Sem vložte zadání Vaší práce.

České vysoké učení technické v Praze
Fakulta informačních technologií
Počítačové systémy a sítě

Diplomová práce

GPU akcelerované vyhodnocení kvality videa

Bc. Václav Fanfule

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

2. listopadu 2016

Poděkování

Děkuji vedoucímu práce Ing. Ivanovi Šimečkovi, Ph.D. za vedení, rady a pomoc při tvorbě této práce. Dále také děkuji rodině za podporu 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 2. listopadu 2016 .....

Odkaz na tuto práci

Fanfule, Václav. *GPU akcelerované vyhodnocení kvality videa.* Diplomová práce. Praha: České vysoké učení technické v Praze, Fakulta informačních technologií, 2016.

Abstrakt

V několika větách shrňte obsah a přínos této práce v češtině. Po přečtení abstraktu by se čtenář měl mít čtenář dost informací pro rozhodnutí, zda chce Vaši práci číst.

Klíčová slova Nahraďte seznamem klíčových slov v češtině oddělených čárkou.

Abstract

Sem doplňte ekvivalent abstraktu Vaší práce v angličtině.

Keywords Nahraďte seznamem klíčových slov v angličtině oddělených čárkou.

Obsah

Seznam obrázků

Úvod

Kapitola 1

Technologie

V této kapitole stručně představím technologie použité v této práci. Nejprve stručně představím možností porovnávání video snímků poté se zaměřím na metriky které budu v této práci používat.

1.1 Metriky pro porovnávání videa

Pod pojmem metrika se v této práci 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 ale vycházejí různé metriky a proto nejsou mezi sebou porovnatelné. Například u metriky PSNR platí, že čím jsou objekty shodnější, tím metrika je vyšší a naopak. Ale tato metrika nemá lineární průběh, průběh je spíše exponenciální, pro shodné objekty vychází plus nekonečno, pro zcela neshodné objekty vychází 0.

V této části se budu zabývat problematikou jak porovávat kvalitu videa. 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 srovávaná videa. U nereferenčních metrik není k dispozici žádné ukázkové video a jsou porovnávána pouze videa mezi sebou bez znalosti toho 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 porování bitratu. Tato metrika je ale velmi nedostatečná protože nezohladňuje mnoho faktorů, například různé kodeky pro enkódování. Další mož-

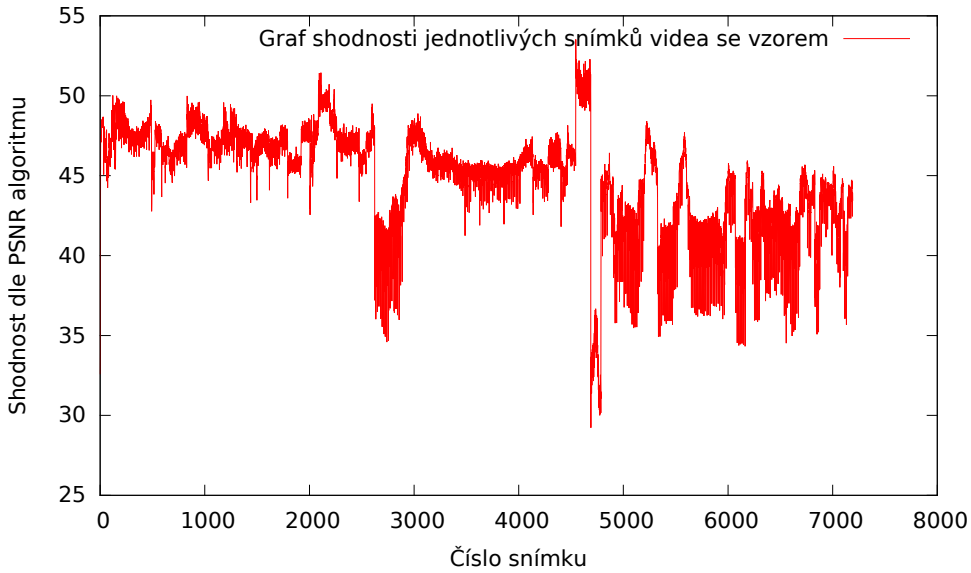
ností je porovnávání jednotlivých snímků videa a zkoumání jejich shodnosti. Tato metrika je znatelně spolehlivější a přenáší problém na to jak porovávat jednotlivé snímky dvou videí mezi sebou. Na tuto problematiku existuje mnoho algoritmů od jednoduchého součtu odchylek barev jednotlivých pixelů až po komplexní algoritmy které by měly zahrnovat i nedokonalé vnímání lidským okem. V této práci jsem zvolil z této oblasti algoritmy PSNR a SSIM. Jedné se pravděpodobně o nejčastěji používané algoritmy pro porovávání obrázků. Mezi další metriky se řadí například MSE nebo MSSIM, jedná se ale o algoritmy odvozené z PSNR nebo SSIM anebo o algoritmy podobné[2].

