České vysoké učení technické v Praze
Fakulta elektrotechnická

katedra počítačové grafiky a interakce

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

Student: **Aleksandr Primak**

Studijní program: Softwarové technologie a management
Obor: Web a multimedia

Název tématu: **Serverová část aplikace pro zpracování videa**

Pokyny pro vypracování:

Navrhněte a implementujte aplikaci, která bude zpracovávat video tak, že z něj extrahuje sadu signifikantních snímků. Extrakce bude řízena vstupními parametry, například určitý počet snímků za časový úsek, počet snímků ve scéně apod.
Uvažujte různé formáty a kodeky videa. Uvažujte celý proces pořízení a zpracování videa pro účely policejního vyšetřování. Aplikaci otestujte na vzorcích videa o maximální délce 10 minut a to:
a) video pořízené amatérskou kamerou v exteriéru,
b) video obsahující reklamní TV vstup,
c) video zachycující sportovní přenos, například fotbal.
Vyhodnoťte úspěšnost rozpoznání střihu, tj. nové scény ve videu.

Seznam odborné literatury:

[1] David Austerberry. The technology of video and audio streaming, Burlington : Focal Press, 2005
[2] H.R. Wu, K.R. Rao. Digital video image quality and perceptual coding, Boca Raton : Taylor & Francis, 2006
[3] Al Bovik. The essential guide to video processing, Burlington : Academic Press, 2009
[4] Charles A. Poynton: A technical introduction to digital video, New York : Wiley, 1996
[5] Iain E.G. Richardson: H.264 and MPEG-4 Video Compression, Chichester : Wiley, 2003

Vedoucí: Ing. Martin Klíma, Ph.D.

Platnost zadání: do konce letního semestru 2014/2015

prof. Ing. Jiří Žára, CSc.
vedoucí katedry

L.S.

prof. Ing. Pavel Ripka, CSc.
děkan

V Praze dne 26. 2. 2014

České vysoké učení technické v Praze
Fakulta elektrotechnická
Katedra počítačové grafiky a interakce

Bakalářská práce

Serverová část aplikace pro zpracování videa

Aleksandr Primak

Vedoucí práce: Ing. Martin Klíma, Ph.D.

Studijní program: Softwarové technologie a management, Bakalářský

Obor: Web a multimedia

3. ledna 2015

Poděkování

Rád bych poděkoval všem, kteří mi pomáhali při vzniku této práce. Zejména vedoucímu mé bakalářské práce, Ing. Martinu Klímovi, Ph.D., pomoc při zpracování, rady a cenné připomínky.

Prohlášení

Prohlašuji, že jsem práci vypracoval samostatně a použil jsem pouze podklady uvedené v přiloženém seznamu.
Nemám závažný důvod proti užití tohoto školního díla ve smyslu §60 Zákona č. 121/2000 Sb., o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon).

V Praze dne 3. 1. 2015. ……..

Abstrakt

Tato bakalářská práce se zabývá návrhem serverové části aplikace pro zpracování videa, která je zodpovědná za nalezení přechodů mezi scénami. Seznamuje čtenáře se současně používanými technologiemi v oblasti kódování a zpracování video dat. Rozebírá již existující řešení detekce scén ve videu, používané nástroje pro detekce přechodů a popisuje návrh, implementaci a testování dvou různých algoritmů.

Abstract

This work deals with design and implementation of the server-side application for video processing, which is responsible for detection of transitions between scenes. It familiarizes readers with modern technologies used for encoding and decoding videos and for working with video data. It analyzes existing solutions of transition detection, tools, which are used for scene detection and shows draft and testing of two different algorithms.

OBSAH

SEZNAM OBRÁZKŮ

SEZNAM TABULEK

Kapitola 1

ÚVOD

Tato práce se zabývá vývojem serverové části aplikace pro zpracování videa s možností detekovat jednotlivé scény, ze kterých se skládá vstupní video.

Aplikace je určena motivována potřebami odvozenými od projektu, který se zabývá vyhledáváním odcizených předmětů. Potenciálním uživatelem je policie České republiky, která potřebuje mít nástroj, pomocí kterého se dá natočit nebo vyfotit nějaké množství předmětů nalezených u pachatelů trestné činnosti, a ověřit pomocí porovnání obrázků s databázi kradených věcí, jestli vyfocené předměty jsou či nejsou kradené. Bylo nutné práci integrovat se sadou již existujících modulů aplikace. Cílem bylo realizovat rozdělení videa do jednotlivých signifikantních snímků, které pak jiné části aplikace porovnají s obrázky v databázi.

Na začátku byla provedena rešerše zaměřená na analýzu existujících řešení a jejich porovnání, následoval návrh a implementace vhodných algoritmů.

1.1 MOTIVACE

Technologie se neustále rozvíjejí a každý se snaží jít s dobou a používat nejnovější nástroje. Vláda je hlavním názorným příkladem pro ostatní. A tím, že obnovuje již existující technologie nebo zavádí nové, upevňuje svou pozici ve světě a zlepšuje životní úroveň. Aby každý občan byl spokojen s životem ve své zemí, musí být moderní a spolehlivá policie, která může toho dosáhnout pomocí pokročilé technologie.

Jedním z důvodů, proč jsem nechtěl studovat a bydlet v Ruské federaci, je nedůvěra ke státním službám a jejich slabý rozvoj. Proto jsem byl rád, když můj vedoucí navrhnul téma bakalářské práce tykající se zabezpečení občanů České republiky.

1.2 CÍL PRÁCE

Cílem práce je navrhnout serverovou část aplikace pro detekci přechodů mezi scénami ve vstupním videu. Aplikace musí podporovat různé formáty a kodeky videí, která jsou současně používána na mobilních zařízeních, umět detekovat přechody scén ve videu, které bylo natočeno pomocí tlačítka pauzy, a každá detekce musí být prvním snímkem nové scény. Konečný výsledek musí být dobře zdokumentován a musí obsahovat pochopitelný kód.

1.3 STRUKTURA PRÁCE

Práce je rozdělena do pěti hlavních části. Na začátku je popsána analýza požadavků uživatele (funkčních a nefunkčních), rozbor existujících formátů a kodeků videí, která používá moderní mobilní zařízení, analýza nástrojů, ve kterých je implementována možnost detekce přechodů mezi scénami, popis již existujících řešení dané problematiky, které byly prezentovány na TRECVIDu nebo které byly otestovány na datech z TRECVIDu a výběr frameworku pro implementace zadání.

Druhá část je věnována návrhu vlastního algoritmu, používajícího základní metody pro práce s videem, včetně rozdělení na podproblémy jako výběr vhodných základních signálů a zpracování a vzorkování těchto signálů. V práci jsou popsány vyzkoušené algoritmy z analýzy a jejich detailnější rozbor.

Třetí část popisuje implementaci aplikace včetně celkové architektury, implementace v jazyce C++ a konečnou implementaci v jazyce Python. Pro lepší porozumění se používají UML diagramy a jsou stručně popsány zajímavé věci nalezené během vývoje.

Ve čtvrté části je demonstrována sada videí, která byla vybrána pro výsledné testování metod. Je popsána základní informace o každém videu z kolekce.

Poslední část ukazuje, jak se testovala aplikace, jaké hodnoty měření byly použity a výsledky všech vyzkoušených metod ve formě grafů a tabulek.

Závěr ukazuje celkové výsledky aplikace, jak byla užitečná pro zákazníka, možnosti pro další rozvoj práce a zkušenosti, které jsem získal v průběhu této práce.

Kapitola 2

ANALÝZA

2.1 FUNKČNÍ A NEFUNKČNÍ POŽADAVKY

Na základě daného zadaní a konzultace s vedoucím projektu byly sestaveny funkční a nefunkční požadavky aplikace. Pak, v procesu implementace, byl změněn jeden z nefunkčních požadavků, který způsobil změnu celkové architektury systému.

2.1.1 FUNKČNÍ POŽADAVKY

1. Aplikace musí umět detekovat náhlé přechody mezi scénami. Náhlým přechodem se rozumí situace, když předchozí snímek je součástí staré scény a běžný snímek je začátkem nové scény. Takové přechody jsou také známé jako ostré střihy. Jiným typem přechodu je postupný přechod, který je výsledkem umělé editace video. Existuje několik typů postupných přechodů jako zatmívání, roztmívání, prolnutí atd. Vzhledem k tomu, že aplikace má zpracovat videa, která jsou natočena pomocí mobilních zařízení, postupné přechody se neberou v úvahu a náhlé přechody v daném problému jsou jediným typem uvažovaných přechodů.
2. Aplikace má pracovat běžnými formáty a kodeky videí, které se používají při natáčení v mobilních zařízeních. Další informace ohledně kodeků a formátů videí je popsána v následující kapitole.
3. Každá metoda, která bude použita pro detekce, musí být otestována pomocí samostatně shromážděné kolekce videí a dat. Výsledkem testování musí být měřicí hodnoty, které se světově používají pro danou problematiku.
4. Pro testovací účely se aplikace může spouštět pomocí příkazového řádku. Vstupem aplikace bude lokální cesta videa, případně další proměnné.
5. Výsledky aplikace se ukládají textově do souboru. Každý výsledek jasně popisuje pozice snímku ve videu, buď jako číslo snímku nebo čas.

