video} \times S_{video} \quad (1.19)$$

## 1.5 Zpracování a uložení videa

Dále se v této práci budu často zabývat tématem souvisejícím s kodeky pro kódování videa, jeho kompresi a formáty uložení audia a videa. Velmi stručně

## 1. TECHNOLOGIE

---

v této sekci popíšu, co to vlastně je video kodek a video formát a také velmi stručně popíší video formáty používané v této práci.

Kodek je program nebo obecně proces, který dokáže transformovat vstupní proud dat. Následně dokáže ukládat tento transformovaný proud dat v zakódované podobě. Stejně tak je ale možné využít kodek pro dekódování takovéto sekvence zpět na tok dat co nejbližší původnímu toku dat. Video formát je specifikace kroků, které je potřeba vykonat na převod vstupního proudu dat na zakódovaný proud dat. Rozdíl mezi kodekem a formátem je, že formát je pouze obecná specifikace nějaké transformace, zatímco kodek je již konkrétní implementace této transformace. Využití kodeků je zejména pro kompresi multimediálních dat, jejich přenos po síti a případně i šifrování. Komprese může být buď ztrátová a nebo bezztrátová, ve video průmyslu se nejčastěji používají ztrátové kodeky, příkladem ztrátového kodeku je například h264 nebo VP9. Mezi známé bezztrátové kodeky patří například FLAC pro audio. Typický obsah video souboru je hlavička, která určuje, jaký kodek je použit a další informace o souboru a samotný audio a video kanál, zakódovaný pomocí určeného kodeku.

### 1.5.1 Formát rawvideo

Speciální kategorii tvoří formáty uložení audio/video streamu bez komprese a bez šifrování, nazývané též rawvideo[3]. Tyto kodeky mohou mít různé formáty uložení dat, mezi tyto formáty patří například yuv420p (tentotéž název je používán v programu [4], v jiné literatuře je možné se setkat s jiným názvem, například I420[5]). YUV značí použití barevného modelu s jasovou složkou (Y) a dvěma barevnými složkami. Jasová složka označuje množství jasu (pro více detailů viz [6]) na daném pixelu. Číslo 420 značí poměr počtu bytů vyhrazených pro Y kanál oproti ostatním složkám[5]. Písmeno p na konci názvu tohoto formátu značí planární formát, tedy že jednotlivé složky jsou uloženy postupně pro celý snímek, nejprve je tedy uložena celá Y složka, poté U a poté V[5]. Mezi další formáty uložení takovýchto nekomprimovaných dat patří rgb24, jak již napovídá název, tento formát je založen na RGB barevném prostoru a na každý pixel jsou použity 3 bajty (24 bitů).

### 1.5.2 Formát H.264

H.264, často nazývaný také jako MPEG-4 AVC [7], je jeden z dnes nejpoužívanějších video formátů. Byl vyvinut pro HD (High-definition) videa, cílem návrhu tohoto formátu bylo vytvoření video formátu, který bude mít dobrou kvalitu videa při nižším datovém toku než jeho předchůdci (MPEG-2, MPEG-4 apod.). Je využíván na Blu-ray discích, stejně, jako je využíván pro DVB-T HD televizní vysílání, dále je také často využíván u satelitního a kabelového televizního signálu. První specifikace tohoto formátu byla vydána v roce 2003.

Nyní je velmi rozšířen, pravděpodobně je stále nejrozšířenějším formátem HD videa na trhu.

### 1.5.3 Formát H.265

Formát H.265, označován také jako HEVC (High Efficiency Video Coding), je nástupce H.264, zaměřuje se na HD videa a to až do kvality 8K ( $8192 \times 4320$ ), zaměřuje se také na co největší možnost paralelizace zpracování tohoto formátu [8]. Jeho plánované využití je v rámci standardu pozemního vysílání DVB-T2 [9]. Dle autorů tohoto formátu je schopen poskytnout srovnatelnou kvalitu videa jako jeho předchůdce při snížení datového toku o 50 %[8]. Vzrostla však výpočetní složitost kodeků pro tento formát. Standardizován byl v roce 2013 [8]. Na tento formát je několik patentů a za využití tohoto formátu je potřeba platit licenční poplatky.

### 1.5.4 Formát VP9

VP9 je formát vyvíjený firmou Google. Jedná se o konkurenční formát k formátu H.265. Typicky je uchováván ve webm kontejneru (soubor s příponou .webm) a je velmi často používán pro videa na webu, například <https://youtube.com> využívá tento formát u svých videí. Většina (téměř 77 %) používaných prohlížečů podporuje a využívá webm kontejner a tudíž i VP9 formát. [10] Podporuje i bezzávodovou variantu nicméně častěji se používá klasická ztrátová varianta tohoto formátu. Stejně jako H.265 je určen pro videa s rozlišením alespoň full HD ( $1920 \times 1080px$  nebo obdobné).

### 1.5.5 Constant Rate Factor

CRF - Constant Rate Factor není formát ani kodek, ale velmi často se v této souvislosti používá. Pokud je potřeba konvertovat nějaké video, je potřeba určit, jakou má mít výsledné video kvalitu. Tato kvalita je možná určit několika různými parametry (například datovým tokem nebo velikostí souboru. U některých programů (například ffmpeg) na video konverzi je možné nastavit parametr crf, který nastaví kvalitu výstupního videa na nějakou hladinu. Při této metodě je hlavním kritériem udržení dané kvality v průběhu celého videa. Velikost souboru a aktuální datový tok jsou méně důležité. Tato metoda zaručuje, že každý snímek dostane potřebný datový tok pro zachování určené kvality. Nevýhodou je však nemožnost určit velikost výsledného souboru.

Rozsah CRF hodnoty je od 0 do 51, kdy 0 znamená bezzávodovou kompresi, 23 je výchozí hodnota a 51 je nejhorší možná hodnota. Rozumné hodnoty se udávají od 18 do 28[11]. Kvalita 18 by měla být vizuálně bezzávodová, nebo alespoň téměř bezzávodová, měla by tedy vypadat velmi podobně jako originál.

Rozsah hodnot má exponenciální měřítko, tudíž zvýšení hodnoty CRF o šest zhruba znamená snížení datového toku videa na polovinu.



# KAPITOLA **2**

---

## **Analýza**

V této kapitole popíšu vlastnosti a možnosti nástrojů, které budu následně využívat při implementaci. Zaměřím se také na srovnání s již existujícím produktem.

### **2.1 OpenMP**

OpenMP (Open Multi-Processing) je rozhraní pro programování aplikací (API), které podporuje víceplatformní, víceprocesorové výpočty nad sdílenou pamětí [23]. Podporuje programovací jazyky C, C++ a Fortran. Funguje na většině platform, na různých architekturách procesoru a operačních systémech včetně systémů Linux a Windows. Sestává ze sady direktiv, procedur a proměnných prostředí, které ovlivňují běh programu. OpenMP je spravována neziskovým konsorcium OpenMP Architecture Review Board (OpenMP ARB), které bylo založeno skupinou velkých firem z oblasti IT včetně firem jako AMD, IBM, Intel, HP a dalších.

Toto API budu využívat pro jendnoduchou paralelizaci jednovlákновé aplikace. OpenMP ale stále využívá pouze CPU, pokud chceme pro naše výpočty použít i GPU a tím potenciálně zvýšit výkon aplikace, není možné využít OpenMP. Pro výpočty akcelerované pomocí grafické karty se běžně používají platformy CUDA nebo OpenCL, z hlediska funkčnosti jsou si velmi podobné, z důvodů mých větších zkušeností s platformou CUDA jsem se rozhodl se v této práci zabývat CUDA.

### **2.2 CUDA**

CUDA - Compute Unified Device Architecture je paralelní výpočetní platforma a programovací model vynalezený firmou NVIDIA [24]. Umožňuje výrazný nárůst výpočetního výkonu díky použití výpočetního výkonu grafického čipu (GPU).

## 2. ANALÝZA

---

CUDA přidává možnost do programovacích jazyků C, C++ a Fortran přidat kód, který bude spuštěn na grafické kartě, tím velmi výrazně zvýšit rychlosť daného algoritmu díky vysoké míře paralelizace. NVIDIA na svých stránkách uveřejňuje všechny nástroje pro práci s CUDA, stejně jako příručky a ukázkové zdrojové kódy. Pro spuštění CUDA kódu je potřebná podporovaná grafická karta od firmy NVIDIA. NVIDIA postupně zlepšuje výpočetní možnosti zařízení (compute capability), programy mohou vyžadovat určitou verzi výpočetních možností (například compute capability 2.0) pro svůj běh, každá podporovaná grafická karta má danou verzi výpočetních schopností, není možné na této grafické kartě nijak aktualizovat tuto verzi na novější. Výpočetní schopnosti jsou ale zpětně kompatibilní, například na verzi 6.0 půjde bez omezení spustit veškerý software, který je psán pro verzi 2.0.

### 2.3 C++

Jako programovací jazyk pro celou tuto práci jsem zvolil C++ a to především kvůli jeho podpoře CUDA architektury a OpenMP API. Jedná se o kompilovaný jazyk který je zpětně kompatibilní s jeho předchůdcem C. Tento jazyk je procedurální a podporuje objektově orientované programování. C++ byl standardizován v roce 1998, i přesto že již předtím byl hojně používán a považován za de facto standard[25]. Mezi jeho výhody patří velká přenositelnost na různé platformy a také jeho rychlosť. Mezi nevýhody může patřit vyšší náročnost na implementaci oproti jiným, zejména interpretovaným jazykům.

### 2.4 Srovnání s konkurencí

Velmi rozsáhlým projektem, který se tímto tématem zabývá je ruský projekt „Everything about the data compression“ (Vše o kompresi dat) [26]. Jedna z významných částí tohoto projektu se zabývá video kompresí. Tato odnož projektu je velmi rozsáhlá a pravidelně zveřejňuje například srovnání různých kodeků, mimo jiné i srovnání HEVC kodeku [27]. Hlavní nevýhodou tohoto projektu je, že jejich nové analýzy a reporty jsou zpoplatněny a to částečnou okolo 900\$. Zdarma je k dispozici pouze omezená část reportu, typicky bez některých srovnání a grafů a nebo bez testovacích videí. Jejich starší reporty jsou pak uveřejňovány zdarma v celém rozsahu. Zároveň také tato skupina lidí vytvořila nástroje na objektivní i subjektivní porovnávání kvality videa. Jejich nástroj na objektivní měření kvality videa (MSU Quality Measurement Tool) obsahuje velké množství metrik ve kterých porovnává videa (mimo jiné i PSNR a SSIM) ale například metriku stVSSIM neobsahuje. Dále také podporuje výpočet pomocí grafické karty (openCL i CUDA). Hlavní nevýhodou tohoto programu je, že jeho plná verze stojí více než 1000 EUR, verze zdarma je omezena například tím, že neumí pracovat s HD videi a neumí srovnávat

## 2.4. Srovnání s konkurencí

---

více než 2 odvozená videa najednou, z těchto důvodů nemohu použít tento program pro měření na mnou zamýšlených videích.

Dalším projektem, který se zabývá tímto tématem je projekt švýcarské univerzity École Polytechnique Fédérale de Lausanne. Tento projekt se nazývá VQMT - Video Quality Measurement Tool[28], vyhodnocuje kvalitu videa na základě několika různých metrik včetně SSIM a PSNR ale metriku stVSSIM také neimplementuje. Tento projekt vyžaduje OpenCV knihovnu [29] a využívá implementace v matlabu. Narozdíl od mého programu nebo od výše zmíněného projektu přijímá jako vstupní soubory nekomprimované video vstupy, výhody a nevýhody tohoto přístupu jsou popsány v sekci 3.1. Neumí také porovnat více než jeden referenční a jeden odvozený soubor najednou.



# KAPITOLA 3

---

## Implementace

V této kapitole popíšu samotnou implementaci algoritmu. Nejprve se zaměřím na obecná specifika algoritmu, jako například jeho požadavky a prerekvizity, dále popíši jeho nároky na hardware a budu pokračovat popisem jednotlivých částí tohoto programu.

Program pro výpočet všech metrik je psán v programovacím jazyce C++. Vzniklý kód je funkční na platformách Windows a Linux. Mezi jeho nutné požadavky patří nainstalovaný program ffmpeg a ffprobe[4] a nebo alespoň existence zkompilované verze těchto souborů. Tento program je volně ke stažení a je dostupný v binární podobě pro Windows [30] i Linux ve formě balíčku [31]. Ffprobe je využíván ke zjištění úvodních informací o videu, například pro zjištění výšky a šířky videa, dále ke zjištění počtu snímků ve videu. U některých videí ale není uvedena informace o počtu snímků, proto je nutné ji dopočítat, setkal jsem se i s videi, která mají v hlavičce uvedenou nesprávnou délku videa a proto není možné využít výpočet ze snímkové frekvence a z délky videa, je nutné použít parametr count\_frames, který vynutí dekódování celého videa z důvodu přepočítání reálného počtu snímků. Kvůli zjednodušení parsování je hlavička výstupu parametry omezena na minimum. Tento příkaz je časově náročný, ale výrazně nižší oproti času potřebného na vyhodnocení videa.

Ffmpeg je použit pro samotné načítání videa. Díky tomuto nástroji je možné jako vstup pro můj program použít širokou škálu formátů videa. Program ffmpeg kromě standardních kodeků jako h264, mpeg4, h265 zvládá i mnoho jiných, včetně nekomprimovaných bezztrátových (např. YUV420p). Právě poslední zmínovaný kodek v tomto programu využívám jako výstupní kodek. V dřívější implementaci jsem využíval nekomprimovaný RGB (rgb24) výstup. Ten ale měl několik nevýhod. Díky největší citlivosti oka na jasovou složku obrazu jsou mnou využívané metriky většinou počítány pouze nad jasovou složkou, z toho důvodu byla nutná konverze z barevného modelu RGB na barevný model YUV. Dále také samotná videa jsou typicky ukládána za využití YUV, proto při použití RGB modelu docházelo zbytečně ke dvojí konverzi navíc.

### 3. IMPLEMENTACE

---

Vytvořený program využívá platformu CUDA (tato platforma je blíže popsána v sekci 2.2), pro spuštění GPU části tohoto programu je tedy potřeba kompatibilní grafická karta značky NVIDIA s výpočetní schopností 2.0 a vyšší. Pro spuštění pouze CPU části není takováto grafická karta vyžadována.