Při použití metrik určených pro porovávání 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 dostatečně 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 níže. 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)linearitu 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 porovávání vidoesekvencí jako celku, nikoliv pouze porovávání 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 takovéhoto algoritmu je stVSSIM, který budu v této práci také implementovat. Tento algoritmus vychází z algoritmu SSIM ale zohledňuje i předchozí a následující snímky při porovávání. 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 ale nespadá pod téma této práce a tudíž se v této práci nebudu zabývat subjektivním porovnáváním.

Obrázek 1.1: Teoretický ideální Graf kvalit jednotlivých snímků ve videu



Obrázek 1.2: Graf kvalit jednotlivých snímků ve videu

1.2 PSNR

Peak signal-to-noise ratio (PSNR) do češtiny přeloženo jako špičkový poměr signálu k šumu[3]. Jedná se o poměr maximálního možného množství informace obsažené v signálu (v našem případě obrázku) ku množství šumu který ovlivňuje kvalitu tohoto signálu. Díky typicky velkému dynamickému rozsahu množství informace v signálu se PSNR udává v logaritmické škále a má jednodku dB.

Předpoklad pro PSNR je, že máme 2 obrázky které mají stejné rozlišení a jsou uloženy ve 2D matici, indexované od $0, 0$, po jednotlivých pixelech.
Poté platí matematický zápis:

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

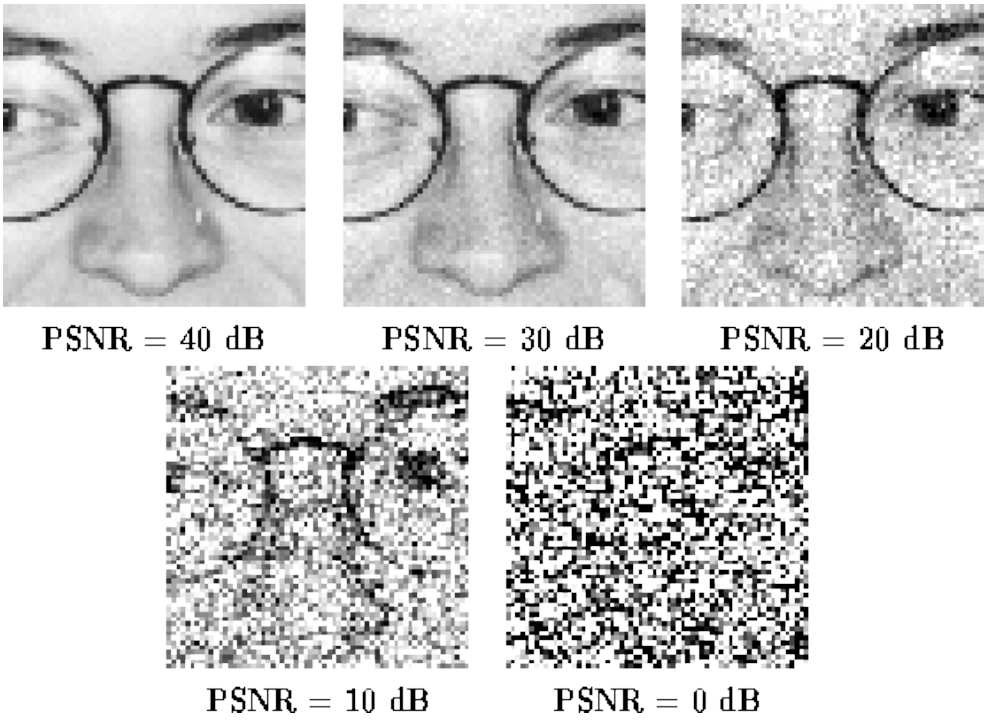
Kde platí, že *MSE* (Mean Squared Error) - průměrná čtvercová odchylka je definována jako:

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

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_I^2 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[4].

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 $50dB$[5] ovšem tato hodnota závisí na obrázku a není možné takto porovávat různé obrázky, je možné porovnávat pouze stejný obrázek v různých kvalitách[6]. Na obrázku 1.3 je vidět ilustrace různých hodnot PSNR pro daný obrázek.

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 porovnání barevných obrázků je možné buď daný obrázek převést do jedné barevné složky (černobílý obraz), nebo počítat průměr ze všech barevných složek a nebo převést obrázek do barevného modelu který má jasovou složku a počítat PSNR pouze



Obrázek 1.3: Srovnání obrázku v různých hodnotách PSNR

na jasové složce. Poslední zmíněný přístup se používá z důvodu vyšší citlivosti lidského oka na jasovou složku oproti ostatním složkám.

1.3 SSIM

Výpočet SSIM je prováděn nad jasovou složkou obrazu. Základní vzorec pro výpočet SSIM je 1.3.

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

Kdy musí platit, že $\alpha > 0$, $\beta > 0$ a $\lambda > 0$, v referenční verzi se používá $\alpha = \beta = \gamma = 1$[7]. Takto je výpočet SSIM rozdělen do 3 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

$$\mu_x = \frac{1}{N}\sum_{i=1}^{N} x_i \tag{1.4}$$

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

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

Následně je odečten průměrný jas složka z obrazu a za pomoci vzorce 1.6 je vypočítána standardní odchylka jednotlivých snímků.

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

Vzorec 1.7 vyjadřuje rozdíl kontrastu dvou snímků.

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

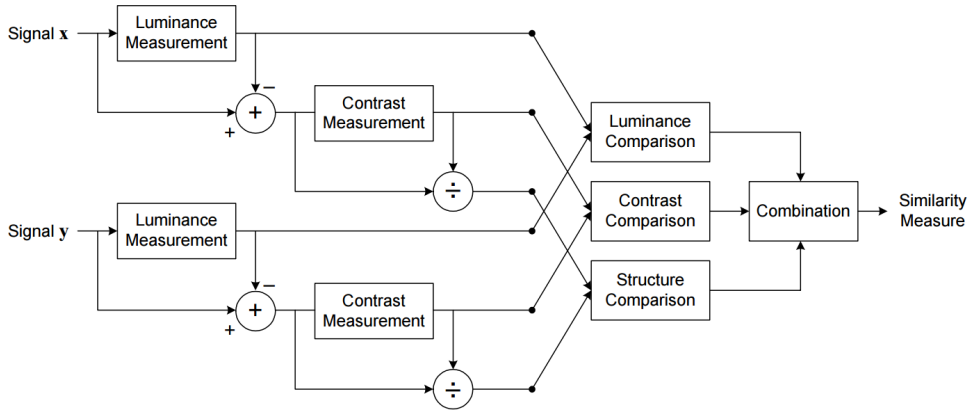
Následně je signál (snímek) znormalizován (vydělen) jeho vlastní standardní odchylkou. Díky tomu mají oba snímky stejnou standardní odchylku. Porovnávání struktury signálů je tedy prováděna na normalizovaném signálu $(x - \mu_x)/\sigma_x$. Korelace těchto signálů je vyjádřena pomocí vzorce 1.8

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

Z toho σ_{xy} se vypočítá dle 1.9.