2.1.2 NEFUNKČNÍ POŽADAVKY

1. Aplikace musí být napsána v jazyce C++. Jazyk se později změnil na Python kvůli integritě celého systému, protože ostatní moduly, které by měly používat tuto aplikaci, jsou naprogramovány pomocí jazyka Python.
2. Pokud budou zvoleny kodeky nebo knihovny, musejí mít licenci, která povoluje použití pro komerční cíle.
3. Kolekce videí má vzorky o maximální délce 10 minut v každé kategorie a to:
    a. video pořízené amatérskou kamerou v exteriéru
    b. video obsahující reklamní TV vstup
    c. video zachycující sportovní přenos, například fotbal

2.2 FORMÁTY A KODEKY VIDEÍ

2.2.1 KODEKY

Kodek je složenina z počátečních slabik slov „kodér a dekodér“ a je převzato s anglického slova codec analogického původu. Je to zařízení nebo počítačový program, který dokáže transformovat datový proud pomocí použití kompresního schématu. Kompresní schéma říká jak se zbavit nadbytečné informace ze vstupních dat a jak to pak zrekonstruovat. Kodeky se dělí na dva typy: ztrátové a bezztrátové. Bezztrátové kodeky plně zachovají data, ale nemůžou dosáhnout potřebné komprese pro ukládání videa. Ztrátové kodeky ztrácejí část informace, kterou si člověk nedokáže všimnout, a používají HVS (Human Visual System) model, který je založen na principech lidského vnímání. Některé kodeky mají efekty změny nebo redukce barevného gamutu a tohle může někdy vést k nepříjemným změnám barvy. Kvůli těm vlastnostem a rozdílem mezi podvzorkováním chrominančních složek kodeky produkují kolísavou kvalitu videa.

2.2.1.1 YCBCR

Většina kodeků používá YUV, YCbCr nebo YPbPr barevné prostory. YUV se používá pro vysílání signálu, YPbPr – pro analogové video a YCbCr – pro digitální video. Proto, pro danou práci zkoumáme jenom YCbCr, na kterém je založena známá rodina kontejnerů a kodeků MPEG.

Jednotlivé složky barevného prostoru YCbCr jsou luminance Y (jas), chrominanční složka Cb (modrá brava) a chrominanční složka Cr (červená barva). Chroma Cb odpovídá složce U a chroma Cr odpovídá složce V obecného barevného prostoru YUV, kde U a V jsou barevné složky v rozsahu od -0.5 do +0.5.

2.2.1.2 PODVZORKOVÁNÍ YCBCR

YCbCr se představuje jako 4:a:b, což znamená, že spolu s 4 složkami jasu (Y) jdou a Cb chrominančních složek a b Cr chrominančních složek. Typicky jsou 4 obecné druhy podvzorkování:

- 4:4:4 – žádné podvzorkování
- 4:2:2 – spoření je 33%
- 4:1:1 – spoření je 50%, je dobré pro malá uložiště dat
- 4:2:0

RGB barevný prostor může být transformován do YCbCr barevného prostoru pomocí rovnice:

$$Y=0.299*R+0.587*G+0.114*B$$

$$Cb=0.168736*R-0.331264G+0.5*B$$

$$Cr=0.5*R-0.418688*G-0.081312*B$$

Kromě barevných prostorů kodeky využívají různé druhy komprese nebo jejich kombinace.

2.2.1.3 PERCEPČNÍ KOMPRESE

Video kodeky, které jsou založeny na daném typu komprese, typicky využívají slabiny lidského zraku. Například, naše oči mají větší citlivost k jasu, než k barvě a proto můžeme relativně zmenšit šířku pásma barevných složek, což způsobuje větší kompresi, jak je to u podvzorkování YCbCr.

2.2.1.4 PROSTOROVÁ KOMPRESE

Prostorová nebo intraframe komprese využívá podobnost snímků ve videu. Často šum nebo filmové zrno můžou zmenšit efektivitu prostorové komprese.

2.2.1.5 TEMPORÁLNÍ KOMPRESE

Temporální nebo interframe komprese umožňuje využit podobnost mezi snímky ve videu. Pokud dva po sobě jdoucí snímky mají stejné pozadí, nemusíme zachovat pozadí dvakrát. Místo toho, můžeme uložit jenom rozdíl mezi snímky.

Některé kodeky zařadí kompenzace pohybu, aby dosáhly větší komprese. Místo jednoduchého porovnání dvou po sobě jdoucích snímků tato technologie registruje pohybující se objekty ve snímku a předpoví, kde budou tyto objekty v následujícím snímku.

Závislost jednoho snímku na jíném dělá potíže se zpracováním temporálně stlačeného videa a editace je v takovém případě složitá nebo vůbec nemožná. Pro editaci nejspíše musíme dekódovat a pak opět zakódovat video, což často způsobuje vznik značných artefaktů. Z tohoto důvodu se temporální komprese nejlépe hodí jenom pro konečný výsledek, když už nikdo nebude měnit obsah videa.

Temporální komprese může přinést významnou redukci dat. Nicméně její efektivita je snížena kvůli obecným filmovým technikám jako panoramatický pohled kamery, sledování, pomalé přibližování nebo oddalování a práce s ruční přenosnou kamerou.

Stejně jako prostorová komprese, temporální komprese je citlivá k šumu a filmovému zrnu.

2.2.1.6 POROVNÁNÍ KODEKŮ

Na základě analýzy byla vytvořena Tabulka **2.1** s charakteristikami kodeků, které jsou nejvíc používány dnes. Je to pouze seznam kodeků, které jsou interně používány uvnitř webových služeb nebo zařízení.[2]

Název	Typ kodeku	Kompresní schéma	Podvzorkování	Popis	Použití	Licence
x264*	Ztrátový/Bezztrátový	MPEG-4 AVC/H.264	4:0:0, 4:2:0, 4:2:2 and 4:4:4	Široce používaný kodek. Mobilní zařízení obecně používají Baseline profil tohoto kodeku.	Mobilní zařízení, přehrávače Blu-ray Disků, streamování na internetu.	GNU GPL
ProRes	Ztrátový	Apple ProRes	4:2:2, 4:4:4	ProRes 422 je intraframe kodek, používající DCT (Dicrete Cosine Transform)	Zařízení Apple.	Vlastnické právo
Motion jpeg (Mjpeg)	Ztrátový/Bezztrátový	Mjpeg (Jpeg)	4:2:2, 4:2:0	Intraframe kodek. Každý video snímek zakódován zvlášť jako JPEG obrázek.	Digitální kamery, IP kamery a webové kamery.	Žádná
Motion jpeg 2000	Ztrátový/Bezztrátový	Motion jpeg 2000	4:2:2	Je moderní aktualizací Mjpeg. Nezahrnuje interframe kódování	Střední nebo konečný stav archivace nebo dodávka koncovému uživatelovi.	Volná
VP8	Ztrátový	VP8	4:2:0	Prostorová komprese pomocí DCT.	Youtube používá směs HTML5 a VP8.	BSD typu
MPEG-2	Ztrátový	MPEG-2	4:4:4, 4:2:2, 4:2:0	Je podobný předchozímu MPEG-1 standardu, ale navíc podporuje prokládání video a není optimalizovaný pro malou přenosovou rychlost, poskytuje prostorovou and temporální komprese.	TV přijímače, TV stanice a DVD přehrávače používají MPEG-2	Je řízena MPEG LA LLC
DV	Ztrátový	DIF	4:1:1, 4:2:0	Poskytuje intraframe kompresi pomocí DCT.	Je používán v kamerách.	Není nalezena

Tabulka 2.1 Seznam kodeků, které jsou nejvíc používány dnes.

2.2.1.7 ZÁVĚR

V dnešní době existuje velké množství různých kodeků, které mohou být placené a bezplatné a které používají různé techniky pro kódování videí. Implementace každého kompresního schématu vlastními prostředky může trvat nepřiměřeně dlouho. Proto je prakticky jediným východiskem najit univerzální sadu kodeků, která je distribuována pod LGPL nebo BSD licencemi. Nejvhodnějším kandidátem pro dekódování je FFmpeg knihovna, která implementuje velkou sadu kompresních schémat a je pod LGPL a GPL, přičemž GPL verze je nutná jenom pro zakódování některých schémat.

2.2.2 FORMÁTY KONTEJNERŮ

Kontejner popisuje strukturu souboru: kde se ukládají různé části, jak jsou prokládány a které kodeky jsou použité pro jednotlivé části. Specifikuje video a audio kodeky, pokud se liší. Kontejner říká počítači jak přečíst informace z daného souboru. V této kapitole probereme nejpoužívanější kontejnery jako Mpeg, Matroska Media Container, Microsoft AVI, Windows Media Video a další.

2.2.2.1 MPEG

MPEG (Moving Pictures Expert Group) je pracovní skupina vyvíjející standardy používané na kódování audiovizuálních informací. Jsou tři hlavní kontejnery, které byly vyvinuty MPEG skupinou: MPEG-1(mpg), MPEG-2(m2v) a MPEG-4(mp4). Podle výzkumu společnosti Sorenson Media [3] MP4 je nejpoužívanějším výstupním kontejnerem pro webová a mobilní videa (69% a 58%). Video uvnitř MP4 je obecně zakódováno pomocí H.264 a audio pomocí AAC, ale podporuje i další kompresní schémata.

2.2.2.2 3GP

Většinou 3GP je asociován s formátem 3GPP. 3GPP a 3GPP2 jsou světové standardy pro vytváření, přenos a přehrání multimédia přes bezdrátové sítě 3 generace. Tyto standardy jsou definovány partnerskými projekty The 3rd Generation Partnership Project a The 3rd Generation Partnership Project 2 v tomto pořadí a jsou založeny na standardu MPEG-4, který je odvozen z Apple Quick Time. QuickTime 6.5 obsahuje podporu klíčových komponentů specifikací 3GPP a 3GPP2.

2.2.2.3 THE QUICKTIME MOVIE FILE FORMAT

Velmi často tento formát zkráceně nazývají QuickTime soubor. Má příponu .mov a používá se společností Apple pro zakódování videa. Je velmi podobný MP4, protože MP4 je založen na tomto kontejneru. QuickTime video obsahuje media data v samostatných dráhách, což umožnuje ukládat mnohonásobné části videa a audia s různými kodeky pro každou část. Typicky videa v QuickTime kontejnerech jsou zakódována pomocí MPEG-4 AVC nebo H.264 a audia pomocí AAC.