Program je parametrizován pomocí přepínačů na příkazové řádce, struktura přepínačů je následující:

```
-r <reference file> -in <video to compare> -in  
<second video to compare> -type <STVSSIM, SSIM or PSNR>  
[-ffpath <path to folder with ffmpeg and ffprobe  
executables>] [CUDA]
```

Podrobnější popis parametrů:

- -r Referenční video - například zdrojové video, ze kterého jsou odvozena další videa
- -in Testované video - například video odvozené ze zdrojového videa, tento parametr se může opakovat a je možné tak srovnávat více videí mezi sebou
- -type Typ metriky - může být jeden z množiny PSNR, SSIM, stvSSIM
- -ffpath Cesta do složky s binárními soubory ffmpeg a ffprobe - volitelný parametr, užitečný v případě, že na daném počítači není nainstalován program ffmpeg
- CUDA Pokud je přítomen tento parametr, využije se k výpočtu grafická karta, v opačném případě budou výpočty probíhat na CPU

#### 3.1 Implementace PSNR

PSNR část algoritmu jsem psal jako první, zároveň jsem na této metrice experimentoval s různými vstupy od programu ffmpeg, v první fázi jsem programem ffmpeg vytvářel sadu bmp obrázků (bmp obrázek je tvořen nekomprimovanou sadou pixelů, každý bajt souboru reprezentuje právě jeden pixel nebo jeho část) reprezentujících dané video, tuto sadu obrázků jsem pak načítal a počítal metriku PSNR. Poté jsem kvůli obrovské prostorové náročnosti nekomprimovaných bmp souborů přešel na bezztrátově komprimované png soubory. Bezztrátově komprimované v tomto kontextu značí obrázky, které prošly kompresí, je ale možné je dekomprimovat přesně do podoby původního obrazu, nedochází zde tedy ke ztrátě informace ale pouze ke snížení velikosti souboru na rozdíl od formátu bmp. Spolu s tím jsem ale musel implementovat načítání souborů png, toho jsem dosáhl využitím knihovny libpng. Přechodem z bmp na png soubory se dramaticky snížila náročnost na diskový prostor, ale stále bylo potřeba přibližně 100 kB na každý full HD snímek. Tento přístup

vyžadoval nejprve spuštění ffmpegu pro vygenerování png souborů a pak spuštění mého programu pro vyhodnocení těchto souborů. Tento přístup přináší mírné zrychlení při ohodnocení jednoho souboru vícekrát (například při hodnocení vzorového souboru s více různými deriváty), ale vyžaduje stále značný diskový prostor. Proto jsem později přešel na načítání přímo z ffmpegu do paměti mého programu bez ukládání jakýchkoliv dat na disk.

Algoritmus PSNR je možné implementovat buď nad černobílým snímkem, nebo pouze nad jedním kanálem a nebo nad více kanály obrazu (například RGB) vytvořením průměru z těchto kanálů. V počáteční fázi implementace jsem počítal průměr z metrik PSNR pro jednotlivé RGB kanály, později jsem z důvodu větší citlivosti lidského oka na jasovou složku obrazu a zároveň nižší výpočetní náročnosti přešel na výpočet pouze nad jasovou složkou obrazu.

## 3.2 Implementace SSIM

Jako další část implementace jsem zvolil SSIM, tedy algoritmus o něco složitější než PSNR.

Stejně jako u algoritmu PSNR, algoritmus postupně počítá metriku SSIM pro jednotlivé snímky. Výpočet jednoho snímku je tvořen dvěma vnořenými for cykly, jeden z nich prochází šířku snímku ( $x$ ), druhý z nich výšku snímku ( $y$ ). Při každém průchodu vnitřního cyklu je spočítána hodnota metriky pro daný čtverec začínající na souřadnicích ( $x, y$ ) o velikosti  $8 \times 8$  pixelů. Po vypočítání se pokračuje na další souřadnice, rozdíl mezi dvěma sousedními hodnotami pro  $x$  a  $y$  je definována direktivou SKIP\_SIZE. Na konci výpočtu snímku je již pouze spočítán průměr z mezivýpočtů vzniklých spočítáním metriky SSIM pro jednotlivé čtverce.

Stejně jako u PSNR, počítal jsem tuto metriku nad jasovou složkou obrazu. V prvních implementacích jsem z rgb24 formátu, který byl výstupem programu ffmpeg, vypočítával jasovou složku obrazu pomocí vzorce 3.1. Tato konverze ale trvala až 75 % výpočetního času. I proto jsem později implementaci upravil tak, aby výstupem ffmpegu byla mimo jiné i nekomprimovaná jasová složka, kterou jsem tak přímo využil.

$$Y = 0.299 \times Red + 0.587 \times Green + 0.114 \times Blue \quad (3.1)$$

## 3.3 Paralelizace PSNR a SSIM

Pro paralelizaci využívám knihovnu openMP, konkrétně využívám:

```
#pragma omp parallel for
```

Tato pragma slouží jako příkaz pro openMP, který říká, že na následujícím řádku bude for cyklus a že openMP má tento cyklus vykonat paralelně. Tato metoda má výhodu ve velmi jednoduché implementaci, nevýhodou je

### 3. IMPLEMENTACE

---

ztráta kontroly nad implementačními detailem. Paralelní výpočty metrik PSNR a SSIM na CPU probíhají za využití  $x$  jader následovně:

1. Načtení  $x$  snímků pomocí programu ffmpeg do předalokovaného paměťového bufferu. Ffmpeg standardně pracuje paralelně a využívá tedy při této operaci plný procesorový výkon.
2. Paralelní výpočet metriky pro  $x$  snímků najednou,  $x$  vláken počítá každé svůj snímek a výsledek uloží do sdíleného pole.
3. Synchronizace vláken a pokračování na další snímky.

Na konci dojde pouze k sériovému dopočítání posledních snímků zbývajících po paralelním výpočtu. Protože PSNR i SSIM metriky jsou poměrně rychlé, u videí s velkou náročností na procesor při dekomprezji může dojít k převážení výpočetního času bodu 1, z toho důvodu škálování rychlosti takové, jak by bylo možné očekávat z použitého počtu jader.

Tento přístup je velmi přímočarý a vzniklý kód je snadno čitelný, od sériové verze se příliš neliší. Nese s sebou ale i nevýhody, nedochází zde k paralelizaci mezi programem ffmpeg a mým programem. Kdyby běželo najednou dekódování videa a výpočet metriky, mohlo by dojít k lepšímu využití procesoru. Toho by se dalo dosáhnout například nějakou formou algoritmu producent/konzument[32]. Takováto implementace by ale byla mnohem složitější, nedala by se využít pro GPU část algoritmu a není jisté, že by přinesla citelný nárůst výkonu.

#### 3.3.1 SSIM s využitím CUDA

Model programování CUDA je heterogenní model programování za využití klasického procesoru ale i procesoru grafické karty. V CUDA modelu budu CPU a paměť RAM referencovat jako hostitel (host) a grafickou kartu budu referencovat jako zařízení (device). V rámci CUDA běží kód na hostiteli i na zařízení, hostitel může spravovat svoji paměť i paměť zařízení a může také spouštět kernely, což jsou funkce vykonávané na zařízení. Tyto kernely jsou vykonávány mnoha (řádově 128 až 1024) paralelními vlákny současně. Tato vlákna jsou uspořádána do bloků, každý kernel daném počtu bloků a každý blok obsahuje daný počet vláken. Celkový počet vláken kernelu je tedy  $\text{početbloků} \times \text{početvláken}$ . Typický CUDA program má následující strukturu:

1. Deklarování a alokování proměnných na hostiteli a zařízení.
2. Načtení vstupních dat do paměti hostitele.
3. Přenos těchto dat do paměti zařízení.
4. Spuštění jednoho nebo více kernelů.

5. Přenos výsledků ze zařízení na hostitele.

Někdy může být toto spouštění kernelů ve smyčce a pak po bodu 5 přecházíme zpět na bod 2, dokud nejsou zpracována všechna vstupní data.

Je několik možností jak algoritmus, SSIM implementovat. Původně jsem zamýšlel, že by každé vlákno dostalo přidělený snímek a ten by počítalo. Tento přístup by ale vyžadoval velmi vysoké množství paměti alokované na GPU, například pro 4K video ( $3840px \times 2160px$ ) by při počtu vláken 1024 bylo potřeba 8GB paměti, což by velmi limitovalo grafické karty, na kterých je tento program efektivně možno spustit. Proto jsem se rozhodl, že každé vlákno bude počítat pouze dílčí část snímku, konkrétně jeden  $8 \times 8$  pixelů velký čtverec. Po ukončení každého kernelu jsou výsledky nakopírovány do hostitelské paměti a je spočítán průměr pro daný snímek. Tato série kroků je opakována postupně pro všechny snímky. Paměťové nároky tohoto algoritmu jsou nízké, přibližně se rovnají počtu pixelů jednoho snímku  $\times$  počet testovaných souborů, řádově tedy několik MB. Tato data jsou však ukládána jak do hostitelské paměti, tak do paměti zařízení.

## 3.4 Implementace stVSSIM

Algoritmus stvSSIM je nejnáročnější na implementaci, je zároveň také nejnáročnější na výpočetní prostředky. U metrik PSNR a SSIM stačilo uchovávat v paměti jeden snímek, u stVSSIM je nutné v paměti uchovávat mnohem větší počet snímků, dle referenční implementace je to 33. V mému programu používám stejný počet snímků.

Tento algoritmus je také jako jediný z implementovaných algoritmů datově citlivý, konkrétně výpočet ARPS může trvat různě dlouhou dobu v závislosti na množství pohybu ve scéně. SSIM část stvSSIM algoritmu využívá předchozí implementaci.

### 3.4.1 Implementace stVSSIM s využitím CUDA

Tento algoritmus vychází z jeho CPU verze, SSIM-3D je implementovaná v CUDA verzi, SSIM část opět využívá GPU verzi SSIM algoritmu. ARPS část jsem na GPU neimplementoval, protože je v něm příliš mnoho nevyhnutelného větvení, což by mělo výrazný dopad na výkonnost celého algoritmu a ztratil by se zde benefit počítání na grafické kartě.

Největší problém při implementaci jsem měl s velmi omezenými možnostmi vícerozměrných polí na grafické kartě, proto jsem musel vícerozměrná pole, použitá v CPU verzi tohoto algoritmu, převádět na jednorozměrná pole, z toho vznikaly problémy s indexací a o něco se zhoršuje čitelnost programu.



## Porovnání algoritmů

### 4.1 Srovnání se subjektivním testováním

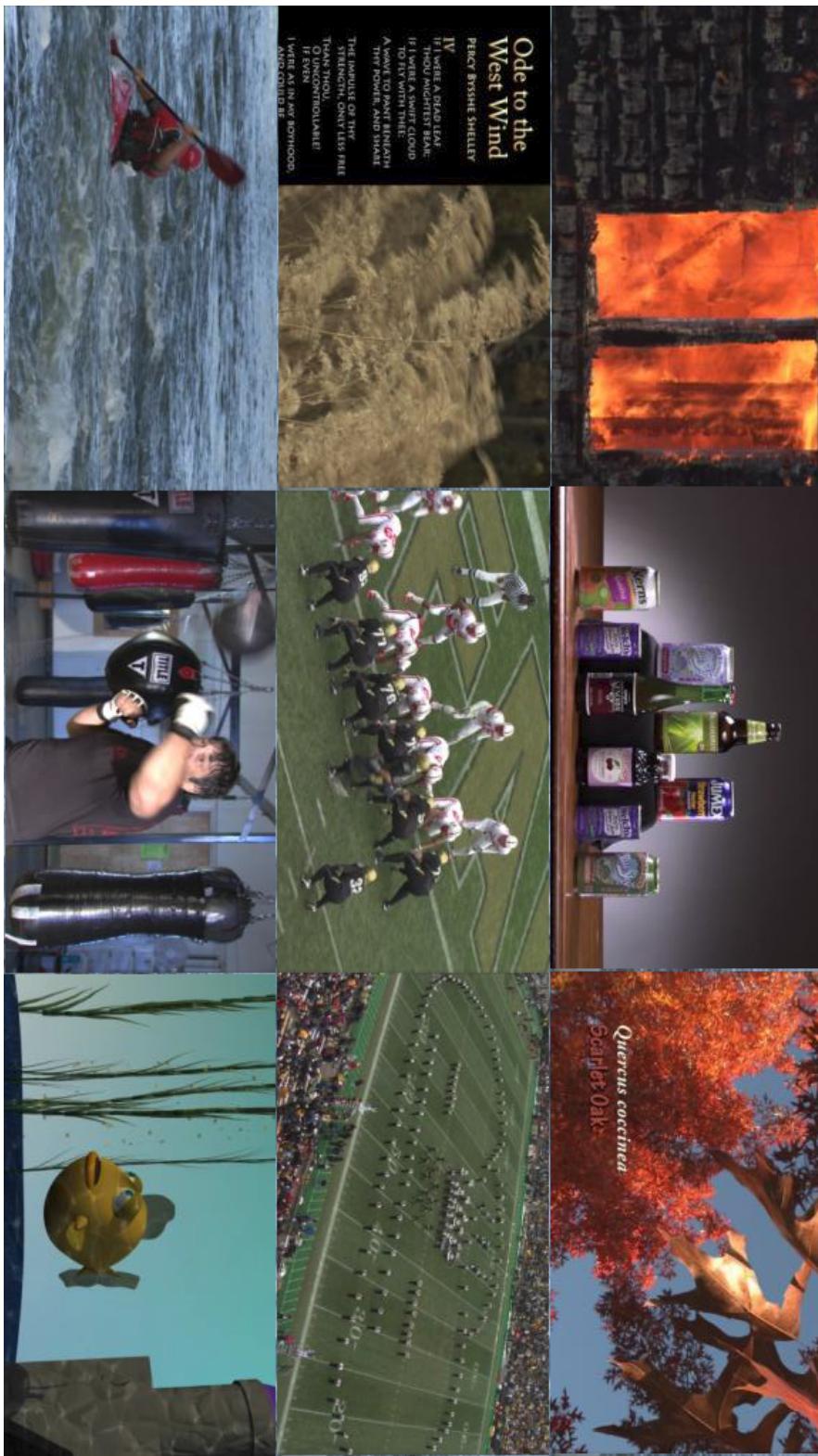
Protože je potřeba zjistit, který algoritmus je nejlepší pro určení kvality videa z pohledu lidského diváka, bylo potřeba tyto algoritmy porovnat s nějakou subjektivní studií. Použil jsem studii skupiny VQEG spadající pod ITU-T. ITU-T (International Telecommunication Union - Telecommunication Standardization Sector) je společnost zabývající se standardizací v telekomunikaci. Je zodpovědná například za standardy H.264 nebo H.265. Konkrétně jsem použil jejich projekt HDTV [33]. Tento projekt se zabývá videem v rozlišení  $1920 \times 1080$  a to v jeho prokládané (i) nebo neprokládané (p) verzi. Pro zjednodušení se omezím na 1080p (1080 px na výšku, neprokládaná verze) videa se snímkovou frekvencí 25 a 30 fps.