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

Kde μ_x and μ_y představují průměr z originálního a testovaného obrázku. σ_x a σ_y představují standardní odchylku. σ_x^2 a σ_y^2 představují varianci. σ_{xy} představuje kovariance těchto dvou obrázků. Na obrázku 1.4 je možné vidět jednotlivé kroky procesu tvorby SSIM indexu.



Obrázek 1.4: Diagram procesu tvorby SSIM

Při použití hodnot $\alpha = \beta = \gamma = 1$ a $C_3 = C_2/2$ [7] vyjde zjednodušený vzorec 1.10.

$$SSIM(x, y) = \frac{(2\mu_x\mu_y + C1)(2\sigma_{xy} + C2)}{(\mu_x^2 + \mu_y^2 + C1)(\sigma_x^2 + \sigma_y^2 + C2} \tag{1.10}$$

C_1, C_2 a C_3 jsou použity pro eliminaci nestability výsledků když některá z hodnot $\mu_x^2 + \mu_y^2$ nebo $\sigma_x^2 + \sigma_y$ 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 $C_1 = (K_1L)^2$ respektive $C_2 = (K_2L)^2$ 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 $K_2 = 0.03$ [8].

Hodnoty μ_x, σ_x a σ_{xy} se počítají na blocích 8×8 pixelů, toto okno se postupně posouvá po celém obrazu, je možné posouvat toto okno po 1 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 8×8 okna. Pro zvýšení přesnosti tohoto algoritmu je možné použít vážený průměr nebo heuristiku zohledňující bloky s horším výsledkem[9].

SSIM má oproti PSNR výhodu v tom, že zohledňuje vnímání obrazu lidským okem a nepočítá pouze absolutní odchylku od výchozího zdroje, například lidské oko je citilivější na jasovou složku obrázku oproti jiným složkám. V SSIM je také využito maskování postupně průměrného jasu a kontrastu pro získání nezávislých výsledků při počítání kontrastu a struktury. 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 porovávání snímků a nikoliv videí, tudíž pro mé využití je potřeba tento algoritmus rozšířit aby byl použitelný i na video.

1.4 stVSSIM

!!když bude málo textu, dá se psát nějaká teorie o MOVIE!! FIXME Algoritmus spatio-temporal video SSIM[9] je další z algoritmů na porovnávání kvality videí. Narozdíl od algoritmů SSIM i PSNR, které porovnávají pouze jednotlivé snímky mezi sebou, stVSSIM algoritmus zohledňuje i okolní snímky videa. Jedná se o full reference metriku, stejně jako PSNR a SSIM. Inspirací pro stVSSIM byl Motion-based Video Integrity Evaluation(MOVIE) index[10]. MOVIE index se ale ukázal být sice 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. stVSSIM se snaží o zachování kvality algoritmu MOVIE při snížení potřebného výpočetního výkonu.

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[11]. Vyhodnocování kvality v časové rovině je dosaženo rozšířením SSIM o časoprostorovou doménu, kterou autoři této metriky nazývají SSIM-3D.