2.2.2.4 MICROSOFT AVI

Je video kontejnerem, který byl vyvinut společností Microsoft. Má příponu .avi a typickými kodeky pro tento kontejner jsou DivX, XviD, DV a Huffyuv. Microsoft AVI nebyl navržen pro jeden formát nebo standard a proto neobsahuje velkou dodatečnou informaci o videu jako například kontejner MP4. Kromě toho, avi soubory mají obecně menší kompresi než kontejnery MPEG nebo MOV.

2.2.2.5 MATROSKA MEDIA CONTAINER

Matroska používá příponu .mkv a prochází z projektu MCF, ale významně od něj odlišuje tím, že je založen na EBML (Extensible Binary Meta Language). EBML umožní vývojovému týmu Matroska získat značné výhody z hlediska budoucí rozšiřitelnosti formátu, aniž by přerušily podporu starých parserů. Zdrojový kód knihovny vývojového týmu Matroska má GNU LGPL licenci a navíc existují bezplatné knihovny, které mají BSD licenci a jsou dostupné pro komerční účely.

2.3 NÁSTROJE, KTERÉ POUŽÍVAJÍ DETEKCE PŘECHODŮ MEZI SCÉNAMI

V rámci analýzy nástrojů používajících metody detekci scén byly vybrány nejznámější produkty, které mají zkušební verze nebo jsou zdarma. U většiny nástrojů má detekce scén dva různé typy: optická metoda a metoda detekce podle data natáčení. Když je zvolena optická metoda, program zpracovává obsah souboru, objevuje scény a rozdělí vstupní video. Pro analýzu optickou metodou se používají jenom data video proudu, a proto není potřeba další zdroj informací. Druhá metoda zpracovává data natáčení každého snímku ve videu, která jsou uschována v metadatech videa. Kamery používající standard DV a jemu podobné, ukládají během natáčení čas a datum pro každý snímek. Pak, když přesuneme video do počítače pomocí digitálního rozhraní FireWare, tato informace bude uložena ve video kontejneru. Pro danou metodu přesnost detekci se blíží 100%.

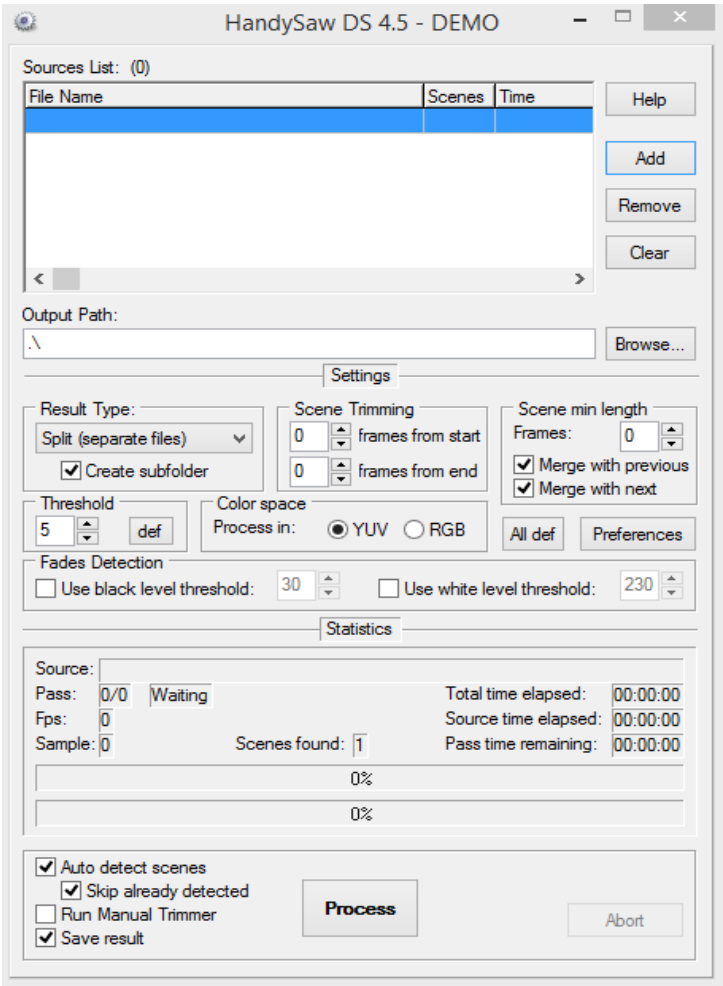
Pro testování nástrojů bylo zvoleno jedno video s kontejnerem .avi a videokodekem Xvid, které knihovna OpenCv má v instalačním baličku. Video má délku 11 vteřin a celkově 5 scén, přičemž první scéna se skládá ze dvou černých snímků. Tím pádem, protože žádný nástroj touto scénu nedetekoval, první scéna se nepočítá při hodnocení výsledků programů.

Kupodivu, některé známe programy jako Final Cut Pro X nebo Adobe Premiere Pro CC 2014, sloužící pro editace videí, nemají integrovanou detekci přechodů. Final Cut Pro X má jenom doplněk, který se nazývá Scene Detector. Adobe Premiere Pro CC 2014 má jenom metodu pro detekci podle data natáčení, proto zde nebyl popsán.

Dále jsou popsány nástroje, které vyvíjeli buď jeden člověk nebo celá společnost a můj vlastní názor ohledně uživatelského rozhraní a detekce scén pro každý program.

2.3.1 HANDYSAW DS

V roce 2000 Dmitrij Sinicyn vyvíjel program pro automatickou optickou detekci scén ve videu, která současně má název HandySaw DS. Je to komerční software a stojí 54.95 EUR, ale k dispozici je demoverze, která povoluje zpracovat 5 souborů, přičemž pro každý soubor bude uloženo jenom prvních 15 detekovaných scén. Obrázek **2.1** znázorňuje uživatelské rozhraní HandyShow DS.



Obrázek 2.1 Uživatelské rozhraní Handyshow DS

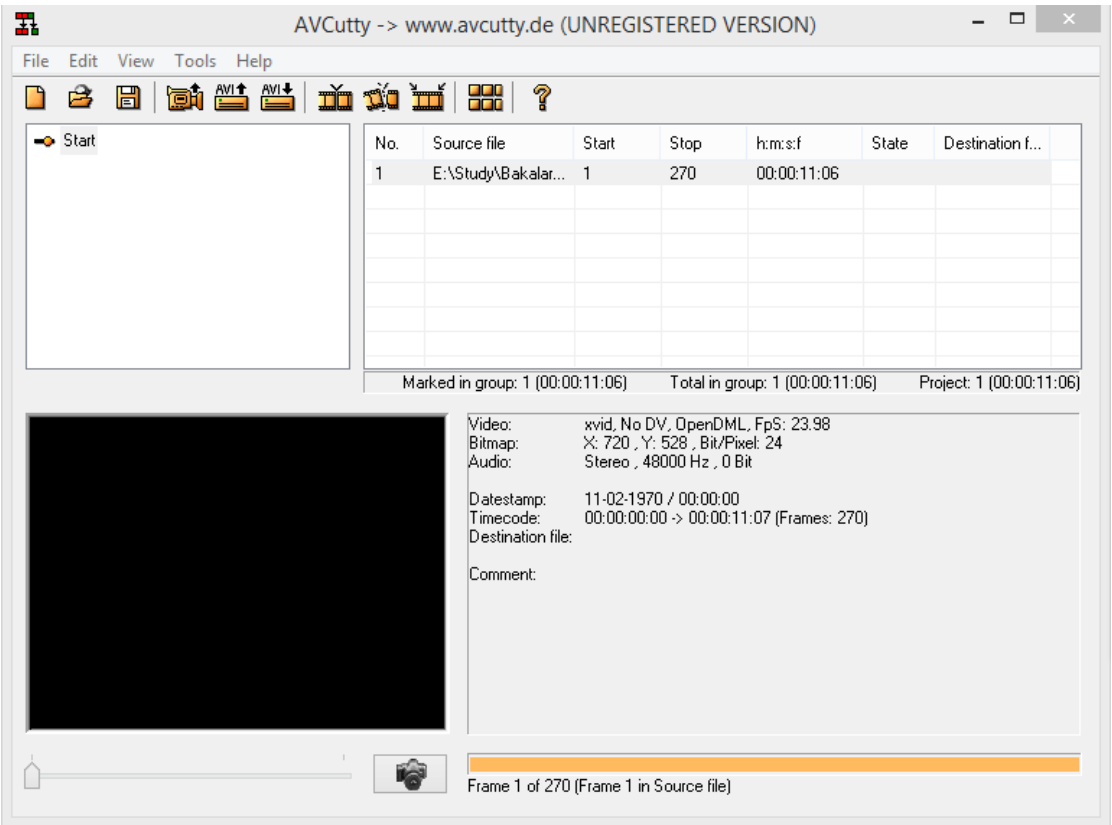
Podle [1], HandySaw má dvě metody detekce scén: optickou a podle data natáčení. Optická metoda samozřejmě nemůže garantovat 100% výsledek, ale autor tvrdí, že úspěšnost je velmi vysoká.

HandyShow má různé nastavení pro zpracování jako způsob zápisu výsledků, určení prahové hodnoty, výběr barevného prostředí, ve které bude zpracováno video, atd. Pro testování bylo použito malé video, které bylo popsáno v úvodu kapitoly. Textový zápis výsledku dopadl dobře (100%), ale při zvolení typu výsledků Rozdělení (Split) výstup byl nepochopitelný. První scéna byla správná, ale ostatní se skládali buď z jednoho snímku, nebo vůbec neměli snímky, jenom zvukovou stopu.

Program HadyShow DS má pěknou rychlost detekce a jednoduché uživatelské rozhraní. Podporuje velké množství kontejnerů a umí pracovat ve dvou barevných prostředích: RGB a YUV. Kromě toho, umožnuje flexibilní nastavení detekce a zpracování celé sady videí najednou. Na druhou stranou, potřebuje ještě dodělání v zápisu výsledků. Detekce střihů byla úspěšná (100% detekovaných přechodů), ale srozumitelným výstup byla pouze zápis do textového souboru. Celkově je to dobrá aplikace pro zpracování videa, potřebuje jenom dodělání zápisu výsledků pomocí funkce Rozdělení.