Budu počítat kvalitu mnou implementovaných algoritmů na 13 video sekvencích, z toho 9 je 1080p při 30 fps a zbylé 4 jsou 1080p 25 fps. Prvních 9 videosekvencí má každá originální soubor a dalších 15 odvozených s různou kvalitou, 4 poslední sekvence má každá celkem 6 verzí, z toho jeden originál. Ze druhé videosekvence byla již autory původní studie vyřazena jedna verze, toto vyřazení se promítlo i do mého testování. Všechny tyto sekvence je možné stáhnout na [34] po zaregistrování. Tato videa jsou všechna uložena bez komprese ve formátu uyvy422, jsou tedy nenáročná na dekódování, ovšem nevýhodou jsou požadavky na diskový prostor těchto snímků (1.2GB na 10 vteřin záznamu). Videa jsou z různých prostředí (včetně animovaného videa), proto by měla dobře pokrývat různé typy snímků. Bližší informace o testovaných videích je možné získat přímo ze závěrečné zprávy této studie [35]. Při subjektivním měření z této studie platí, že čím větší číslo, tím větší kvalita, spodní mez tohoto čísla je jedna, číslo 5 označuje kvalitu referenčního videa. V případě, že video bylo hodnoceno lépe než k němu přidružená reference, pak může být jeho kvalita vyšší než 5.

Na obrázcích 4.1 a 4.2 jsou vidět ukázky z testovacích videí pro demonstrování rozmanitosti těchto videí.

#### 4. POROVNÁNÍ ALGORITMŮ

---



Obrázek 4.1: VQEG HDTV Sada 1, video snímek 1 až 9

#### 4.1. Srovnání se subjektivním testováním



Obrázek 4.2: VQEG HDTV, video snímek 11 až 14

#### 4. POROVNÁNÍ ALGORITMŮ

---

Video set	Video	PSNR - medián	SSIM - medián	SSIM-3D - medián	SSIM-16 - medián	stVSSIM	Subjektivní
1	1	25.0194	0.7106	0.7192	0.6978	0.5195	2.3333
1	2	25.8597	0.7091	0.7886	0.7099	0.5962	2.2083
1	3	24.9212	0.7106	0.5948	0.6742	0.5165	1.6667
1	4	25.2083	0.6824	0.7011	0.6815	0.4723	1.5000
1	5	25.8727	0.7091	0.7886	0.7099	0.5962	2.2917
1	6	27.6228	0.7085	0.8512	0.7069	0.5949	3.3333
1	7	24.9920	0.7085	0.7723	0.7069	0.5644	2.8333
1	8	24.5091	0.6721	0.6606	0.6690	0.4419	1.7500
1	9	27.6228	0.7085	0.8512	0.7069	0.5949	3.1667
1	10	26.8476	0.7077	0.8613	0.7316	0.5933	4.5000
1	11	26.8477	0.7077	0.8627	0.7348	0.5933	4.6667
1	12	25.3892	0.7077	0.8534	0.7348	0.5933	3.4167
1	13	24.8687	0.7077	0.7123	0.6950	0.5933	2.9167
1	14	24.9250	0.6729	0.6603	0.6677	0.4187	1.7500
1	15	26.8244	0.7071	0.8743	0.7578	0.5929	4.8333

Tabulka 4.1: VQEG HDTV video 1, mediány

V tabulkách 4.1 a 4.2 je možné nalézt hodnoty mediánů a průměrů pro jednotlivé video snímky. Základ pro výpočet těchto hodnot jsou hodnoty metrik pro každý snímek z daného videa, nebo hodnoty metrik pro subsety snímků v případě stVSSIM. Díky tomu, že platí, že u všech metrik (objektivních i subjektivní) vyšší hodnota znamená lépe hodnocené video, tak i vyšší korelace je vždy lepší, korelace se pohybuje v mezích od -1 do 1 včetně.

Hodnoty SSIM-16 a SSIM-3D jsou pouze pomocné hodnoty pro stVSSIM, výsledná hodnota stVSSIM je rovna součinu těchto hodnot, jak je zmíněno v kap. 1. V tabulce 4.1 se nachází hodnoty mediánů vypočítaných pro jednotlivá videa z video setu 1 a zároveň se zde nachází hodnoty subjektivních hodnocení těchto videí. Korelace mediánů hodnot metrik s hodnotami subjektivního měření byla nižší, než hodnoty korelace průměrů se subjektivními hodnotami (viz tabulka 4.5). Z toho důvodu budu v dalších tabulkách uvádět průměry hodnot, nikoliv mediány. Hodnoty metrik, stejně jako hodnoty subjektivního hodnocení se vždy stahují k jednomu video setu, kvalita videa určená metrikami i subjektivním pozorováním je vždy relativní k danému video setu, není tedy možné porovnávat navzájem data z různých video setů. V tabulce 4.2 se nachází hodnoty průměrů vypočítaných pro jednotlivá videa z video setu 1 a zároveň se zde nacházejí hodnoty subjektivních hodnocení

#### 4.1. Srovnání se subjektivním testováním

---

Video set	Video	PSNR - průměr	SSIM - průměr	SSIM-3D - průměr	SSIM-16 průměr	stVSSIM - průměr	Subjektivní
1	1	24.7259	0.6858	0.7091	0.6961	0.4996	2.3333
1	2	26.2263	0.7217	0.7904	0.7310	0.5813	2.2083
1	3	22.7051	0.6532	0.6007	0.6559	0.3939	1.6667
1	4	25.2849	0.6838	0.7062	0.6864	0.4909	1.5000
1	5	26.5827	0.7253	0.7933	0.7310	0.5837	2.2917
1	6	27.3132	0.7434	0.8373	0.7385	0.6222	3.3333
1	7	25.8207	0.7239	0.7943	0.7172	0.5725	2.8333
1	8	24.4380	0.6649	0.6780	0.6597	0.4574	1.7500
1	9	27.3132	0.7434	0.8373	0.7385	0.6222	3.1667
1	10	27.6195	0.7634	0.8524	0.7525	0.6461	4.5000
1	11	27.6448	0.7646	0.8545	0.7543	0.6492	4.6667
1	12	26.7664	0.7518	0.8346	0.7431	0.6231	3.4167
1	13	25.1964	0.7045	0.7513	0.7004	0.5330	2.9167
1	14	22.8783	0.6389	0.6135	0.6431	0.3974	1.7500
1	15	27.6857	0.7721	0.8557	0.7604	0.6563	4.8333

Tabulka 4.2: VQEG HDTV video 1, průměry



Obrázek 4.3: VQEG HDTV snímek 2, subjektivně nízká kvalita

těchto videí. Hodnoty SSIM-3D a SSIM-16 velmi dobře korelují se subjektivním hodnocením, průměr z SSIM-3D koreluje dokonce nejlépe ze všech veličin. Tato velmi dobrá korelace u SSIM-3D je dána především tím, že v daných testovacích videosekvenčích nedochází k prudké změně scény. Za těchto podmínek by SSIM-3D koreloval o něco hůře. Autoři zmíněné subjektivní studie ovšem počítali pouze s metrikami založenými na počítání snímek po snímku, proto nebylo potřeba zahrnovat prudké změny scény. Protože se jedná pouze o mezinárodní srovnání, nebudu je v dalších tabulkách uvádět. Je možné ale tyto hodnoty dohledat v příloze, stejně jako mediány.

Tabulka 4.3 je vzata z testovacího videa 2 (na obrázku 4.1 nahoře upro-

#### 4. POROVNÁNÍ ALGORITMŮ

---



Obrázek 4.4: VQEG HDTV snímek 2, subjektivně vysoká kvalita

Video set	Video	PSNR - průměr	SSIM - průměr	stVSSIM - průměr	Subjektivní
2	3	16.7679	0.5867	0.3178	1.2500
2	14	16.0693	0.5603	0.2674	1.3750
2	8	16.7505	0.6122	0.3776	1.4167
2	4	16.7891	0.5891	0.3241	1.6667
2	1	16.8014	0.5888	0.3251	1.7500
2	5	16.7887	0.5891	0.3242	1.9167
2	2	16.7891	0.5891	0.3241	1.9583
2	12	16.6889	0.6279	0.4051	2.2917
2	6	17.0381	0.6367	0.4309	2.5000
2	7	17.0423	0.6373	0.4321	2.5000
2	13	16.2741	0.5829	0.3183	2.5000
2	9	16.5773	0.5901	0.3324	3.0000
2	11	16.3306	0.5891	0.3280	4.7500
2	10	15.8844	0.5508	0.2328	4.8333

Tabulka 4.3: VQEG HDTV video 2, průměry

střed). Toto video mělo jako jediné zápornou korelaci mezi objektivními a subjektivními hodnotami. Tento fakt je možné velmi dobře pozorovat díky seřazení dle subjektivního hodnocení. Zatímco subjektivní hodnocení roste, objektivní hodnocení ve všech případech klesají. Tuto situaci si vysvětlují zejména tím, že, jak je vidět na obrázku 4.3, pozorovatel je negativně ovlivněn téměř nečitelnými titulkami, ale pro program není mezi tímto obrázkem a originálem 4.4 příliš velký rozdíl, zejména, pokud přihlédneme k faktu, že výpočty probíhají nad jasovou složkou.

V prvních 9 videích je při zhoršování kvality zaměřeno spíše na rozpadání chyby při přenosu videa, rozpadání obrazu a podobně, jak je možné vidět i na obrázku 4.3. Na dalších 4 videích je mířeno spíše na kvalitu a ztrátovost komprese, typicky se přílišná komprese projevuje kostičkováním obrazu, viz 4.5,

#### 4.1. Srovnání se subjektivním testováním

---



Obrázek 4.5: VQEG HDTV snímek 12, subjektivně nízká kvalita, kostičkování

oproti výřezu z originálního snímku 4.6. V této práci mě zajímají právě chyby vzniklé kompresí, nikoliv chyby vzniklé při přenosu, a proto je zde souhrnná tabulka 4.4 z videí 11-14 . Zde je možné pozorovat velmi dobrou korelací mezi subjektivním a objektivními metrikami.

V tabulce 4.5 je zaznamenána korelace mediánů pro každý set videí, zároveň jsou v této tabulce uvedeny i průměry korelací pro jednotlivé metriky. Dle těchto průměrů nejlépe koreluje metrika stVSSIM, jako druhá je metrika PSNR a SSIM je na posledním místě.

V tabulce 4.6 je zaznamenána korelace průměrů pro každý set videí, všechny metriky škálují lépe pro průměr než pro medián, proto nebudu medián dále používat. Nejlépe škáluje metrika stVSSIM, o něco hůře SSIM, nejhůře pak PSNR, pro měření kvality kodeků tedy budu převážně používat metriky SSIM a stVSSIM.

Pokud se omezíme pouze na videa s chybami způsobenými velkou kompresí, vyjde korelace průměrů dle tabulky 4.7. I dle této tabulky stVSSIM koreluje nejlépe a druhé je SSIM. Na této tabulce je však zajímavé, že SSIM i stVSSIM mají korelací vyšší než 0.98, je tedy vidět, že při tomto typu chyb jsou tyto algoritmy velmi vhodné.



Obrázek 4.6: VQEG HDTV snímek 12, subjektivně vysoká kvalita

## 4.2 Srovnání rychlosti algoritmů

### 4.2.1 Parametry měření

Veškerá měření výkonu probíhala na serveru star.fit.cvut.cz. CPU výpočty jsem pouštěl na uzlech gpu-01 a gpu-02, které mají z hlediska CPU shodné parametry. Jejich detaily jsou níže:

- CPU: 2ks 6core Xeon 2620 v2 @ 2.1Ghz
- RAM: 32GB

GPU výpočty jsem pouštěl na uzlu gpu-01 na jeho grafické kartě Tesla K40c. Její specifikace jsou opět níže:

- GPU: Tesla K40c
- Počet CUDA jader: 2880
- Frekvence: 745Mhz
- Pamět grafického adaptéru: 12GB GDDR5

video set	video	PSNR - průměr	SSIM - průměr	stVSSIM - průměr	Subjektivní
11	1	29.2518	0.8028	0.6813	<i>2.3750</i>
11	8	33.5311	0.8911	0.8358	<i>4.1667</i>
11	4	32.1385	0.8839	0.8017	<i>4.2500</i>
11	11	33.9190	0.9014	0.8378	<i>4.5000</i>
11	12	36.4729	0.9395	0.9050	<i>5.1250</i>
12	10	25.9782	0.6891	0.5228	<i>1.7500</i>
12	7	27.8764	0.7558	0.6287	<i>3.0417</i>
12	3	29.7273	0.8246	0.7066	<i>3.1667</i>
12	1	31.7382	0.8724	0.7894	<i>4.3333</i>
12	11	31.8664	0.8979	0.8214	<i>4.6667</i>
13	1	25.1179	0.5881	0.4624	<i>1.6250</i>
13	8	27.8030	0.6691	0.5996	<i>3.0417</i>
13	5	29.5029	0.7521	0.6924	<i>3.5417</i>
13	11	29.3155	0.7809	0.7204	<i>4.2500</i>
13	2	35.0019	0.9068	0.8839	<i>4.7917</i>
14	1	24.8181	0.6595	0.4973	<i>1.6667</i>
14	6	26.9963	0.7373	0.6218	<i>2.5417</i>
14	4	30.6962	0.8764	0.7949	<i>4.0833</i>
14	5	32.5885	0.9167	0.8578	<i>4.5833</i>
14	9	35.8078	0.9512	0.9169	<i>4.8333</i>

Tabulka 4.4: VQEG HDTV video 11 až 14, průměry

- Výpočetní výkon: 4.29Tflops
- Výpočetní výkon v dvojitě přesnosti: 1.43 Tflops

Metriky SSIM a PSNR jsou ve smyslu výpočetní náročnosti nezávislé na vstupních datech. Rychlosť týchto algoritmů závisí pouze na rozlišení snímků a na počtu snímků. Z toho dôvodu nemá smysl testovať mnoho rôznych videí pri stejném rozlišení, výsledky by odráželi pouze náročnosť dekódovania daného videa programom ffmpeg. Pro měření škálování programu s rostoucím počtem jader jsem zvolil videosekvence nenáročné na dekódování na omezení ovlivnění výsledků programem ffmpeg.