1.4.1 SSIM-3D

SSIM je počítáno z jednoho snímku po blocích dané velikosti(typicky 8×8 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 rámsci daného obrazu a k značí číslo snímku. V tomto případě nás při každém výpočtu zajímá blok pixelů v okolí pixelu (i, j, k). Definujme α, β jako rozměry v prostoru a γ 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. při každém vyhodnocovaném snímku 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:

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

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

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

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

a

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

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

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

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

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

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

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

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

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

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

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

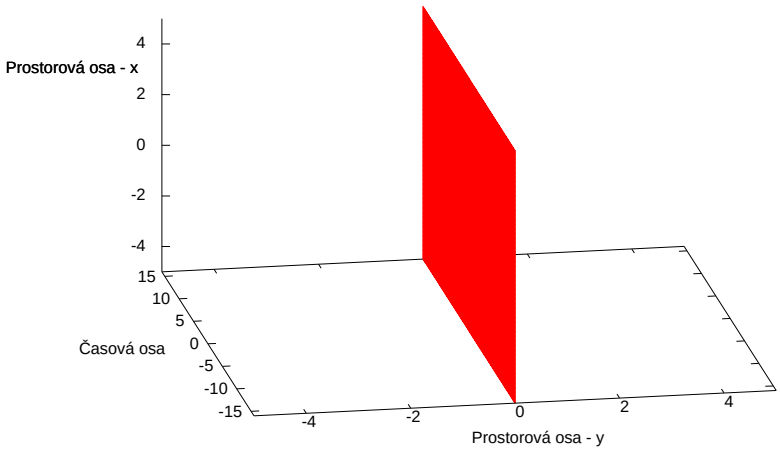
$$\text{SSIM-3D} = \frac{(2\mu_{x(i,j,k)}\mu_{y(i,j,k)} + C1)(2\sigma_{x(i,j,k)y(i,j,k)} + C2)}{(\mu^2_{x(i,j,k)} + \mu^2_{y(i,j,k)} + C1)(\sigma^2_{x(i,j,k)} + \sigma^2_{y(i,j,k)} + C2} \tag{1.16}$$

Konstanty $C1$ a $C2$ jsou stejné jako v případě SSIM. Váhová funkce w záleží na typu filtru který je použit. V případě SSIM-3D jsou použity 4 různé filtry, viz obrázky 1.5, 1.6, 1.8, 1.7. $w(m, n, o)$ nabývá hodnoty 1 pokud bod (m, n, o) leží v rovině daného filtru, v opačném případě nabývá hodnoty 0. m a n značí horizontální a vertikální pozici od výchozího pixelu z bloku pixelů. Záporné hodnoty značí pixely vlevo od tohoto pixelu a naopak. o značí číslo snímku od výchozího snímku. Záporné hodnoty o značí předchozí snímky a naopak. V referenčním publikaci jsou spočítány SSIM-3D hotnoty pro všechny 4 filtry pro každý (i,j,k) pixel a následně je jich pří 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 8×8. Použití bloku 8×8 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 zkusit změnit velikost bloku. Pro výpočet pohybového vektoru je použit algoritumus Adaptive Rood Pattern Search (ARPS)[12].

1.4.2 ARPS

1.4.2.1 Popis algoritmu

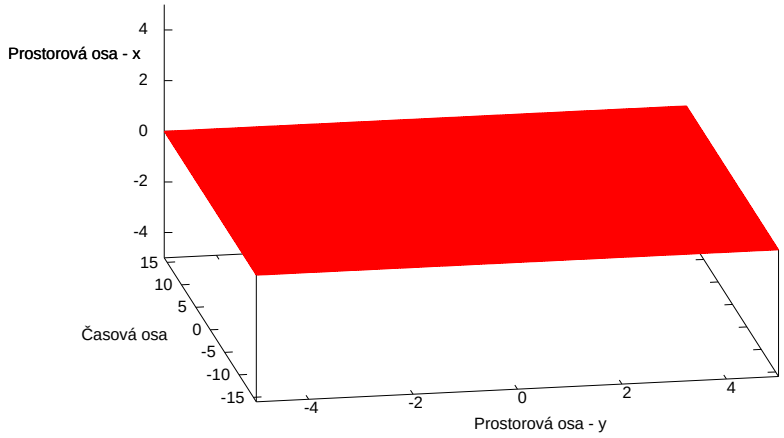
Adaptive Rood Pattern Search (ARPS)[12] je algoritmus, který se zabývá hledáním pohybových vektorů ve videu. Pohybový vektor (motion vector) je způsob reprezentace bloku v daném snímku na základě stejného (nebo podobného) bloku v předchozím snímku. Pro nalezení pohybového vektoru je potřeba nalézt v předchozím snímku co blok co nejpodobnější aktuálnímu bloku. Při