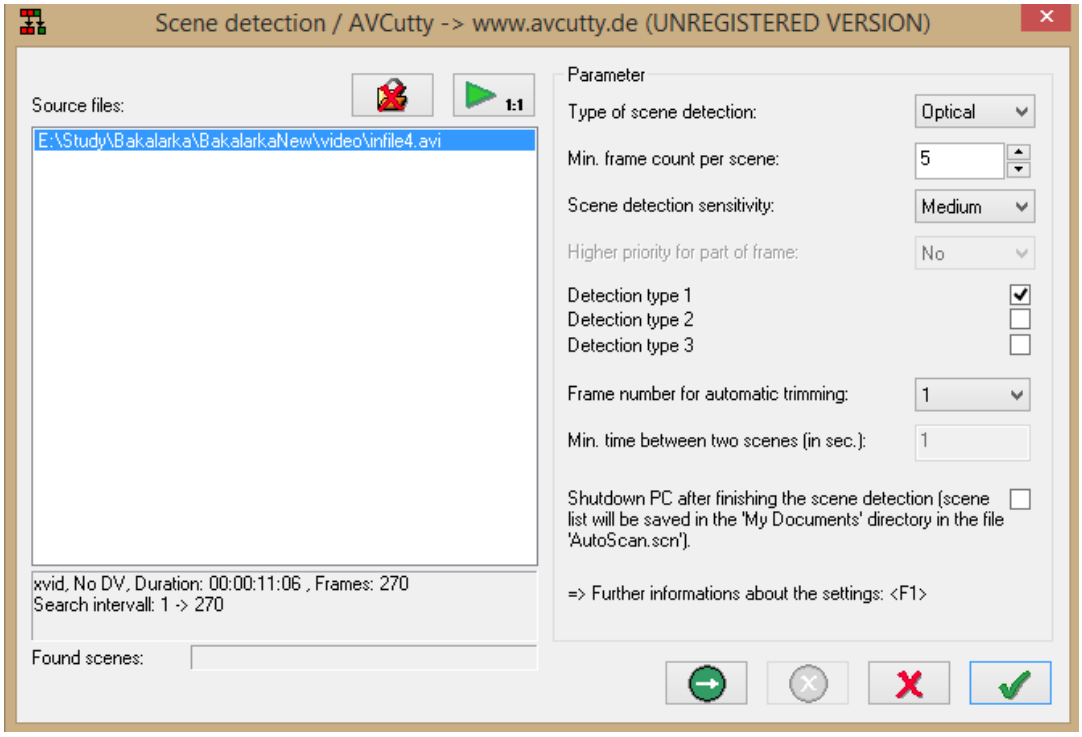
2.3.2 AVCUTTY

AVCutty je program, který umožnuje snadnou editaci videa včetně digitální (pomocí informaci z DV kamery) a optickou detekci scén, rozdělení videa do menších úseků a řazení scén pomocí rozdělení do skupin. Webová stránka projektu [4] popisuje základní funkce aplikace. AVCutty umí pracovat jenom s kontejnerem avi a ukládá snímky jako BMP. Obrázek **2.2** demonstruje hlavní rozhraní pro ovládání aplikací. Uživatelské rozhraní je jednoduché, ale někde je trochu uživatelsky nepříjemné. Například, **OBRÁZEK 2.3** znázorňuje menu automatické detekce scén, které nabízí různé typy detekcí, ale není napsáno, jak každý druh funguje a čím se typy detekcí liší.



Obrázek 2.2 Hlavní menu

V hlavním menu můžeme vytvořit nový projekt a přidat nové video buď z DV kamery pomocí rozhraní FireWare, aby pak bylo možné detekovat scény pomocí digitálního způsobu, nebo obecné avi video ze souboru. Navíc je možnost přetáhnout scény z jedné skupiny do druhé pro řazení a další funkce.



Obrázek 2.3 Rozhraní pro detekci scén

Detekce přechodů mezi scénami nebyla úspěšná, žádná scéna nebyla nalezena. Zkoušel jsem všechny tři typů detekce, ale ani jeden neměl dobrý výsledek.

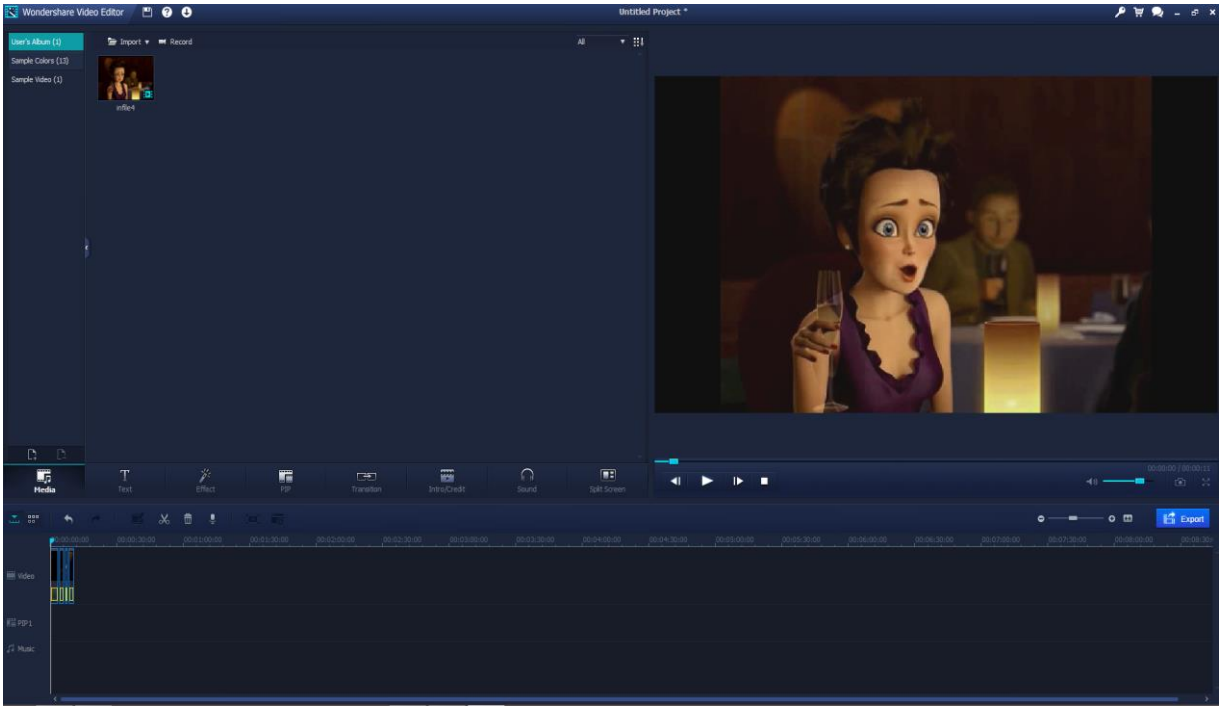
AVCutty je samostatný projekt německého programátoru Andreas von Damaros. Je k dispozici bezplatná verze produktu, ale je „Donationware“, což znamená, že pokud někdo používá aplikaci, musí podpořit autora nějakou částkou, kterou zvolí sám. Je relativně jednoduchý, protože umí pracovat jenom s avi videi a zpracovat data z DV kamery. Z hlediska detekce scén nemůžu ocenit program dobře, protože tento nástroj nenalezl žádnou scénu (i když bylo zvoleno několik typů detekce).

2.3.3 Wondershare Video Editor

Společnost Wondershare se zabývá nástroji pro obnovu dat a jejich převod mezi různými zařízeními, business aplikacemi jako PDF Editor a stavitel webových knih a nástroji pro práci s videi. Jedním z nejúspěšnějších nástrojů je Wondershare Video Editor (dále jen WVE), který umožnuje rychlou a jednoduchou editaci videí. Hlavní stránka WVE [5] hlásí, že tento produkt má velké množství funkcí jako separace audia, automatická detekce scén, PIP (Picture-In-Picture) nástroj pro mix dvou videí, zoomování důležitých oblastí ve videu atd. Kromě toho, WVE obsahuje obrovskou sadu grafických filtrů, prvků pro doplnění, efektů pro přechody mezi scénami a animaci textu, a velkou knihovnu zvukových souborů.