Metrika stVSSIM je na rozdíl od předchozích metrik závislá na samotných datech, a to díky výpočtu ARPS, který je popsán v sekci 1.4.2. Z toho dôvodu jsem provedl měření na stejné sadě videí, na které probíhalo měření pro srovnání se subjektivní metrikou, abych zjistil závislost délky výpočtu stVSSIM na konkrétním videu.

Souhrnná data z měření datové citlivosti metriky stVSSIM pro všech 13 referenčních videí jsou znázorněna v tabulce 4.8, u každého videa jsem srovná-

#### 4. POROVNÁNÍ ALGORITMŮ

---

Video set	PSNR - medián	SSIM - medián	SSIM-3D - medián	SSIM-16 - medián	stVSSIM - medián
1	0.690596	0.500117	0.841331	0.898835	0.684866
2	-0.537575	-0.066981	-0.272943	-0.167096	-0.261822
3	0.840888	0.728316	0.863023	0.918513	0.833131
4	0.644429	-0.666082	0.731151	0.722859	0.601464
5	0.762517	0.400643	0.750691	0.620806	0.498402
6	0.165194	0.092715	0.319951	0.503333	0.183508
7	0.723941	0.678180	0.693482	0.630734	0.771067
8	-0.371889	-0.080195	0.357084	0.409720	0.279103
9	-0.042905	0.223276	0.371481	0.054575	0.347050
11	0.939118	0.985753	0.964853	0.988268	0.984104
12	0.958389	0.956998	0.991969	0.974914	0.997023
13	0.936142	0.932994	0.957339	0.967960	0.951329
14	0.980073	0.995700	0.990503	0.998709	0.988471
Průměrná korelace	<b>0.514532</b>	<b>0.437033</b>	<b>0.658455</b>	<b>0.655548</b>	<b>0.604438</b>

Tabulka 4.5: Korelace mediánů pro jednotlivé sady videí

Video set	PSNR - průměr	SSIM - průměr	SSIM-3D - průměr	SSIM-16 - průměr	stVSSIM - průměr
1	0.805992	0.885327	0.825809	0.835021	0.843664
2	-0.507684	-0.209836	-0.250340	-0.192756	-0.224113
3	0.870728	0.888306	0.849342	0.875317	0.861515
4	0.810155	0.690776	0.739492	0.747026	0.770538
5	0.507270	0.742464	0.753244	0.742011	0.760174
6	0.357782	0.122064	0.449968	0.134241	0.285002
7	0.788094	0.793338	0.749755	0.724006	0.759719
8	0.102863	0.176697	0.274505	0.277544	0.268531
9	0.300671	0.246319	0.457494	0.266381	0.398861
11	0.949565	0.994421	0.961818	0.993162	0.980836
12	0.971818	0.968538	0.989633	0.969249	0.981567
13	0.903291	0.962162	0.976068	0.958560	0.971238
14	0.967705	0.999412	0.986196	0.999323	0.997367
Průměrná korelace	<b>0.602173</b>	<b>0.635384</b>	<b>0.674076</b>	<b>0.640699</b>	<b>0.665761</b>

Tabulka 4.6: Korelace průměrů pro jednotlivé sady videí

### 4.3. Srovnání rychlosti metrik

---

Video set	PSNR - průměr	SSIM - průměr	stVSSIM - průměr
11	0.949565	0.994421	0.980836
12	0.971818	0.968538	0.981567
13	0.903291	0.962162	0.971238
14	0.967705	0.999412	0.997367
Průměrná korelace	<b>0.948095</b>	<b>0.981133</b>	<b>0.982752</b>

Tabulka 4.7: Korelace průměrů pro videa 11 až 14

	Čas výpočtu	Relativní odchylka
1	49.90673	0.165911
2	37.17427	0.131542
3	34.08353	0.203747
4	48.6912	0.137514
5	49.52863	0.157078
6	46.69673	0.09092
7	23.72017	0.445854
8	34.58287	0.192082
9	38.11317	0.109608
11	44.25327	0.033836
12	55.75357	0.302504
13	39.56377	0.075719
14	54.39597	0.270788
Průměr	42.80491	0.178239

Tabulka 4.8: Průměrné relativní odchylky při čase výpočtu stVSSIM

val originální video a 3 odvozené verze. Průměrná relativní odchylka je 0.178, je potřeba tedy tuto odchylku mít na paměti při stanovování výsledků.

### 4.3 Srovnání rychlosti metrik

Pro toto srovnání jsem zvolil video co nejpodobnější reálnému použítí, a to trailer k filmu Marvel's Avengers: Age of Ultron, stažený z [36]. Srovnával jsem dvě přibližně stejně velká videa o Full HD rozlišení (konkrétně  $1920 \times 800$  pixelů), jedno s formátem h264 a crf 26 (crf bylo detailně popsáno v sekci 1.5.5), druhé s formátem h265 a crf 22, obě videa měla přibližně 36.5 MB při délce 130 sekund a 3121 snímku. Níže se nachází souhrnná tabulka 4.9.

Hodnota pro metriku stVSSIM při 1 CPU jádře je odhadnuta z dob běhu pro více jader, tato úloha nedoběhla v časovém limitu úloh na serveru star (30

#### 4. POROVNÁNÍ ALGORITMŮ

---

Metrika	CPU/GPU	Počet vláken	Čas (s)
<b>PSNR</b>	<b>CPU</b>	<b>1</b>	<b>37.6078</b>
PSNR	CPU	2	26.1386
PSNR	CPU	4	21.8604
PSNR	CPU	6	21.3599
PSNR	CPU	12	20.3831
<b>PSNR</b>	<b>CPU</b>	<b>24</b>	<b>20.2511</b>
<b>SSIM</b>	<b>CPU</b>	<b>1</b>	<b>111.37</b>
SSIM	CPU	2	60.6387
SSIM	CPU	4	35.3204
SSIM	CPU	6	27.7424
SSIM	CPU	12	23.2465
<b>SSIM</b>	<b>CPU</b>	<b>24</b>	<b>22.5449</b>
<b>SSIM</b>	<b>GPU</b>	<b>64</b>	<b>20.09</b>
SSIM	GPU	128	22.1376
SSIM	GPU	256	21.8936
SSIM	GPU	512	22.0869
SSIM	GPU	1024	21.8099
<b>stVSSIM</b>	<b>CPU</b>	<b>1</b>	<b>2813.51</b>
stVSSIM	CPU	2	1427.58
stVSSIM	CPU	4	724.365
stVSSIM	CPU	6	504.247
stVSSIM	CPU	12	289.567
<b>stVSSIM</b>	<b>CPU</b>	<b>24</b>	<b>256.617</b>
<b>stVSSIM</b>	<b>GPU</b>	<b>64</b>	<b>349.783</b>
stVSSIM	GPU	128	388.064
stVSSIM	GPU	256	385.366
stVSSIM	GPU	512	351.42
stVSSIM	GPU	1024	350.921

Tabulka 4.9: Souhrnná tabulka srovnání rychlosti metrik, Marvel's Avengers: Age of Ultron - Trailer 3

### 4.3. Srovnání rychlosti metrik

---

Metrika	CPU/GPU	Čas (s)	Čas / snímek (s)	Čas / film (min)
PSNR	CPU	20.2511	0.006489	18.68733
SSIM	CPU	22.5449	0.007224	20.80401
stVSSIM	CPU	256.617	0.082223	236.8013
SSIM	GPU	20.09	0.006437	18.53867
stVSSIM	GPU	349.783	0.112074	322.7732

Tabulka 4.10: Nejlepší časy jednotlivých metrik a časy na snímek a film

minut). Počet vláken u GPU verze značí počet vláken v rámci jednoho bloku (detailly viz sekci 3.3.1), počet bloků je volen takový, aby každé vlákno počítalo právě jednu oblast snímku. Z této tabulky je zřejmé, že metriky PSNR a SSIM jsou brzděny rychlostí enkódování videa při větším počtu vláken. Zároveň je také zřejmé, že metrika stVSSIM je podstatně pomalejší než zbylé dvě metriky a to i při zahrnutí možného mírného ovlivnění výsledků způsobeného datovou citlivostí algoritmu stVSSIM. Také platí, nejlepší metrika je nejpomalejší a naopak.

GPU verze algoritmu stVSSIM je limitovaná tím, že výpočet ARPS probíhá na CPU a to z důvodu nevyhnutelného velkého množství podmíněného větvení tohoto algoritmu. Implementovat tuto část na GPU by nepřineslo zrychlení, protože výkon v CUDA razantně klesá při takovémto větvení programu [37]. GPU verze SSIM algoritmu je pouze o málo rychlejší protože je limitována rychlostí enkódování videa na CPU. Zároveň jsou na v této implementaci velmi často nahrávána data do a z paměti zařízení, při jiném řešení by ale vznikl problém s kapacitou paměti GPU, viz sekci 3.3.1.

V tabulce 4.10 jsou vidět nejlepší výsledky jednotlivých metrik včetně časů potřebných pro průměrné zpracování jednoho snímku a odvozeného času potřebného na porovnání dvou hodinového filmu při 24 fps. Pro metriku stVSSIM je toto měření spíše orientační z důvodu nedostatečného počtu různých testovacích videí.



## Porovnání kodeků

V této kapitole se budu zabývat porovnáváním kodeků pro HD obsah, HD obsahem se v tomto kontextu rozumí videa s alespoň HD rozlišením ( $1280px \times 720px$ ). Konkrétně se omezím na dnes velmi často používané kodeky h264, h265, VP9. Cílem této kapitoly je určit, který codec je nejlepší.

Aby bylo možné určit, který codec je nejlepší, je ale nejprve potřeba stanovit způsob porovnávání kodeků. Zatím víme pouze to, že čím vyjde metrika vyšší, tím je dané video blíže k originálu. Z toho je možné říct, že v rámci setu odvozených videí od jednoho referenčního videa má video s vyšší hodnotou metriky vyšší kvalitu než video s nižší hodnotou metriky. Pro samotné porovnání kodeků je možné například použít porovnání, kolikrát kvalitnější je výstup při použití jednoho koduku oproti výstupu z koduku jiného při stejné velikosti výstupních souborů. Protože ale metrika SSIM ani stVSSIM nemají lineární škálu, není možné říct, že protože metrika vyšla například dvojnásobná, video je dvojnásobně lepší. Takto je pouze možné rozhodnout, že při dané velikosti má nějaký koduk vyšší kvalitu, není ale možné říct o kolik vyšší kvalitu. Z toho důvodu jsem se rozhodl použít porovnání velikosti výstupních souborů při dané kvalitě videa. Zkoumám tedy jak veliké video je potřeba na dosažení dané kvality při použití různých kodeků. Z toho je pak také možné určit, o kolik větší soubory jsou potřeba při stejně kvalitních videích v různých kodecích.

Dalším krokem bylo připravit testovací sadu videí. Jako referenční videa jsem použil testovací sadu videí z projektu VQEG HDTV[33], který je zmíněn v předchozích kapitolách. Z každého referenčního videa jsem pomocí programu ffmpeg vygeneroval sadu odvozených videí, a to videa o různých kvalitách a s různými použitými kodekami. Protože u odvozených videí nebylo primárním cílem dosáhnout určité velikosti nebo přesného datového toku, použil jsem při generování parametr *crf*, konkrétně jsem použil na konverzi příkaz:

```
ffmpeg -i in.avi -c:v libx264 -crf 0 out.mkv
```

Kdy jednotlivé parametry znamenají:

## 5. POROVNÁNÍ KODEKŮ

---

```
-i vstupní soubor  
-c:v výstupní video kodek použitý pro kompresi  
-crf určuje kvalitu videa  
out.mkv výstupní soubor
```

Takto jsem vygeneroval pro každý ze 3 kodeků celkem 32 odvozených videí s crf v rozmezí 0 až 51, v oblasti 18 až 28 jsem generoval video pro každou hodnotu crf, ve zbylých oblastech jsem generoval videa pro sudé hodnoty crf a pro 51 z důvodu doporučeného používání hodnot v rozsahu 18 až 26 [11]. Pro 13 referenčních videí jsem tedy vytvořil celkem  $3 \times 32 \times 13 = 1248$  videí. I přesto, že videa s kodekem vp9 při crf rovno 0 (nejlepší kvalita) mají přibližně srovnatelnou velikost jako videa h264 a h265 se stejnou hodnotou crf, při vysokých hodnotách crf (40 a výše) mají videa s vp9 kodekem přibližně desetinásobnou velikost, ffmpeg tedy není schopen vygenerovat videa s kodekem vp9 při srovnatelné kvalitě jako video s kodeky h264 nebo h265 s crf vysším než 40. Je potřeba tedy tuto skutečnost vzít v potaz při srovnávání metrik.

Po vygenerování videí jsem provedl měření na všech odvozených videích, pro každé video jsem počítal metriky SSIM a stVSSIM vzhledem k jeho referenčnímu videu. Metriku PSNR jsem vyneschal, protože jak ukázala měření výše, je mnohem méně přesná a má přibližně stejné výpočetní nároky jako SSIM. Ukázka z části měření pro jednu sadu videí se nachází v tabulce 5.1. Všechny velikosti souborů v této i dalších tabulkách jsou v MB.

I z této ukázky je vidět, že není možné porovnat přímo data v jednotlivých řádcích, protože obzvláště pro vyšší crf neplatí, že videa se shodným crf a rozdílným kodekem mají stejnou kvalitu (příklad v tabulce zvýrazněn *kurzívou*). Bylo tedy potřeba tato data nějak normalizovat. Je možné buď srovnávat data se stejnou kvalitou, a nebo data se stejnou velikostí souboru, jak jsem již zmínil výše, větší vypovídající hodnotu má srovnávání dat se stejnou (nebo co nejpodobnější) kvalitou.

Pro tuto normalizaci jsem vždy vzl jeden výchozí kodek a postupně k jeho jednotlivým souborům o různých kvalitách jsem přiřazoval vždy nejbližší soubor s jiným kodekem. V tabulce 5.2 jsou nejprve všechny hodnoty velikostí souborů a jejich příslušné hodnoty metriky SSIM, dále jsou tam také velikosti souborů pro kodeky h264 a h265 seřazené tak, aby kvality těchto souborů v daném řádku odpovídaly kvalitě výchozího kodeku, v tomto případě vp9. Toto srovnávání jsem dělal nezávisle pro metriky SSIM a stVSSIM s postupně každým ze tří kodeků použitým jako výchozí a zbylé dva se k němu vztahovaly.