Obrázek 1.5: Vertikální filtr

videokompresi 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 kompresi videí. Hledání pohybových vektorů je časově nejnáročnější akce při kompresi videa a proto může tvořit velké omezení při kompresi/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 bloků budeme nazývat předpovídaný 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 zmíněném 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 kodeku H263[13]. Předpovídaný 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 složitejší metody, ty ale vzhledem k zanedbatelnému přínosu v přesnosnosti a rychlosti nemá smysl používat. Dle [12] je velmi malý rozdíl v přesnosti a vý-



Obrázek 1.6: Horizontální filtr

konu při používání různých podporujících sousedních bloků, z toho důvodu byl použit typ D.

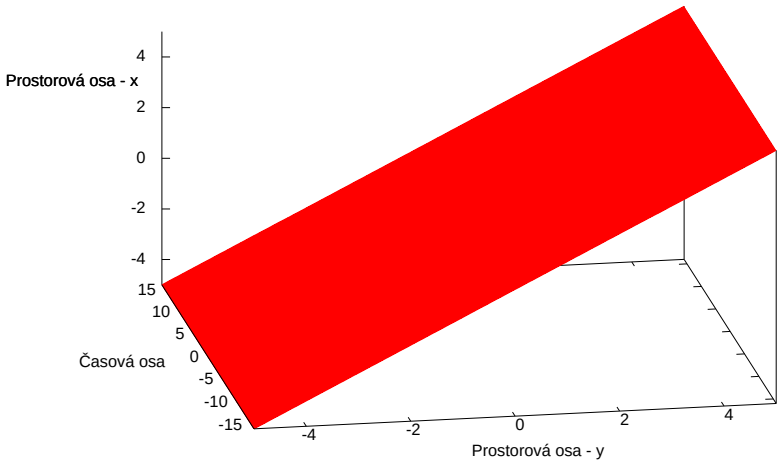
Algoritmus ARPS se snaží najít vzor aktuálního bloku dle kříže, jak je znázorněno na obrázku 1.11. Na tomto obrázku je vidět jak využití pohybového vektoru z předchozího snímku, který byl $(2, -1)$, tak využití 4 okolních bodů a středu. Vzdálenost okolních bodů od středu (Γ) může být dána například rovnicí 1.17. Alternativně je možné použít rovnici 1.18. Protože druhá zmíněná rovnice nevyžaduje ani zaokrouhlování, ani umocňování a odmocňování, je mnohem méně výpočetně náročná. Protože v praxi nepřináší první rovnice lepší přesnost, bude použita rovnice 1.18. Pokud není dostupný předchozí blok, tak bude použita vzdálenost 2 pixely, tedy $\Gamma = 2$.

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

$$\Gamma = Max(|MV_{predicted}(x)|, |MV_{predicted}(x)|) \tag{1.18}$$

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

Další vlastností algoritmu ARPS je takzvaný předsudek 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



Obrázek 1.7: Pravý filtr

omezené části obrazu. Algoritmus tedy v prvním kroku spočítá sumu absolutních rozdílů oproti stejně umístěnému bloku v předchozích snímků 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, čili při velikosti 8×8 je možné stanovit hranici na $T = 128$. Tuto hranici je možné posunout, čim výše je tato hranice nasazena, tím lepší dosáhneme zrychlení tohoto algoritmu, ale ztrácíme přesnost výpočtu.