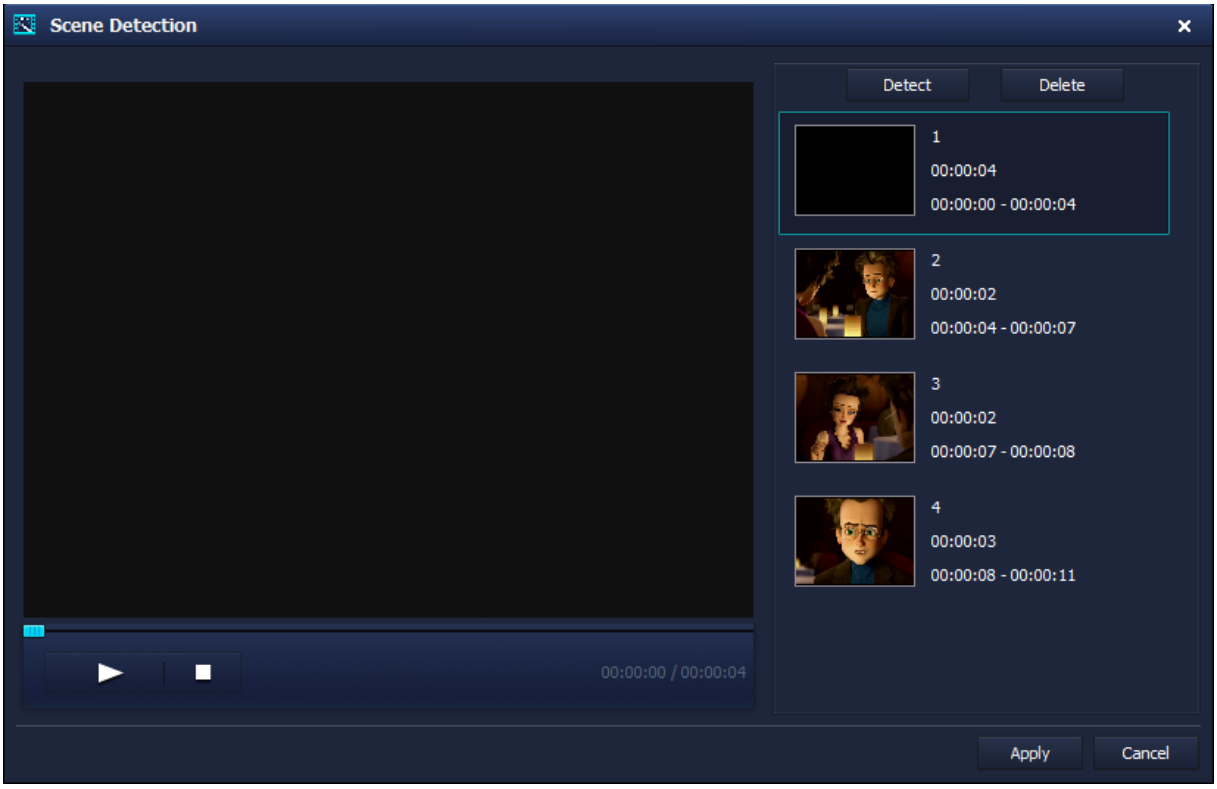
Tento nástroj je komerční, stojí cca 40 dolarů, ale je k dispozici zkušební verze produktu, která má podle [6] několik omezení: výstupní video bude mít firemní vodoznak, není technická podpora a není aktualizace aplikace.

Uživatelské rozhraní je intuitivní a má hezký vzhled. Obrázek **2.4** demonstruje tento vzhled. Projekt může obsahovat několik videí a každé video se dá editovat zvlášť.



Obrázek 2.4 Hlavní menu

Detekce scén (Obrázek **2.5**) je velmi jednoduchá. Má jenom tlačítko „Detect“ pro detekci a „Delete“ pro editaci jednotlivých scén. Pro testované video byla velmi rychlá a výsledek byl výborný (100%). Výstupní video se dá uložit do různých formátů a zařízeni jako například do iPhonů, PS3, HTC One atd. Kromě toho, je možné nastavit kodeky pro audio a video, rozlišení, počet snímků a bitů za sekundu a další volby.



Obrázek 2.5 Rozhraní pro detekci scén

Wondershare Video Editor je komerční produkt společnosti Wondershare a podle mého názoru je nejlepším výběrem mezi nástroji pro práce s videi, které jsem vyzkoušel. Oproti ostatním aplikacím je docela levný a má zkušební verzi, ve které člověk může pochopit, zda mu tento program vyhovuje nebo ne. Podle [7] WVE se neustále rozvíjí a stále sleduje moderní technologie. Tento nástroj má obrovské možnosti pro editaci videí a automatickou detekci scén libovolného videa, která je rychlá a docela důvěryhodná.

2.4 EXISTUJÍCÍ ŘEŠENÍ

V této kapitole jsou popsány metody / algoritmy detekce scén ve videu. Ty, které nebyly vybrány pro implementaci z různých důvodů, jsou zvlášť označeny.

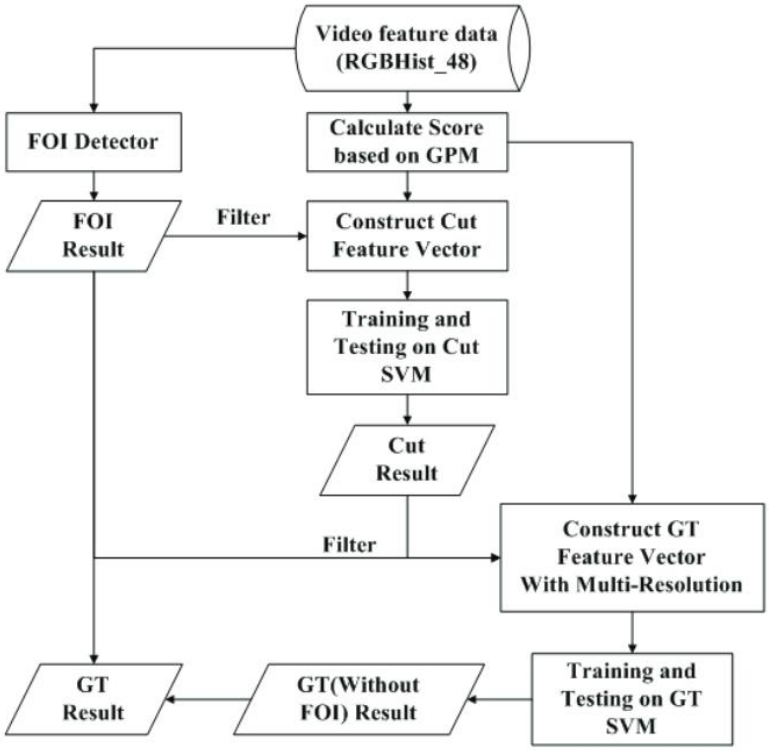
Problémem detekce přechodů mezi scénami se vývojáři zabývají již několik let a v poslední době se projevilo mnoho různých řešení tohoto problému. Hlavní institucí, která se věnuje detekci scén a celé oblasti videosémantiky, je série konferencí TRECVID, která je sponzorována NIST (National Institute of Standards and Technology) s dodatečnou podporou od jiných vládních agentur v USA. Podle [8], cílem TRECVIDu je podporovat výzkum v oblasti vyhledávání informací poskytováním velké testovací kolekce videí a nezávislé komise, zabývající porovnáním výsledků detekce scén. Každý rok se objevují různé kategorie, kterých se bude týkat TRECVID, pak účastnici odesílají své výzkumy a kód pro testování do nějakého určeného data. Na konci TRECVIDu se projednávají kategorie pro výzkum na příští rok.

Problémem detekce změn scén se zabývali na TRECVIDu během sedmi let (pak už se začaly složitější výzkumy v oblasti videosémantiky). Během studia materiálů [9] a [10] z TRECVIDu jsem vybral dvě nejlepší metody. Ty jsou vítězi (1, 2 místo) z TRECVIDu 2007, který byl posledním shromážděním, týkajícím SBD problematiky. Hlavním důvodem bylo to, že testovací videa se mění každý rok a testovací videa roku 2007 obsahovala především náhlé přechody (abrupt transitions) - 90.8%, které jsou nejčastějším výskytem v aplikaci pro danou bakalářskou práci a jejich detekce je důležitým funkčním požadavkem pro systém.

Z TRECVIDu byly zvoleny 2 metody: první je metoda univerzity Tsinghua s podporou Intel China a druhá je metoda univerzity Bradford. Kromě toho, byl zvolen algoritmus Ali Amiri a Mahmooda Fathy ze článku [11], který podle popisu byl taký testován na datech z TRECVIDu a autoři tvrdí, že výsledek jejich aplikace byl lepší, než je u jiných metod.

2.4.1 TSINGHUA

Algoritmus univerzity Tsinghua byl poprvé představen v roce 2004, ale v roce 2005 algoritmus byl kompletně změněn a pak se každý rok zlepšoval. Podle [12], první verze roku 2005 se skládala ze třech komponentů: FOI(Fade out/in) detektor, všeobecný detektor ostrých přechodů a detektor dlouhých postupných přechodů. Detekce náhlých přechodů byla detekována pomocí jednoho algoritmu podpůrných vektorů, který bral na vstup vektor hodnocení vypočítaný pomocí modelu rozdělení grafů. Obrázek **2.6** ukazuje celkovou architekturu systému.



Obrázek 2.6 Architektura Detektoru scén univerzity Tsinghua, zdroj [12]

V roce 2006, podle [13], algoritmus byl změněn o 3 malé zlepšení: přidání detekce blesků, krátkých postupných přechodů a vylepšená detekce FOI. Ve FOI detektoru byl změněn modul detekce monochromatických snímků ve scéně. Myšlenkou modulu je, že snímky, které zároveň mají jednotnou barvu a rovnoměrné rozmístění, jsou klasifikovány jako černobílé. Detektor blesků je založen na tzn. Poměru Změny Hran (anglicky Edge Change Ratio), který podle Lienhartu [16] je schopen detekovat blesky a falešné detekce. Detektor krátkých postupných přechodů je učen pro detekci postupných přechodů, které mají délku v rozmezí od 5 do 10 snímků. Systém univerzity Tsinghua používá tento detektor po zpracování ostrých a dlouhých postupných přechodů pro další detekci krátkých přechodů. Tréninková metoda pro tento detektor je stejná jako u ostatních detektorů.

Finální algoritmus roku 2007 [14] má skoro stejnou strukturu jako předchozí algoritmy, ale byly provedeny tři významných změny. První změna se tyká množství bloků pro různé typy detektorů. Dřívé, všechny detektory měly stejnou délku bloků pro každý snímek. To znamená, že každý snímek byl rozdělen do 4 krát 4 bloků, ale výsledky vyhodnocení ukázaly, že detektor náhlých přechodů měl by mít větší množství bloků a v nové verzi detektor je rozdělen do 16 krát 16 bloků. Druhou změnu bylo přidání nového modulu pro detekci pohybů do detektoru postupných přechodů, aby zredukoval falešné detekce, které jsou způsobeny pohybem kamery nebo pohybem velkých objektů. Poslední změna se týkala zpracování dat po detektorech přechodů. Byl přidán nový modul, který se nazývá SIFT (Scale invariant feature tranform). Je to algoritmus, který se používá pro rozpoznání objektů a popis obrazů.