Když mám takto seřazená data, vytvářím z nich relativní rozdíly jejich velikostí, mezi každými dvěma kodeky. Pro každé dva kodeky v daném řádku (v každém řádku se nachází velikosti souborů s co nejbližšími hodnotami metrik) s danou metrikou vytvořím relativní odchylku od jejich průměru a to pomocí vzorce:  $(A - B)/((A + B)/2)$ . Kde A a B jsou hodnoty metrik pro

	h264	h264	h264	h265	h265	h265	vp9	vp9	vp9
crf	Vel. soub.	SSIM	stVSSIM	Vel. soub.	SSIM	stVSSIM	Vel. soub.	SSIM	stVSSIM
0	327.94	1.0000	1.0000	212.28	0.9974	0.9974	354.75	0.9999	1.0000
4	198.08	0.9974	0.9963	139.76	0.9952	0.9947	109.42	0.9950	0.9935
8	131.49	0.9952	0.9932	89.01	0.9916	0.9905	66.90	0.9903	0.9874
12	86.00	0.9912	0.9872	53.48	0.9856	0.9832	48.06	0.9854	0.9811
16	50.73	0.9821	0.9736	29.84	0.9759	0.9711	36.52	0.9805	0.9746
19	31.41	0.9722	0.9588	18.49	0.9667	0.9588	31.84	0.9778	0.9709
21	22.90	0.9652	0.9487	13.55	0.9598	0.9490	28.76	0.9757	0.9681
23	16.86	0.9573	0.9373	10.09	0.9520	0.9379	26.49	0.9739	0.9656
25	12.54	0.9480	0.9238	7.57	0.9426	0.9243	23.61	0.9712	0.9619
27	9.45	0.9367	0.9070	5.70	0.9310	0.9073	20.99	0.9682	0.9577
30	6.28	0.9146	0.8740	3.68	0.9088	0.8742	18.21	0.9640	0.9519
34	3.59	0.8732	0.8104	1.94	0.8688	0.8127	13.20	0.9569	0.9417
38	1.99	0.8187	0.7226	0.93	0.8193	0.7339	9.80	0.9480	0.9292
42	1.08	0.7610	0.6233	0.42	0.7660	0.6431	7.36	0.9376	0.9146
46	0.60	0.7164	0.5416	0.27	0.7302	0.5733	5.57	0.9255	0.8974
50	0.39	0.6932	0.4965	0.29	0.7217	0.5616	4.26	0.9118	0.8777
51	0.37	0.6896	0.4889	0.31	0.7210	0.5613	3.99	0.9081	0.8724

Tabulka 5.1: Vybraná část z video setu 1, bez úprav

daný kodek v daném řádku. Toto číslo vyjadřuje odchylku od jejich průměru, pokud je záporné, pak  $A$  potřebuje méně dat na dosažení stejné kvality jako  $B$ . Toto číslo je možné jednoduše převést na údaj určující o kolik procent více dat je potřeba při použití jednoho kodeku oproti druhému kodeku. Tyto odchylky pro video set 1 jsou vidět v tabulce 5.3, ke kodeku h264 jsem přiřazoval nejbližší verze h265 a vp9, kodek h264 budu tedy v tomto případě nazývat referenční. V tabulce 5.3 je vidět, že kodek vp9 ztrácí výrazně oproti h265 zejména od  $crf = 38$ , to je způsobeno tím, že vp9 při těchto hodnotách crf generuje přibližně desetinásobná videa oproti h264 a h265. Nemá tedy žádného kandidáta pro velmi nízké hodnoty metrik SSIM a stVSSIM, proto je dosazen soubor se sice vyšší metrikou, ale s výrazně větší velikostí. Tato data je potřeba z měření odfiltrovat, vedlo by to pouze ke zkreslení.

V tabulce 5.4 již jsou průměry pro jednotlivá videa, kodek h264 je v tomto případě referenční, jedná se ale stále o data bez filtrování souborů, které nemají ekvivalent v jiných kodecích.

V tabulkách v souboru který je na přiloženém CD jsou i průměry pro jednotlivá videa, všechny vyhodnocované kodeky jsou postupně referenční. Meření pro kodek vp9 má nejmenší rozsah hodnot metrik, proto pro srovnání ve kterých figuruje kodek vp9 je toto potřeba zohlednit.

## 5. POROVNÁNÍ KODEKŮ

---

	h264	h264	h265	h265	vp9	vp9	h265	h264
crf	Vel. soub.	SSIM	Vel. soub.	SSIM	Vel. soub.	SSIM	Vel. soub. dle vp9	Vel. soub. dle vp9
0	327.94	1.000	212.28	0.997	354.75	1.000	212.28	327.9
2	241.46	0.998	172.61	0.996	151.56	0.997	172.61	198.1
4	198.08	0.997	139.76	0.995	109.42	0.995	139.76	131.5
6	161.44	0.996	112.69	0.994	83.19	0.993	112.69	107.0
8	131.49	0.995	89.01	0.992	66.90	0.990	69.48	86.0
10	107.02	0.994	69.48	0.989	55.75	0.988	69.48	67.4
12	86.00	0.991	53.48	0.986	48.06	0.985	53.48	67.4
14	67.39	0.988	40.36	0.981	41.78	0.983	40.36	50.7
16	50.73	0.982	29.84	0.976	36.52	0.980	40.36	50.7
18	36.94	0.976	21.68	0.970	33.19	0.979	40.36	36.9
19	31.41	0.972	18.49	0.967	31.84	0.978	29.84	36.9
20	26.76	0.969	15.79	0.963	29.99	0.977	29.84	36.9
21	22.90	0.965	13.55	0.960	28.76	0.976	29.84	36.9
22	19.63	0.961	11.68	0.956	27.78	0.975	29.84	36.9
23	16.86	0.957	10.09	0.952	26.49	0.974	29.84	31.4
24	14.52	0.953	8.74	0.948	25.44	0.973	29.84	31.4
25	12.54	0.948	7.57	0.943	23.61	0.971	21.68	31.4
26	10.87	0.943	6.57	0.937	22.15	0.970	21.68	26.8
27	9.45	0.937	5.70	0.931	20.99	0.968	18.49	26.8
28	8.23	0.930	4.94	0.924	19.90	0.967	18.49	22.9
30	6.28	0.915	3.68	0.909	18.21	0.964	15.79	22.9
32	4.77	0.896	2.70	0.890	14.93	0.960	13.55	19.6
34	3.59	0.873	1.94	0.869	13.20	0.957	11.68	16.9
36	2.69	0.847	1.36	0.845	11.36	0.952	10.09	14.5
38	1.99	0.819	0.93	0.819	9.80	0.948	8.74	12.5
40	1.47	0.789	0.62	0.793	8.51	0.943	7.57	10.9
42	1.08	0.761	0.42	0.766	7.36	0.938	6.57	9.4
44	0.80	0.735	0.31	0.743	6.38	0.932	5.70	8.2
46	0.60	0.716	0.27	0.730	5.57	0.926	4.94	8.2
48	0.47	0.703	0.27	0.724	4.85	0.919	4.94	6.3
50	0.39	0.693	0.29	0.722	4.26	0.912	3.68	6.3
51	0.37	0.690	0.31	0.721	3.99	0.908	3.68	6.3

Tabulka 5.2: Data z video setu 1, srovnána podle dat vytvořených kodekem vp9

	h264-h265		h264-vp9		h265-vp9	
crf	SSIM	stvSSIM	SSIM	stvSSIM	SSIM	stvSSIM
0	0.428218	0.428218	-0.07854	-0.07854	-0.502536491	-0.50254
4	-0.06921	0.137381	0.266064	0.266064	0.33373739	0.12987
8	-0.061	0.153983	0.183193	0.183193	0.2435128	0.029417
12	-0.03439	0.212548	0.249891	0.249891	0.283674654	0.037845
16	0.227773	0.518656	0.193463	0.325722	-0.03469276	-0.20144
19	0.366319	0.517618	0.209769	0.345787	-0.159616962	-0.17988
21	0.212933	0.513082	0.228202	0.421195	0.015456786	-0.09713
23	0.362748	0.502461	0.243296	0.389886	-0.122147118	-0.11837
25	0.35756	0.494222	0.245833	0.382802	-0.11423793	-0.11695
27	0.358886	0.494625	0.247559	0.386866	-0.113855949	-0.11317
30	0.52234	0.52234	0.383806	0.444681	-0.145843982	-0.08245
34	0.59578	0.59578	-0.10544	-0.10544	-0.690377522	-0.69038
38	0.726399	0.726399	-0.67058	-0.67058	-1.245330282	-1.24533
42	0.875124	0.875124	-1.14884	-1.14884	-1.617431903	-1.61743
46	0.63455	0.63455	-1.4783	-1.4783	-1.711484942	-1.71148
50	0.238921	0.238921	-1.64024	-1.64024	-1.711484942	-1.71148
51	0.172467	0.172467	-1.66164	-1.66164	-1.711484942	-1.71148

Tabulka 5.3: Relativní rozdíly ve velikosti při dané kvalitě pro jednotlivé kodeky a metriky, video set 1, h264 jako referenční kodek

	h264-h265		h264-vp9		h265-vp9	
Video set	SSIM	stvSSIM	SSIM	stvSSIM	SSIM	stvSSIM
1	0.34918	0.464614	-0.18509	-0.12882	-0.45886	-0.51946
2	0.245869	0.243446	-0.20793	-0.15306	-0.38757	-0.33333
3	0.338393	0.345032	0.004597	0.04134	-0.29162	-0.25974
4	0.273795	0.205427	-0.32582	-0.35666	-0.51071	-0.47274
5	0.309456	0.297636	-0.20827	-0.21469	-0.47565	-0.47065
6	0.367151	0.324653	-0.24182	-0.2233	-0.51725	-0.45514
7	0.501322	0.457276	-0.16375	-0.16375	-0.55708	-0.51215
8	0.57498	0.602445	0.064512	0.077559	-0.40177	-0.41611
9	0.346784	0.197966	-0.34861	-0.45051	-0.54543	-0.49723
11	0.294373	0.189992	-0.27951	-0.32645	-0.49443	-0.43849
12	0.345095	0.267557	-0.46716	-0.51137	-0.71388	-0.68232
13	0.32167	0.271792	-0.44953	-0.47823	-0.64855	-0.61588
14	0.372884	0.282621	-0.3169	-0.36181	-0.57298	-0.52743
Průměr	0.356996	0.319266	-0.24041	-0.24998	-0.50583	-0.47697

Tabulka 5.4: Průměr relativních rozdílů ve velikostech dat při stejně kvalitě, h264 jako referenční kodek

## 5. POROVNÁNÍ KODEKŮ

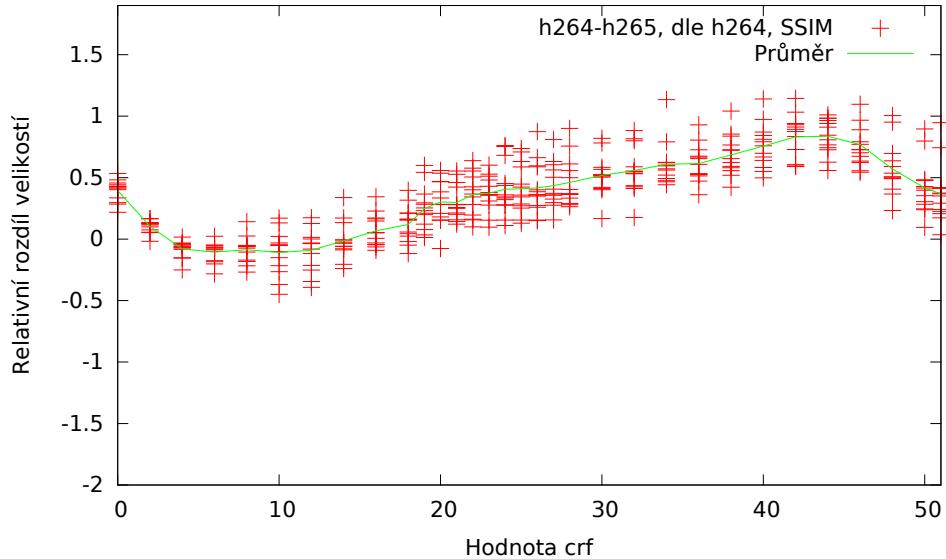
---

### 5.1 Vyhodnocení měření

V této sekci vyhodnotím celé měření a určím, který kodek je dle tohoto měření nejlepší, tato sekce je rozdělena na vyhodnocení kodeků z pohledu metriky SSIM a z pohledu metriky stVSIIM, zároveň je zde část zaměřená na porovnávání kvality 4K videa (video s rozlišením  $4096 \times 2160$  nebo podobným) a vhodnost jednotlivých kodeků pro 4K.