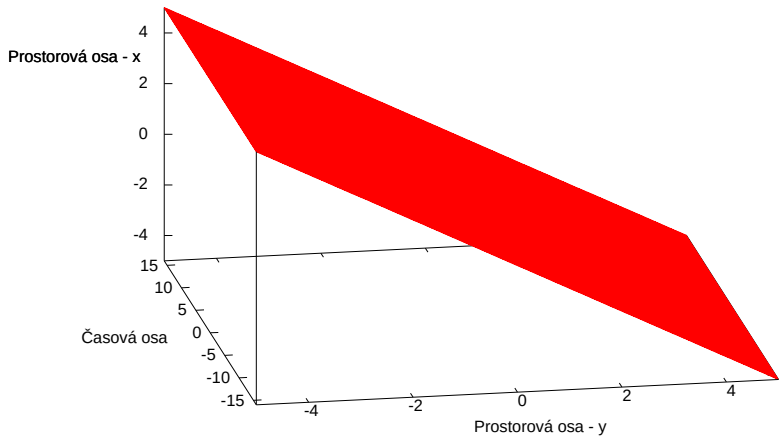
1.4.2.2 Výpočet ARPS

Jak bylo již zmíněno dříve, budeme používat D typ podporujícího sousedního bloku dle obrázku 1.9. Pro všechny nejlevější 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 $\Gamma = Max(|MV_{predicted}(x)|, |MV_{predicted}(x)|)$

3. Nastavíme střed kříže, který je znázorněn na obrázku 1.11, na stejné souřadnice jako je aktuální blok.



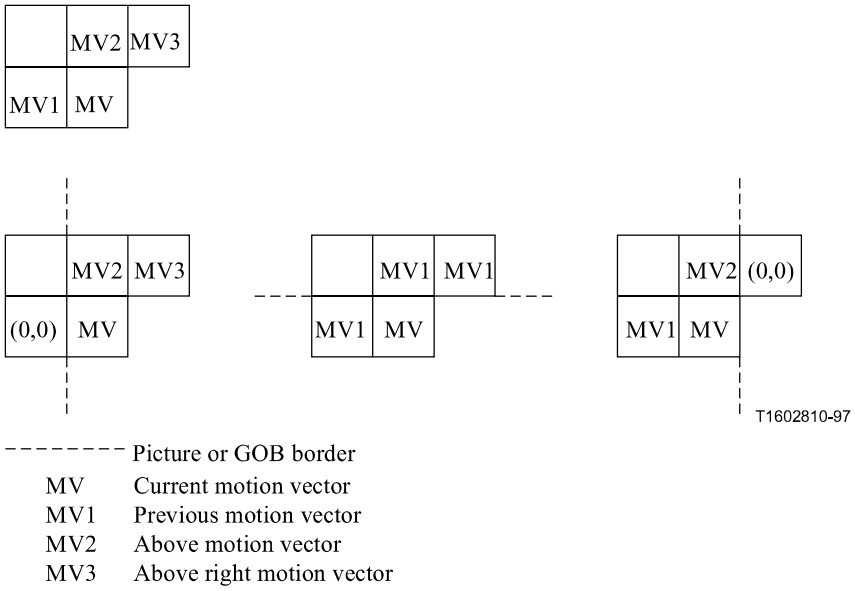
Obrázek 1.8: Vertikální filtr

Obrázek 1.9: Typy podporujících sousedních bloků

4. Spočítáme sumy absolutních rozdílů pro bloky se středy danými těmito body. Pokud nám vyjde střed jako bod s nejnižší sumou rozdílů, tak 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 náročností je vhodné implementovat nějakým způsobem identifikování, zda-li už byla spočítána odchylka daného bodu, například pomocí bit mapy. Takto se postupně dostaneme do bodu, který budeme používat jako vzor pro náš aktuální blok.