Algoritmus univerzity Tsinghua vykazoval dokonalé výsledky během 3 let a v roce 2007 byl nejlepším. Je to velmi složitý algoritmus, který se rozvíjel pomocí desítek pracovníků univerzity a podporou společnosti Intel. Kromě detekce obecných (ostrých) přechodů se struktura skládá

z velkého množství modulů určených pro detekci krátkých a dlouhých postupných přechodů, což je z hlediska funkčních požadavků pro touto bakalářskou práci zbytečné.

2.4.2 BRADFORD

Algoritmus anglické univerzity Bradford [15] byl poprvé prezentován na TRECVIDu 2007 a měl jeden z nejlepších výsledků. Oproti např. algoritmu univerzity Tsinghua, tato metoda pracuje přímo s komprimovanými videy, které používají kompresní schéma MPEG.

Hlavní jednotkou pro extrakci charakteristik videa je makroblok 16X16 pixelů. Výsledný vektor charakteristik vypadá takto:

$$V_i = (err(i), E_y(i), \mu(i), \sigma(i), p_1(i), p_2(i))$$

Kde $err(i)$ je veličina, charakterizující možnost falešné detekce pohybu, $E_y(i)$ je normalizována energie jasové složky barevného prostoru YCbCr, $\mu(i)$ a $\sigma(i)$ je střední hodnota a směrodatná odchylka rozdílů mezi snímky, $p_1(i)$ a $p_2(i)$ jsou dvě části, reprezentující procentní podíl pixelů v rozdílu mezi snímky, které jsou větší než dvě dané prahové hodnoty $\lambda_1(i)$ a $\lambda_2(i)$.

Ostré přechody jsou rozděleny do 5 podprostorů, které jsou klasifikovány sekvenčním způsobem, tj. pro každý přechod zjišťujeme, zda jsou splněny podmínky pro nějakou kategorii. Pokud podmínky jsou splněny pro danou kategorii, přechod je zařazen do odpovídajícího podprostoru. V opačném případě jsou zkoumány následující podmínky: pokud se pro změnu ve videu nepodařilo najit vhodnou kategorii, tato změna není přechodem. I když tato pravidla mohou úspěšně detekovat správné přechody, budou také nalezeny falešné přechody způsobené pohybem kamery nebo objektů. Proto se navíc používá postupná korelační metoda.

Pro každý typ postupných přechodů se používá různá strategie. Například, dojde-li k prolínačce, obvykle můžeme zjistit, že v několika po sobě jdoucích snímcích je velká $err(i)$. Zároveň, střední hodnota $\mu(i)$ také je velká. To je základním principem pro detekci prolínaček. Předpokládáme, že snímek je kandidátem pro prolínačku, pokud má $err(i) \geq t_e$, kde $t_e = 15$. Pak je každý kandidát prodloužen do malého klipu pomocí postupného spojení sousedních snímku, pokud sousední snímek má $err(i)$ vetší než jedna třetina průměrné $err(i)$ vytvořeného klipu. Posledním krokem je sjednocení všech klipů do jednoho, pokud mají vzdálenost mezi sebou menší než 3 snímky. Pokud výsledný klip obsahuje méně než 3 snímky, je opuštěn.

Algoritmus univerzity Bradford nemůže být použit v rámci této bakalářské práce, protože používá pouze komprimovaná videa formátu MPEG, což nevyhovuje funkčním požadavkům. Měl dobré výsledky na TRECVIDu 2007 a je vhodný pro případ, když víme, jakého formátu jsou vstupní videa.

2.4.3 ALGORITMUS ALI AMIRI A MAHMOODA FATHY

Tento algoritmus je vyvinut v roce 2011 íránskými vědci Ali Amiri a Mahmoodem Fathy. Je založen na tzn. Zobecněné Dekompozici vlastních čísel a detekci Gaussových přechodů. Zobecněná dekompozice vlastních čísel (dále jen GED) spočívá v následující rovnici:

$$A = B.P.\Sigma.P_r^{-1} ,$$

kde A je matice $m \times n$ a $m \leq n$, B je jednotková matice stejného rozměru jako A, $P = [V_1, V_2, \dots, V_m]$ a je množinou *m* lineárně nezávislých zobecněných vlastních vektorů, $\Sigma = \mathrm{diag}(\lambda_1, \dots, \lambda_m)$ a λ_i je vlastní číslo odpovídající vlastnímu vektoru V_i pro $i \in \{1,2, \dots, m\}$ a P_r^{-1} je pravostranná inverzní matice k matici P.

Extrakce charakteristik z jednotlivých obrazů se provádí pomocí kombinaci barevných histogramů a GED. Pro každý obrázek se vytváří 500 rozměrný vektor charakteristik. Abychom spočítali tento vektor, vytváříme trojrozměrný histogram v barevném prostoru RGB, který má pět binů pro každou složku RGB prostoru. Bin zodpovídá za úsek hodnot pro dané rozmezí informace. Daný histogram má pět binů a je v RGB prostoru. To znamená, že pomocí binů zkrátíme interval [0,255] do intervalu $[[0,50], [51,101], [102,152], [153,203], [204,255]]$. Pak, pokud barva, například, má hodnotu 65, druhý bin v histogramu se inkrementuje. Tímto způsobem, výsledný histogram obsahuje 125 binů. Pro zachování prostorové informace, každý snímek se rozdělí do 4 bloků stejného rozměru a pro každý blok se vytváří trojrozměrný histogram. Tím pádem vzniká 500 rozměrný vektor charakteristik. Z jednotlivých vektorů dál vzniká matice charakteristik pro celé video.

Pomocí GEDu autoři odvádějí následující teorém:

Nechť $k \leq r = rank(A)$ a $A_k = \sum_{i=1}^{n} \lambda_i q_i p_i'$, pak máme

$$\min_{rank(B)=k} = \|A - B\|_2 = \|A - A_k\|_2 = \|\lambda_{k+1}\|_2$$

Tento teorém slouží pro nalezení tzv. Euklidové vzdálenosti váženou zobecněnými vlastními čísly:

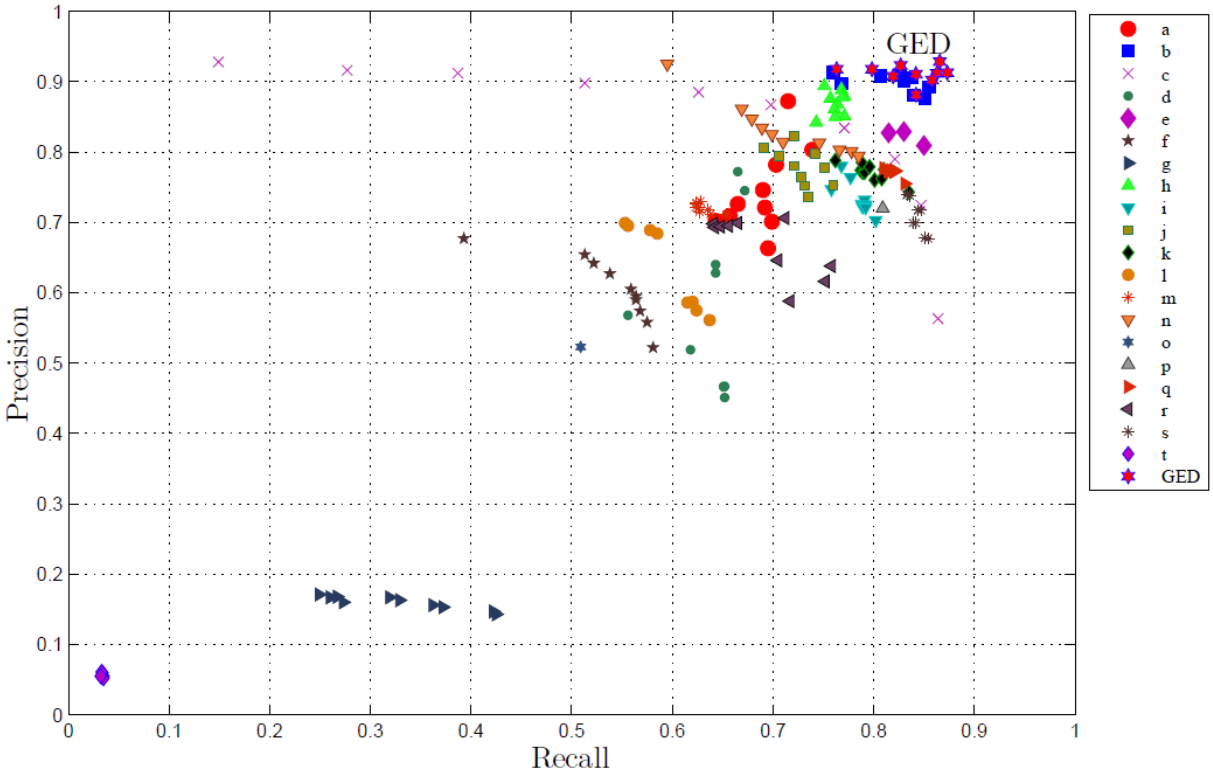
$$\Gamma^k(i) = D^k(p_i', p_{i+1}') = \sqrt{\sum_{h=1}^{k} |\lambda_h|^2 \left|p_{h,i}' - p_{h,i+1}'\right|^2},$$

kde p_i' je sloupec a vektor charakteristik matice P_r^{-1} a k je parametr, který specifikuje, kolik vlastních čísel musíme použit, a který je určen pomocí tréninkových množin dat.

Pro detekci ostrých přechodů používáme konstantní prahovou hodnotu. Pokud Euklidova vzdálenost je větší než prahová hodnota, nalezli jsme přechod. Pro nalezení postupných přechodů se používají další rovnice a algoritmy, které nejsou důležité v rámci této bakalářské práci.

Algoritmus Ali Amiri a Mahmooda Fathy byl testován na datech TRECVIDu 2006 a ukázal dobré výsledky mezi ostatními metodami. Obrázek **2.7** demonstruje výsledky testování. Bohužel, pro realizaci tohoto algoritmu jsem potřeboval víc materiálu od autorů, protože článek neobsahuje plný popis řešení. Kromě toho, během konzultace s pánem Ing. Dostálem jsme

nalezli možné chyby, které překážely v dalším pochopení algoritmu. Zkusil jsem napsat autorům, ale žádnou odpověď jsem nedostal.