#### 5.1.1 SSIM

Všechny obrázky v této části jsou vztaženy na metriku SSIM. Na obrázku 5.1 je hodnota relativní odchylky mezi kodeky h264 a h265 (dle vzorce výše je h264 bráno jako A a h265 bráno jako B), tento a všechny další obrázky v této sekci vycházejí již z dat adekvátních (majících stejnou kvalitu) k referenčnímu kodeku. Průměr na tomto obrázku je vždy průměr všech měření pro danou kvalitu crf. Tento průměr není spojitý, ale pro větší přehlednost grafů oproti použití diskrétních průměrů ho budu v této sekci zobrazovat jako spojitý. Zároveň budu ve všech grafech používat stejné rozsahy os pro jednoduché porovnání mezi jednotlivými grafy. Na obrázku 5.2 jsou znázorněny průměry

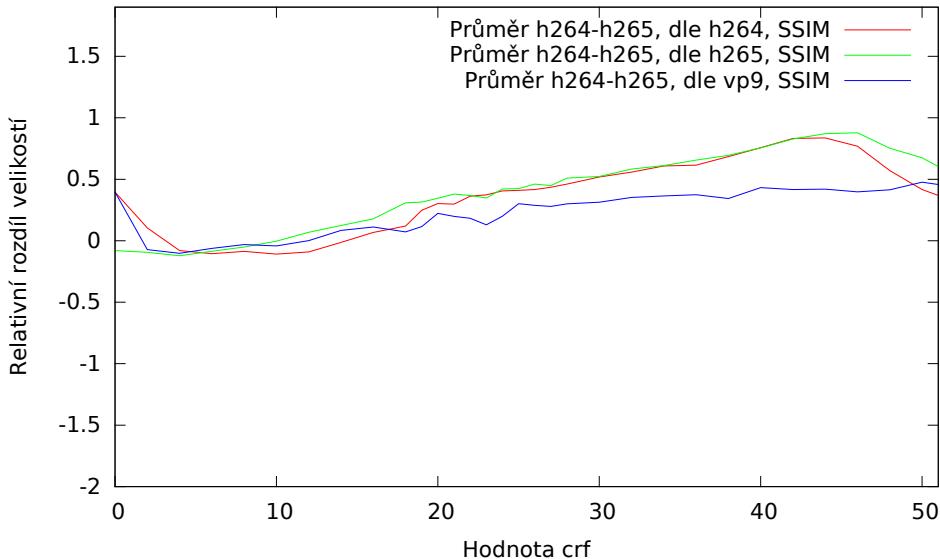


Obrázek 5.1: Rozdíl h264 a h265, h264 jako referenční, metrika SSIM

rozdílu h265 a h264 při různých referenčních kodéckách. Při referenčních kodéckách h264 a h265 vypadají průměry velmi podobně, kromě samotného začátku a konce, to je dáno lehce rozdílným rozsahem metriky pro kodeky h264 a h265. Při referenčním kodeku vp9 vypadá toto srovnání mírně jinak, to je

## 5.1. Vyhodnocení měření

---



Obrázek 5.2: Rozdíl h264 a h265, průměry všech kodeků jako referenčních, metrika SSIM

zapříčiněno odlišným rozsahem hodnot metriky pro tento kodek, nemá smysl tedy toto měření zohledňovat, pro stanovení závěru tedy použiju průměr z měření při referenčních kodecích h264 a h265, konkrétně s hodnotami crf od 4 do 46.

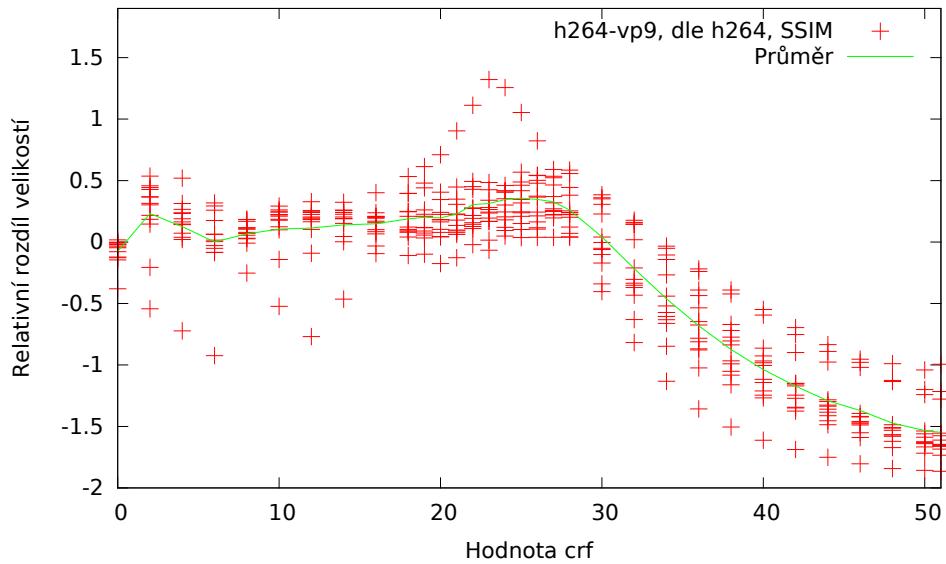
Dále z obrázku 5.2 vyplývá, že lepší vlastnosti kodeku h265 oproti h264 se naplno projeví až při vyšší míře komprese, v celém rozsahu uživatelsky rozumných hodnot crf je ale kodek h265 jasně lepší, než kodek h264.

Jako další budu porovnávat kodeky h264 a vp9, na obrázku 5.3 je hodnota relativní odchylky mezi kodeky h264 a vp9 při referenční metrice h264, na obrázku 5.4 je pak to samé, pouze při referenční metrice vp9. Při referenčním kodeku h265 vypadá tento graf velmi podobně jako 5.3. Výrazné zhoršení kodeku vp9 u vyšších hodnot crf při použití referenčních kodeků h264 a h265 je způsobeno absencí malých souborů s nízkou hodnotou metriky SSIM v případě vp9. Začátek grafu dle kodeku h264 tedy přibližně odpovídá celému grafu dle vp9. Pro toto porovnání tedy použiji měření s referenčním kodekem vp9 s rozsahem crf od 4 do 51. Dle tohoto měření je ale patrné, že kodek vp9 potřebuje méně dat než kodek h264 při stejné kvalitě. Zároveň je také vidět, že při vyšší komprezji je náskok vp9 ještě lepší, stejně jako tomu bylo v případě kodeku h265.

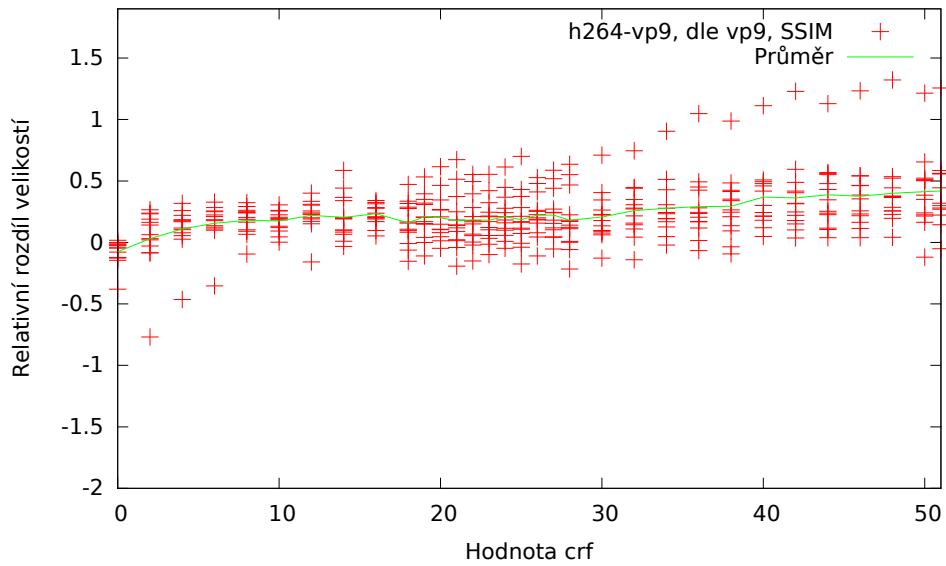
V případě kodeků h265 a vp9 se do jisté míry opakuje situace jako u kodeků h264 a vp9. Na obrázcích 5.5 a 5.6 jsou hodnoty relativních odchylek mezi kodeky h265 a vp9 při referenčních metrikách h265 a vp9. Graf při referenční

## 5. POROVNÁNÍ KODEKŮ

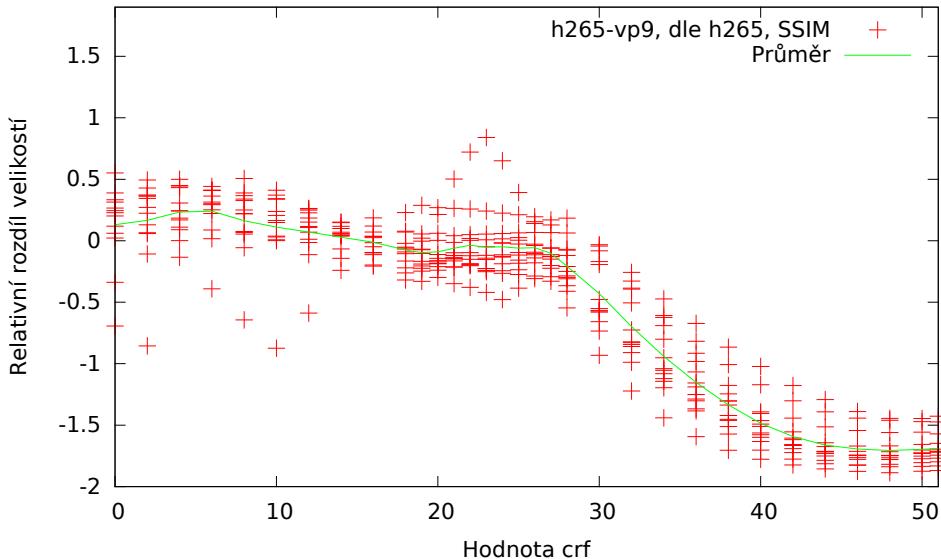
---



Obrázek 5.3: Rozdíl h264 a vp9, h264 jako referenční, metrika SSIM



Obrázek 5.4: Rozdíl h264 a vp9, vp9 jako referenční, metrika SSIM



Obrázek 5.5: Rozdíl h265 a vp9, h265 jako referenční, metrika SSIM

metrice h264 je na obrázku 5.7, je ale velmi podobný jako graf s referenční metrikou h265. Stejně jako u rozdílu h264 a vp9 použiju měření s referenčním kodekem vp9 s rozsahem crf od 4 do 51. Z tohoto grafu již není přímo patrné který z kodeků je lepší, h265 je lepší při vyšších kompresích, vp9 je lepší při nižších kompresích. Uživatelsky rozumné hodnoty crf jsou v případě h264 18 až 28 [11], což v případě kodeku vp9 přibližně odpovídá crf 23 až 51, pokud se tedy omezíme na tuto část, tak by pravděpodobně byl kodek h265 mírně lepší než vp9.

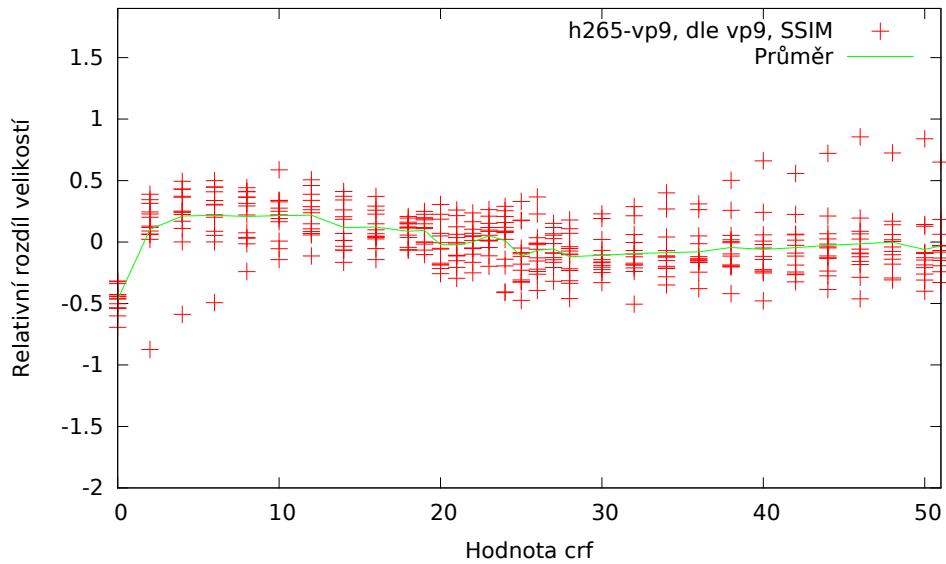
V tabulce 5.5 jsou shrnuté výsledky růzdílů mezi kodeky h264 a h265, jedná se o průměr měření při referenčních kodecích h264 a h265 s kvalitami crf v rozsahu 4 až 46. Průměrná relativní odchylka od průměru je 0.38. Z toho je možné vypočítat, že kodek h265 tedy při stejně kvalitě potřebuje o 55% méně dat než stejně kvalitní video s kodekem h264. Pokud se omezíme pouze na uživatelsky rozumné hodnoty, tak vyjde téměř identický výsledek, je potřeba o 54% méně dat ve prospěch kodeku h265.

Ve stejné tabulce jsou shrnuté i výsledky růzdílů mezi kodeky h264 a vp9, referenční kodek je vp9, z důvodů uvedených výše. Dle tohoto měření je kodek vp9 lepší, potřebuje o 40% méně dat než kodek h264, pokud se omezíme pouze na crf v rozsahu 23 až 51, pak je potřeba pouze 53% dat.

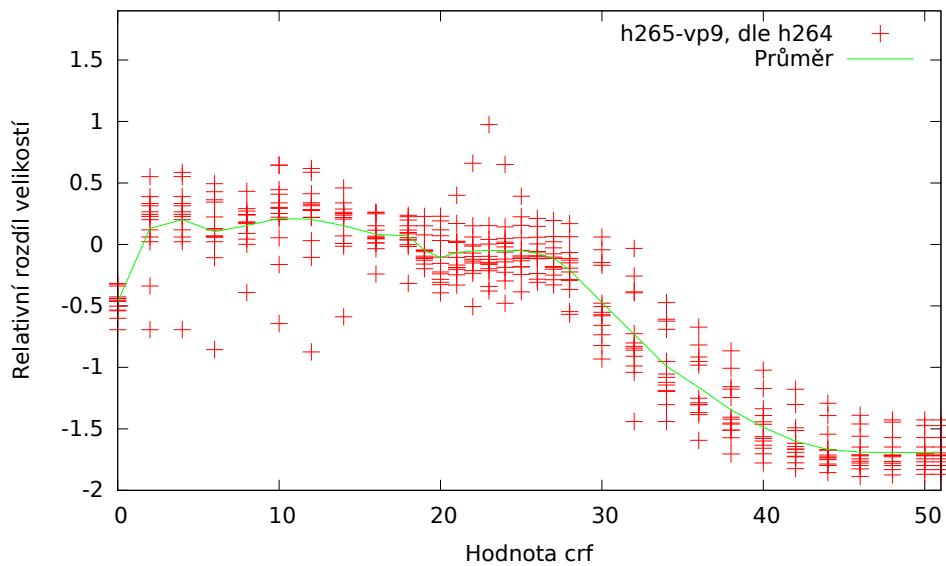
Zároveň se v tabulce zmíněné výše nacházejí shrnuté výsledky růzdílů mezi kodeky h265 a vp9, opět je jako referenční kodek použit vp9. Výsledky pro tyto kodeky jsou velmi vyrovnané, výraznější rozdíly ve prospěch vp9 při vyšší kvalitě jsou způsobeny částečně i rozdílem hodnotam metriky SSIM u porov-