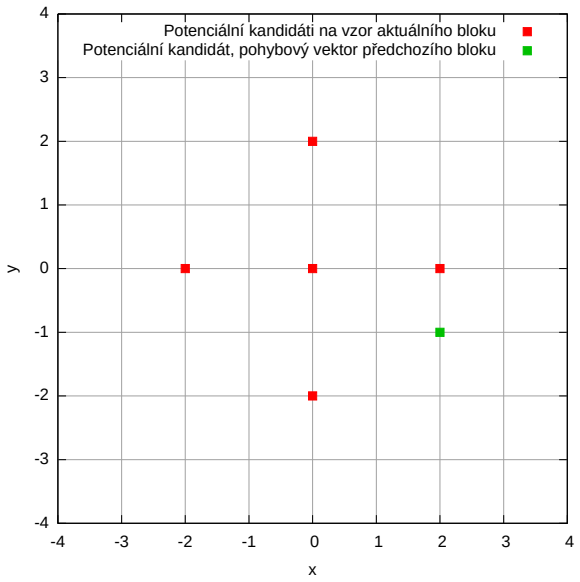
Obrázek 1.10: Podporující sousední bloky u okrajů

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 filtrů 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 desetinými čísly. Pokud je vzniklý pohybový vektor přesně mezi dvěmi filtry, je použit průměr z těchto 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 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 se počítá SSIM i SSIM-3D pouze z každého šestnáctého nsímku. Výsledný index stVSSIM je



Obrázek 1.11: Graf potenciálních kandidátů na vzor aktuálního bloku

následně spočítán jako 1.19.

$$stVSSIM = T_{video} \times S_{video} \tag{1.19}$$

Kapitola 2

Analýza

2.1 Využití pro video

2.2 Srování s konkurencí

Kapitola 3

Implementace

3.1 PSNR

3.2 SSIM

3.3 stVSSIM

Kapitola 4

Měření

http://4ksamples.com/category/4k/movies/ http://www.hd-trailers.net/movie/elysium/ $ffmpeg -ih264_fHD_orig.mp4 -c : vh264 -c : acopy -b : v5000000 -vfscale = 1920 : -1h264_fHD_5000Kb.mkv ffmpeg - ih264_fHD_orig.mp4 - c : vvp9 - c : acopy - b : v5000000 - vfscale = 1920 : -1vp9_fHD_5000Kb.mkv ffmpeg - ih264_fHD_orig.mp4 - c : vhevc - c : acopy - b : v5000000 - vfscale = 1920 : -1h265_fHD_5000Kb.mkv ffmpeg -ih264_fHD_orig.mp4 -c : vmpeg2video -c : acopy - b : v5000000 - vfscale = 1920 : -1mpeg2_fHD_5000Kb.mkv ffmpeg - ih264_fHD_orig.mp4 - c : vmpeg4 - c : acopy - b : v5000000 - vfscale = 1920 : -1mpeg4_fHD_5000Kb.mkv$

4.1 Srovnání rychlosti metrik

4.2 Výkonnost CPU

4.3 Výkonnost GPU

Kapitola 5

Porovnávání kodeků

Závěr

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

[3] 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/

[4] YALMAN, Y.; ERTURK, 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

[5] 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/

[6] 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

[7] 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

[8] Wang, Z.; Simoncelli, E. P.; Bovik, A. C.: MULTI-SCALE STRUCTURAL SIMILARITY FOR IMAGE QUALITY ASSESSMENT. Listo-

pad 2002, [cit. 2016-07-20]. Dostupné z: http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=1292216

[9] 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

[10] 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

[11] 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

[12] 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

[13] 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

Příloha **A**

Seznam použitých zkratek

GUI Graphical user interface

XML Extensible markup language

Příloha **B**

Obsah přiloženého CD

- `readme.txt` ..... stručný popis obsahu CD
- `exe` ..... adresář se spustitelnou formou implementace
- `src`
  - `impl` ..... 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