Obrázek 2.7 Výsledky TRECVIDu 2006 a výsledky algoritmu Ali Amiri a Mahmooda Fathy, které jsou označené jako GED, zdroj [11]

2.4.4 SHRNUTÍ

Během studia bylo nalezeno, že metoda univerzity Tsinghua se rozvíjela během 3 let a je složitá pro realizaci v rámci bakalářské práce kvůli velkému objemu informací pro studium. Metoda univerzity Bradford také není vhodná, protože používá přímo komprimovaná videa formátu MPEG, což není vhodné pro konečnou aplikaci, protože z funkčních požadavků plyne, že aplikace musí umět pracovat s různými formáty. Algoritmus Ali Amiri a Mahmooda Fathy je dobrý, ale jejich článek neobsahuje dostatek informací. Celkový přehled těchto metod a prostudovaného materiálu ukázal, že většina úspěšných algoritmů využívá porovnání barevných histogramů, detekci a porovnání hran a strojové učení jako moduly pro SBD. Proto jsem se dále zaměřil takovou detekci, která využívá některý z popsaných principů.

2.5 VÝBĚR FRAMEWORKU

Vzhledem k tomu, že v dnešní době existuje velké množství různých kodeků, kontejneru a kompresních schémat, bylo rozhodnuto najít knihovnu, která již umí zpracovat skoro jakékoliv video soubor a provádět nad nim základní operace pro práci s multimédia. Byly zvoleny dvě knihovny, které jsou známější a spolehlivější než jiné.

2.5.1 FFmpeg

FFmpeg je vedoucí multimediální knihovnou, která je schopná dekódovat, kódovat, vysílat, filtrovat a přehrávat skoro všechny známé formáty. Má podporu od nejstarších do nejmodernějších formátů a pořad se rozvíjí. Kromě toho je bezplatná a pro většinu dekodérů má LGPL licenci, která je vhodná pro komerční použiti.

FFmpeg poskytuje různé nástroje:

- ffmpeg – nástroj příkazové řádky, který umožnuje převést multimédia soubory do jiných formátů.
- ffserver – server pro živé vysílání.
- ffplay – jednoduchý přehrávač založený na SDL a FFmpeg knihovnách.
- ffprobe – jednoduchý analyzátor multimediálního potoku.

Navíc, poskytuje sadu knihoven pro programátory:

- libavutil – knihovna pro zjednodušení programování, která obsahuje funkce jako generátor náhodných čísel, datové struktury, běžné matematické operace, základní multimediální nástroje atd.
- libavcodec – knihovna, obsahující kodéry a dekodéry pro audio a video kodeky.
- libavformat – knihovna, obsahující muxery a demuxery pro kontejnery.
- libswscale – knihovna, která provádí měřítko a operace pro konverzi barevných prostorů.

FFmpeg je široce používána a má velkou komunitu a dokumentaci, která urychluje seznámení s knihovnou. Je napsána v C++ a má několik obalů, které jsou v Javě, ale většina nebyla aktualizována skoro 5 let.

2.5.2 OpenCV

OpenCV (Open Computer Vision library) je svobodná a otevřená knihovna, která je pod BSD licenci a slouží pro manipulaci s obrazem. Je zaměřena především na počítačové vidění a zpracování obrazu v reálném čase.

Knihovna má více než 2500 optimalizovaných algoritmů, mezi kterými patří klasické a nejmodernější algoritmy v oblasti počítačového vidění a strojového učení. Tyto algoritmy mohou být použité pro detekci a rozpoznání tváří, identifikaci objektů, klasifikaci lidských činností ve videích, sledování pohybu kamery a objektů, extrakce 3D modelů z objektů atd.

Knihovnu je možné využít z prostředí jazyků C, C++ a Python. OpenCV obsahuje několik různých modulů, sloužících pro různé účely:

- opencv_core — hlavní knihovna. Obsahuje základní datové struktury, matematické operace a lineární algebru, diskrétní Fourierovou transformaci, diskrétní kosinovou transformaci, zpracování vstupů a výstupů pro XML a YAWL atd.
- opencv_imgproc — zpracování obrazů (filtrace, geometrické transformace, konverze barevných prostorů atd.).
- opencv_highgui — jednoduché uživatelské rozhrání pro zobrazení obrazů a videí.
- opencv_ml — modely strojového učení (SVM, rozhodovací stromy atd.).
- opencv_features2d — rozpoznání a popis charakteristik obrazů (FAST, SIFT atd.).

- opencv_video — analýza pohybu a sledování objektů.
- opencv_objdetect — nalezení objektů na obrazech.
- opencv_calib3d — kalibrace kamery, zpracování trojrozměrných dat.
- opencv_gpu — urychlení některých funkci pomocí použití NVidia CUDA.

Pro kódování a dekódování OpenCV používá FFmpeg, je to vlastně nadknihovnou FFmpeg pro zpracování obrazů.

2.5.3 ZÁVĚR

V současné době existuje velmi málo knihoven, které jsou zaměřeny na práci s multimediálními soubory a které podporují skoro všechny typy kodeků, kontejnerů a kompresních schémat. Nejznámější knihovnou je FFmpeg, ale ta má jenom jednoduché operace pro počítačové vidění. Proto byla základní knihovnou zvolena OpenCV, která má obrovské množství operací užitečných pro zpracování obsahu videí.

Kapitola 3

NÁVRH

V rámci bakalářské práce bylo navrženo několik algoritmů. První algoritmus byl vyvinut samostatně po přečtení základních technik pro detekci scén ve videu. Ostatní dva algoritmy jsou ze článků.

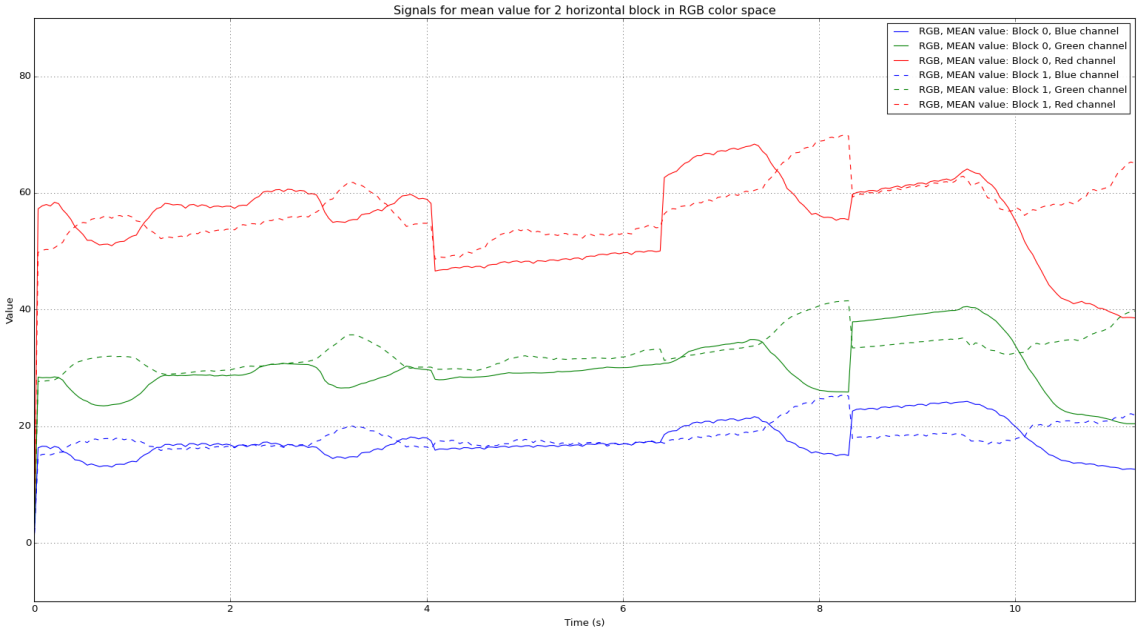
3.1 VLASTNÍ ALGORITMUS

Podle studia materiálů o detekci přechodů mezi scénami, byla zvolena nejjednodušší technika – detekce pomocí porovnání pixelů. Výsledný algoritmus se skládá ze zpracování videa do signálů a z převodu signálů do výsledků pomocí adaptivní prahové hodnoty.

3.1.1 PŘEVOD VIDEA DO SIGNÁLŮ

Každé video obsahuje jednotlivé snímky, ze kterých se skládají scény. Proto můžeme rozdělit video na diskrétní signály, které budou popisovat všechny snímky ve videu.

Základními hodnotami pro zpracování videa byly zvoleny střední hodnota a směrodatná odchylka barvy pixelů pro každý snímek. Abychom uchovali prostorovou charakteristiku, pro každý snímek se určuje množství bloků, do kterého bude snímek rozdělen. Každý snímek je možné zpracovat buď v RGB prostoru nebo v Grayscale(černobílém) prostoru. Tím pádem, pokud máme nastavené zpracování signálu pomocí střední hodnoty v RGB prostoru s 2 bloky pro každý snímek, výstupem zpracování videa bude 8 signálů: střední hodnota červené barvy pro 1 a 2 bloky, střední hodnota zelené barvy pro 1 a 2 bloky a střední hodnota modré barvy pro 1 a 2 bloky. Proto, každý signál obsahuje popis, který se skládá z čísla bloku, názvu barevného kanálu a používaného barevného prostoru a typu používaných hodnot. Obrázek **3.1** demonstruje tento příklad.