## 5. POROVNÁNÍ KODEKŮ

---



Obrázek 5.6: Rozdíl h265 a vp9, vp9 jako referenční, metrika SSIM



Obrázek 5.7: Rozdíl h265 a vp9, h264 jako referenční, metrika SSIM

### 5.1. Vyhodnocení měření

---

crf	h264-h265		h264-vp9		h265-vp9	
4	-0.1007		0.10981		0.2111	
6	-0.0955		0.15612		0.21684	
8	-0.0691		0.17936		0.20847	
10	-0.0559		0.17477		0.21509	
12	-0.0113		0.21993		0.21822	
14	0.0554		0.20396		0.11856	
16	0.12293		0.23382		0.12171	
18	0.21436	0.21436	0.1595		0.08768	
19	0.28119	0.28119	0.20934		0.09493	
20	0.32386	0.32386	0.2012		-0.0223	
21	0.3391	0.3391	0.17571		-0.0234	
22	0.36484	0.36484	0.18378		0.00148	
23	0.36042	0.36042	0.18005	0.18005	0.05119	0.05119
24	0.41296	0.41296	0.21024	0.21024	0.01193	0.01193
25	0.41707	0.41707	0.18531	0.18531	-0.1157	-0.1157
26	0.43837	0.43837	0.22306	0.22306	-0.0668	-0.0668
27	0.44218	0.44218	0.22278	0.22278	-0.0565	-0.0565
28	0.4853	0.4853	0.18018	0.18018	-0.1202	-0.1202
30	0.52055		0.20667	0.20667	-0.1076	-0.1076
32	0.57061		0.25915	0.25915	-0.0946	-0.0946
34	0.60974		0.27806	0.27806	-0.0864	-0.0864
36	0.63549		0.2913	0.2913	-0.0804	-0.0804
38	0.68951		0.29408	0.29408	-0.0448	-0.0448
40	0.7556		0.36863	0.36863	-0.0603	-0.0603
42	0.82962		0.36167	0.36167	-0.0454	-0.0454
44	0.85343		0.38798	0.38798	-0.0268	-0.0268
46	0.82344		0.37528	0.37528	-0.0141	-0.0141
48			0.41467	0.41467	-0.0018	-0.0018
50			0.47669	0.47669	-0.0613	-0.0613
51			0.45777	0.45777	-0.0297	-0.0297
průměr	0.37827	0.3709	0.2527	0.29853	0.01664	-0.0527

Tabulka 5.5: Průměry relativních rozdílů mezi všemi kodeky

návaných verzí. Pokud použijeme plný rozsah crf, pak vycházejí metriky téměř identicky, pokud se omezíme na rozsah crf od 23, tak kodek h265 má o 10% nároky na velikost souboru při stejné kvalitě, dá se tedy říct, že kodek h265 je lepší, byť rozdíly jsou malé.

## 5. POROVNÁNÍ KODEKŮ

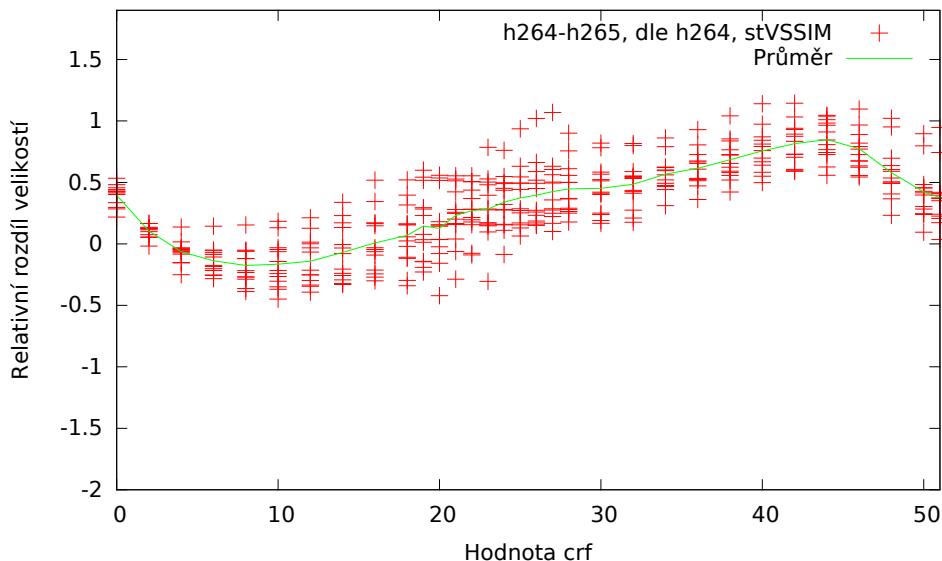
---

	h264-h265	h264-vp9	h265-vp9
stVSSIM	0.33765	0.21833	0.06037
stVSSIM, omezené crf	0.30831	0.29507	0.02371

Tabulka 5.6: Souhrné výsledky pro rozdíly mezi jednotlivými kodeky, metrika stVSSIM

### 5.1.2 stVSSIM

Zde budu porovnávat výše zmíněné kodeky pomocí metriky stVSSIM. Jak je vidět z obrázků 5.8, 5.9, 5.10, výsledky pro stVSSIM jsou podobné těm pro metriku SSIM, použil jsem také stejné referenční kodeky.

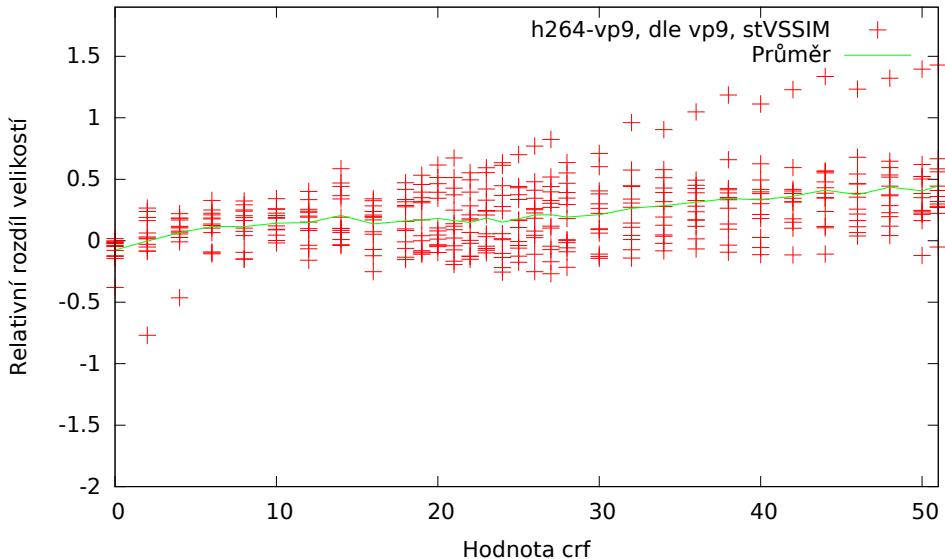


Obrázek 5.8: Rozdíl h264 a h265, h264 jako referenční, metrika stVSSIM

Souhrné výsledky pro stVSSIM jsou v tabulce 5.6, pod pojmem omezené crf se rozumí crf 18 až 28 v případě referenčních kodeků h264 a h265 a crf 23 až 51 pro referenční kodék vp9.

### 5.1.3 Testy ve 4K rozlišení

Pro toto testování jsem použil krátký film Tears of Steel [38] který je ke stažení i ve velmi kvalitní verzi 4K verzi ( $4096 \times 1710\text{px}$ )[39], která má datový tok videa přibližně 150Mbit/s. Toto video jsem zvolil záměrně z důvodu snahy o získání co nejméně ztrátového zdrojového souboru. Z důvodu velmi vysoké výpočetní náročnosti zpracování 4K videa jsem použil pouze deseti sekundový



Obrázek 5.9: Rozdíl h264 a vp9, vp9 jako referenční, metrika stVSSIM

	h264-h265	h264-vp9	h265-vp9
stVSSIM, omezené crf	0.401627536	0.250359253	-0.167526
SSIM, omezené crf	0.45468414	0.28014453	-0.177237
průměr, omezené crf	0.428155838	0.265251891	-0.172382

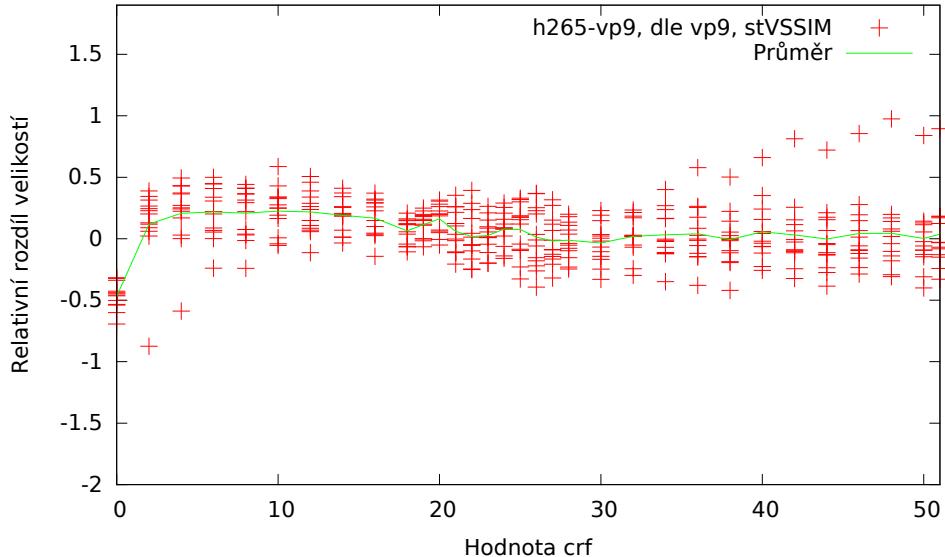
Tabulka 5.7: Výsledky při omezeném crf pro rozdíly mezi jednotlivými kodeky, 4K video

výřez z filmu začínající na desáté sekundě. Jako v předchozím případě jsem z tohoto videa vytvořil různá odvozená videa pomocí parametru crf. Opět jsem použil všechny 3 testované kodeky i stejné hodnoty crf jako v předchozím případě.

Výsledky pro 4K video jsou v tabulce 5.7, z této tabulky vyplývá, že oproti stavu ve full HD codec vp9 výrazně ztrácí na codec h265. Pokud se omezíme na uživatelsky rozumné hodnoty crf, pak vyjde, že h265 potřebuje při stejné kvalitě o 30% méně dat než codec vp9. Codec h264 potřebuje 2.44 krát tolik dat než h265 nebo 1.74 krát tolik dat jako codec vp9. Při 4K videu se tedy jasně vyplatí codec h265, následovaný codecem vp9, codec h264 již velmi výrazně ztrácí.

## 5. POROVNÁNÍ KODEKŮ

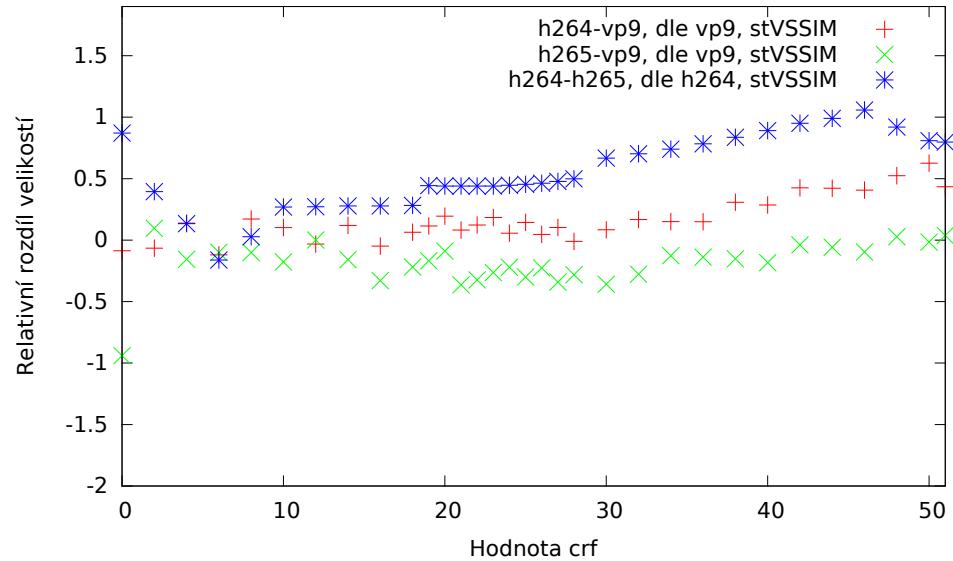
---



Obrázek 5.10: Rozdíl h265 a vp9, vp9 jako referenční, metrika stVSSIM

### 5.1.4 Souhrnné výsledky

Souhrnné výsledky pro metriky SSIM i stVSSIM včetně výsledků s omezeným crf a včetně výsledků 4K jsou v tabulce 5.8. Z této tabulky vyplývá, že kodek h264 je z testovaných kodeků nejhorší, kodeky h265 a vp9 jsou si ve full HD rozlišení kvalitou velmi podobné, záleží především na míře komprese a na zvolených videích. Kdybych zvolil h265 jako referenční video a omezil se na crf 18 až 28, mírně by vyhrál kodek h265 i v metrice stVSSIM. I z porovnání h264 oproti zbylým dvěma kodekům je možné říct, že tyto dva kodeky produkují podobně kvalitní videa při podobných velikostech, h265 ale vychází z tohoto srovnání o něco lépe. Dle grafů je h265 lepší při větší míře komprese a naopak. Není tedy možné určit jasného vítěze pro full HD videa mezi těmito dvěma kodeky. Oba dva kodeky ale značně převyšují kodek h264. Jak plyne z tabulky 5.8, při full HD videích potřebuje kodek h265 na dané kvalitě o 50% méně dat než kodek h264, kodek vp9 pak o 46% dat méně než h264. Při rozlišení 4K potřebuje kodek h265 o 30% méně dat než kodek vp9 a o 60% méně dat než h264. Kodek vp9 potřebuje o 43% méně dat než h264. Po zprůměrování dat mezi full HD a 4K videem je tedy nejlepší kodek h265, následovaný kodekem vp9 a kodek h264 je nejhorší.



Obrázek 5.11: Rozdíly mezi kodeky, metrika stVSSIM, 4K video

	h264-h265	h264-vp9	h265-vp9
full HD, stVSSIM	0.337647914	0.218325869	0.06037
full HD, SSIM	0.377188763	0.231796049	0.004191
full HD, průměr	0.357418338	0.225060959	0.032281
full HD, stVSSIM, omezené crf	0.308308023	0.295066582	0.023711
full HD, SSIM, omezené crf	0.370876748	0.298630213	-0.05273
full HD, průměr, omezené crf	0.339592386	0.296848397	-0.014509
4K, stVSSIM	0.525496037	0.16450191	-0.188867
4K, SSIM	0.542577393	0.175998171	-0.204031
4K, průměr	0.534036715	0.170250041	-0.196449
4K, stVSSIM, omezené crf	0.401627536	0.250359253	-0.167526
4K, SSIM, omezené crf	0.45468414	0.28014453	-0.177237
4K, průměr, omezené crf	0.428155838	0.265251891	-0.172382
průměr	0.445727527	0.1976555	-0.082084
průměr, omezené crf	0.383874112	0.281050144	-0.0934455

Tabulka 5.8: Souhrnné výsledky pro rozdíly mezi jednotlivými kodeky, full HD i 4K video



---

# Závěr

V úvodní, teoretické části této práce jsem popsal, jak je možné porovnávat kvalitu videa. Popsal jsem různé metriky na porovnávání kvality videa, včetně pravděpodobně nejčastěji používané a to SSIM. Také jsem popsal některé moderní formáty dnes velmi často používané pro ukládání HD obsahu, mimojiné i H.265, který je využíván v moderním televizním vysílání. Zmínil jsem i některé konkurenční programy, které se věnují stejnemu tématu.

Dále jsem vytvořil program pro porovnávání kvality různých verzí videa. Tento program implementuje metriky PSNR, SSIM a stVSSIM. Umožňuje i paralelní zpracování a podporuje i výpočty na grafické kartě (CUDA).

V další části jsem porovnával výsledky mnou implementovaných metrik se subjektivním měřením. Z mých měření vyplynulo, že všechny tři implementované metriky dobře korelují se subjektivním měřením. Nejlépe ovšem koreluje metrika stVSSIM. Na videích s chybami způsobenými kompresí má korelace dokonce lepší než 0.98. Dále jsem měřil časy výpočtu v CPU i CUDA verzi, algoritmus škáluje dobře s rostoucím počtem použitých vláken, ale zejména metriky SSIM a PSNR jsou omezeny rychlostí dekódování videa. CUDA verze algoritmu SSIM je také výrazně bržděna dekódováním videa. U CPU verze algoritmu stVSSIM je vidět velmi dobré škálování s rostoucím množstvím použitých procesorových jader je zde také výrazně vyšší výpočetní čas nutný pro tuto metriku. CUDA verze této metriky je bržděna výpočtem ARPS, který běží na CPU z důvodu nevyhnuteльného větvení v této části algoritmu. Pro efektivnější implementaci zbytku algoritmu na GPU by bylo potřeba extrémně velké množství paměti zařízení.

V neposlední řadě jsem srovnával různé video kodeky, jejich formáty konkrétně byly H.264, H.265 a VP9. Toto srovnání jsem dělal na videích s rozlišením full HD a 4K, na full HD videích byly formáty H.265 a VP9 shodně na prvním místě. na 4K videích již jasně vedl formát H.265 před VP9 a H.264. Pro použití na videu s rozlišením alespoň full HD ( $1920 \times 1080px$ ) je tedy možné z hlediska jejich kvality doporučit formát H.265. Oproti staršímu formátu H.264 zde došlo přibližně k 50% úspoře dat při zachování kvality.

## ZÁVĚR

---

Tato práce je využitelná pro srovnání kvality videí, je možné pomocí ní určit vhodné parametry při konverzi/komprese videa. Také může sloužit pro porovnání výkonosti různých kodeků za cílem určení, který z nich potřebuje nejméně dat při dané kvalitě.

---

## Literatura

- [1] Anish Mittal, A. C. B., Anush Krishna Moorthy: No-Reference Image Quality Assessment in the Spatial Domain. Prosinec 2012, [cit. 2016-10-28]. Dostupné z: <http://live.ece.utexas.edu/publications/2012/TIP%20BRISQUE.pdf>
- [2] Nisha, S. K.: Image Quality Assessment Technique. Červenec 2013, [cit. 2016-10-28]. Dostupné z: [https://www.ijarcsse.com/docs/papers/Volume\\_3/7\\_July2013/V3I7-0279.pdf](https://www.ijarcsse.com/docs/papers/Volume_3/7_July2013/V3I7-0279.pdf)
- [3] The FFmpeg developers: *FFmpeg Formats Documentation*. Prosinec 2016, [cit. 2016-12-11]. Dostupné z: <https://www.ffmpeg.org/ffmpeg-formats.html#toc-rawvideo>
- [4] The FFmpeg developers: *FFmpeg*. Prosinec 2016, [cit. 2016-12-11]. Dostupné z: <https://www.ffmpeg.org/>
- [5] Microsoft: YUV Video Subtypes. 2016, [cit. 2016-12-11]. Dostupné z: [https://msdn.microsoft.com/en-us/library/windows/desktop/dd391027\(v=vs.85\).aspx](https://msdn.microsoft.com/en-us/library/windows/desktop/dd391027(v=vs.85).aspx)
- [6] Reichl, J.; Všetička, M.: *Fotometrické veličiny*. 2013, [cit. 2017-01-06]. Dostupné z: <http://fyzika.jreichl.com/index.php/main.article/view/535-fotometricke-veliciny>
- [7] H264info.com: What is H.264. 2010, [cit. 2016-12-11]. Dostupné z: <http://www.h264info.com/h264.html>
- [8] Sullivan, G. J.; Ohm, J.-R.; Han, W.-J.; aj.: Overview of the High Efficiency Video Coding (HEVC) Standard. Prosinec 2012, [cit. 2016-12-11]. Dostupné z: <http://ieeexplore.ieee.org/stamp/stamp.jsp?reload=true&tp=&arnumber=6316136>

## LITERATURA

---

- [9] ČESKÉ RADIOKOMUNIKACE A.S.: DVB-T2 Digitální TV vysílání budoucnosti. 2016, [cit. 2016-12-11]. Dostupné z: <https://www.cra.cz/dvb-t2>
- [10] Deveria, A.: *Can I use WebM?* Prosinec 2016, [cit. 2016-12-11]. Dostupné z: <http://caniuse.com/#search=WebM>
- [11] The FFmpeg developers: *FFmpeg and H.264 Encoding Guide*. Prosinec 2016, [cit. 2016-12-22]. Dostupné z: <https://trac.ffmpeg.org/wiki/Encode/H.264>
- [12] Peak Signal-to-Noise Ratio as an Image Quality Metric. Září 2013, [cit. 2016-06-25]. Dostupné z: <http://www.ni.com/white-paper/13306/en/>
- [13] Yalman, Y.; Ertur, I.: A new color image quality measure based on YUV transformation and PSNR for human vision system. Listopad 2011, [cit. 2016-06-25]. Dostupné z: <http://journals.tubitak.gov.tr/elektrik/issues/elk-13-21-2/elk-21-2-20-1111-11.pdf>
- [14] Colucci, E.: Image and Video quality assessment – Part One: MSE & PSNR. Duben 2011, [cit. 2016-06-26]. Dostupné z: <http://emanuelecolucci.com/2011/04/image-and-video-quality-assessment-part-one-mse-psnr/>
- [15] Veldhuizen, T.: Measures of image quality. Leden 1998, [cit. 2016-06-26]. Dostupné z: [http://homepages.inf.ed.ac.uk/rbf/CVonline/LOCAL\\_COPIES/VELDHUIZEN/node18.html](http://homepages.inf.ed.ac.uk/rbf/CVonline/LOCAL_COPIES/VELDHUIZEN/node18.html)
- [16] Wang, Z.; Bovik, A. C.: Image Quality Assessment: From Error Visibility to Structural Similarity. Květen 2004, [cit. 2016-07-01]. Dostupné z: <http://www.cns.nyu.edu/pub/lcv/wang03-preprint.pdf>
- [17] Wang, Z.; Simoncelli, E. P.; Bovik, A. C.: MULTI-SCALE STRUCTURAL SIMILARITY FOR IMAGE QUALITY ASSESSMENT. Listopad 2002, [cit. 2016-07-20]. Dostupné z: <http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=1292216>
- [18] Moorthy, A. K.; Bovik, A. C.: Efficient Motion Weighted Spatio-Temporal Video SSIM Index. 2010, [cit. 2016-10-08]. Dostupné z: [http://live.ece.utexas.edu/publications/2010/moorthy\\_spie\\_jan10.pdf](http://live.ece.utexas.edu/publications/2010/moorthy_spie_jan10.pdf)
- [19] Seshadrinathan, K.; Bovik, A. C.: Motion-based Perceptual Quality Assessment of Video. [cit. 2016-10-08]. Dostupné z: <https://pdfs.semanticscholar.org/a272/ed5819b70d9987d2c436bd4e65ef1222293f.pdf>

- [20] Wang, Z.; Simoncelli, E. P.; Bovik, A. C.: MULTI-SCALE STRUCTURAL SIMILARITY FOR IMAGE QUALITY ASSESSMENT. [cit. 2016-10-08]. Dostupné z: <https://ece.uwaterloo.ca/~z70wang/publications/msssim.pdf>
- [21] Nie, Y.; Ma, K.: Adaptive Rood Pattern Search for Fast Block-Matching Motion Estimation. Prosinec 2002, [cit. 2016-10-11]. Dostupné z: <https://pdfs.semanticscholar.org/7f78/41b7b9c98b1e1f52282bdc3c2710cf83d3f0.pdf>
- [22] The International Telecommunication Union - Telecommunication Study Group 16: ITU-T H.263. Leden 2005, [cit. 2016-10-12]. Dostupné z: [https://www.itu.int/rec/dologin\\_pub.asp?lang=e&id=T-REC-H.263-200501-I!!PDF-E&type=items](https://www.itu.int/rec/dologin_pub.asp?lang=e&id=T-REC-H.263-200501-I!!PDF-E&type=items)
- [23] OpenMP: The OpenMP API specification for parallel programming. 2017, [cit. 2017-01-06,]. Dostupné z: <http://www.openmp.org/>
- [24] NVIDIA Corporation: WHAT IS CUDA? 2017, [cit. 2017-01-06]. Dostupné z: [http://www.nvidia.com/object/cuda\\_home\\_new.html](http://www.nvidia.com/object/cuda_home_new.html)
- [25] NVIDIA Corporation: WHAT IS CUDA? 2017, [cit. 2017-01-06]. Dostupné z: <http://www.cplusplus.com/info-description/>
- [26] Vatolin, D.; Smirno, M.: Everything about the data compression. 2016, [cit. 2017-01-06]. Dostupné z: [http://www.compression.ru/index\\_en.htm](http://www.compression.ru/index_en.htm)
- [27] Kulikov, D.; Erofeev, M.; Dolganov, S.; aj.: HEVC Video Codecs Comparison. 2016, [cit. 2017-01-07]. Dostupné z: [http://www.compression.ru/video/codec\\_comparison/hevc\\_2016/](http://www.compression.ru/video/codec_comparison/hevc_2016/)
- [28] Hanhart, P.: *VQMT: Video Quality Measurement Tool*. 2013, [cit. 2017-01-08]. Dostupné z: <http://mmspgr.epl.ch/vqmt>
- [29] Itseez: *OpenCV (Open source Computer Vision)*. 2017, [cit. 2017-01-08]. Dostupné z: <http://opencv.org/>
- [30] The FFmpeg developers: *FFmpeg*. Prosinec 2016, [cit. 2016-12-18]. Dostupné z: <https://ffmpeg.org/download.html#build-windows>
- [31] The FFmpeg developers: *FFmpeg*. Prosinec 2016, [cit. 2016-12-18]. Dostupné z: <https://ffmpeg.org/download.html#build-linux>
- [32] Oracle Corporation: *The Producer/Consumer Problem*. 2010, [cit. 2016-12-20]. Dostupné z: <https://docs.oracle.com/cd/E19455-01/806-5257/sync-31/index.html>

## LITERATURA

---

- [33] Webster, A.; Speranza, F.: *Video Quality Experts Group (VQEG) HDTV*. 2010, [cit. 2016-12-27]. Dostupné z: <http://www.its.bldrdoc.gov/vqeg/projects/hdtv/hdtv.aspx>
- [34] Institute for Telecommunication Sciences: *The Consumer Digital Video Library*. 2013, [cit. 2016-12-22]. Dostupné z: <http://www.cdvl.org/find-videos/find-videos.php>
- [35] Webster, A.; Speranza, F.: *VQEG Final Report of HDTV Validation Test*. 2010, [cit. 2016-12-27]. Dostupné z: [http://www.its.bldrdoc.gov/media/4212/vqeg\\_hdtv\\_final\\_report\\_version\\_2.0.zip](http://www.its.bldrdoc.gov/media/4212/vqeg_hdtv_final_report_version_2.0.zip)
- [36] *Marvel's Avengers: Age of Ultron*. 2016, [cit. 2016-12-21]. Dostupné z: <http://www.hd-trailers.net/movie/avengers-age-of-ultron/>
- [37] Giles, M.: Warp divergence. 2016, [cit. 2017-01-08]. Dostupné z: <https://people.maths.ox.ac.uk/gilesm/cuda/lecs/lec3-2x2.pdf>
- [38] Roosendaal, T.: *Tears of Steel*. 2012, [cit. 2017-01-05]. Dostupné z: <https://mango.blender.org/about/>
- [39] Roosendaal, T.: *Tears of Steel, stažení 4K verze*. 2013, [cit. 2017-01-05]. Dostupné z: [http://download.blender.org/demo/movies/ToS/ToS\\_4k\\_DCP.zip](http://download.blender.org/demo/movies/ToS/ToS_4k_DCP.zip)

## **Seznam použitých zkratek**

- VQEG Video Quality Experts Group  
fps Frames per second  
CRF Constant Rate Factor  
PSNR Peak signal-to-noise ratio  
SSIM Structural similarity  
stVSSIM Spatio-Temporal Video SSIM  
CUDA Compute Unified Device Architecture



## Obsah přiloženého CD

```
readme.txt.....stručný popis obsahu CD
├── data.....adresář se testovacími videi
├── exe .....adresář se spustitelnou formou implementace
└── src
    ├── src .....zdrojové kódy implementace
    └── thesis .....zdrojová forma práce ve formátu LATEX
└── text .....text práce
    ├── thesis.pdf .....text práce ve formátu PDF
    └── thesis.ps .....text práce ve formátu PS
```