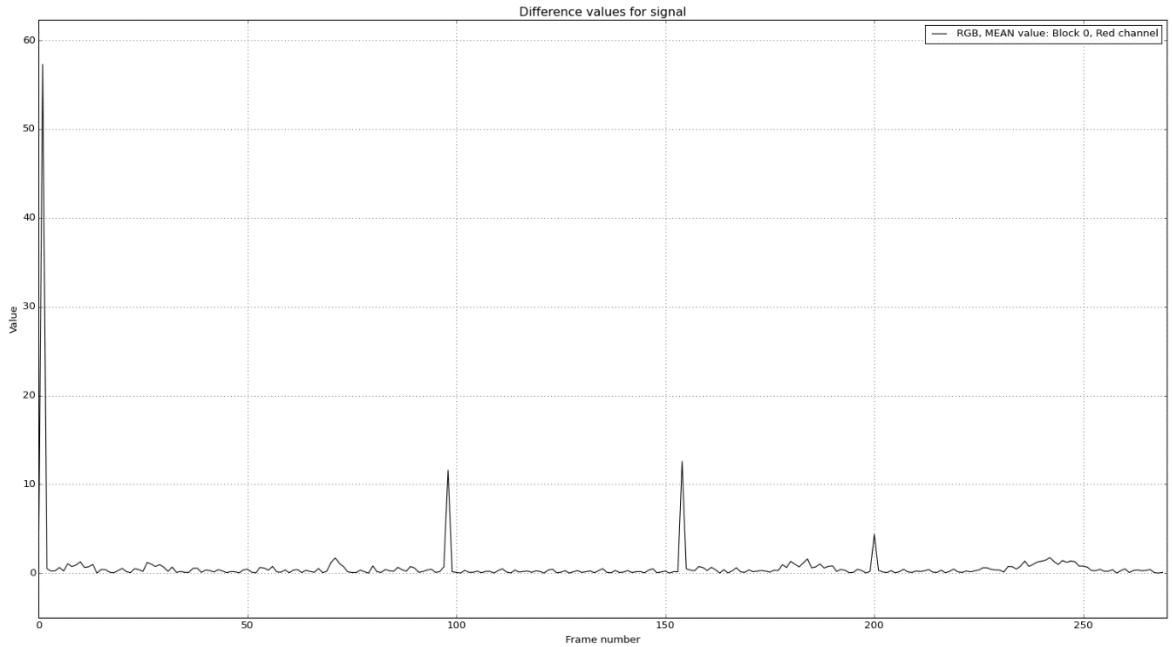


Obrázek 3.1 Signály velikosti 2 bloky na šířku v RGB prostoru se střední hodnotou

Rozdělení do bloků probíhá zvlášť na výšku a zvlášť na šířku. Rozměr každého bloku je určen podílem celkové šířky videa na číslo bloků na šířku a podílem celkové výšky videa na číslo bloků na výšku. Tím způsobem se může ztrácet malá informace od okrajů bloků, pokud šířka nebo výška nejsou dělitelné číslem bloků.

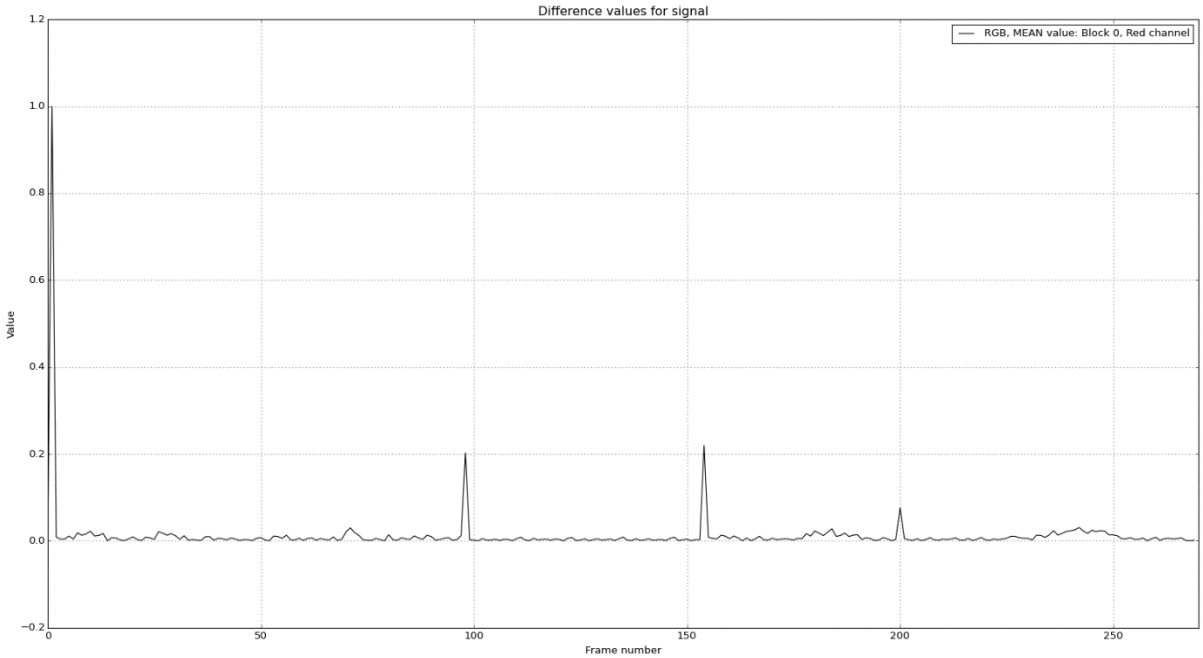
3.1.2 ZPRACOVÁNÍ SIGNÁLŮ

Když video rozdělíme na jednotné signály, pak každý signál se zpracovává zvlášť. Z pole hodnot signálu vytvoříme pole absolutních rozdílů mezi snímky. Obrázek **3.2** znázorňuje příklad.



Obrázek 3.2 Absolutní rozdíly pro signál

Dál normalizujeme pole absolutních rozdílů, aby se bylo možné zbavit od určitých hodnot signálu. Tímto způsobem máme pole rozdílů v každém signálu ve stejném rozmezí, což zjednodušuje zpracování hodnot. **OBRÁZEK 3.3** ukazuje příklad normalizovaného pole absolutních rozdílů .



Obrázek 3.3 Normalizované absolutní rozdíly

Pomocí adaptivní prahové hodnoty pak vypočítáme časové výsledky pro signál a tyto výsledky uložíme do množiny výsledků, která obsahuje výsledky všech signálů.

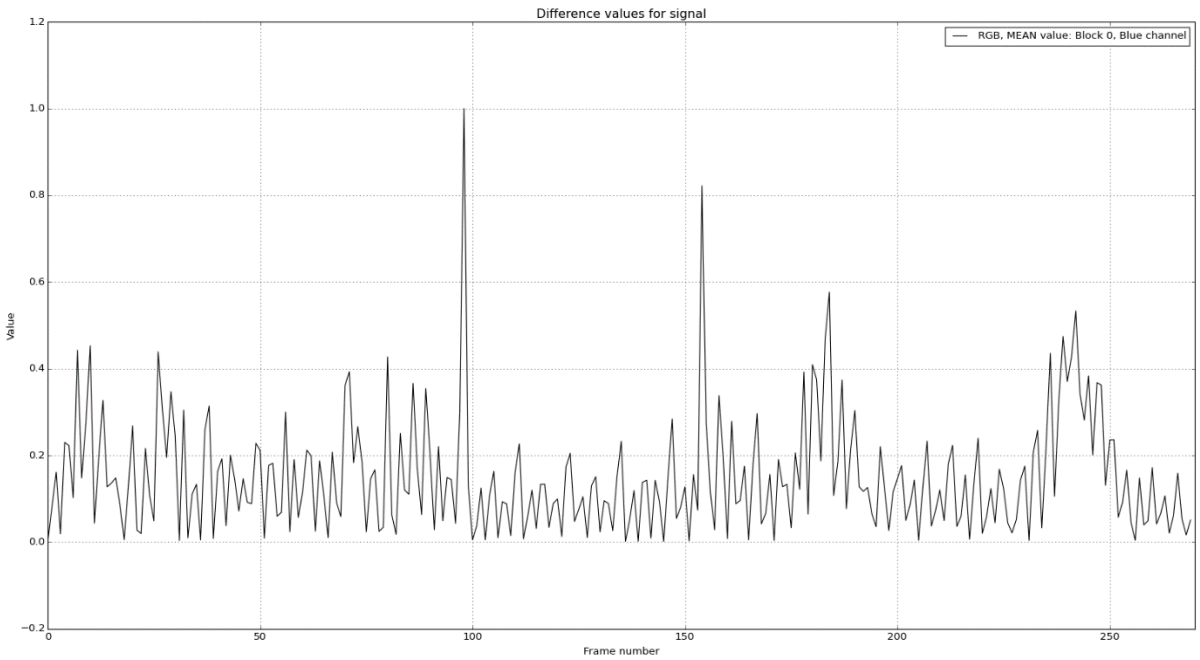
3.1.3 ADAPTIVNÍ PRAHOVÁ HODNOTA

Prahová hodnota se určuje rovnicí:

$$s = \frac{\sum_{i=0}^{n} diff(i)}{n}$$

Kde s je prahová hodnota, $diff(i)$ je hodnota *i*-tého prvku normalizovaného pole absolutních rozdílů a n je délka pole.

V každém cyklu zapisujeme do výsledků index hodnoty pole, která se rovná 1, pak tuto hodnotu měníme na jednu polovinu součtu předchozí a následující hodnoty, abychom ne vytvořili nový skok mezi hodnotami. Dál znovu normalizujeme pole a počítáme prahovou hodnotu. Cyklus se iteruje, pokud prahová hodnota je menší než konstantní hodnota P, která je stejná pro všechna videa. **OBRÁZEK 3.4** demonstruje graf pole, když cyklus skončil a konstantní hodnota se rovnala 0.08.



Obrázek 3.4 Normalizované absolutní rozdíly v posledním cyklu

Pokud používáme více než 1 signál pro detekci přechodů, pak sjednocujeme výsledky z každého signálu. Nejprve se všechny výsledky ukládají do celkové množiny časových hodnot. Potom zjišťujeme, kolik signálů má každou časovou hodnotu ve výsledcích. Časová hodnota se ukládá do výstupní množiny, pokud je splněna tato podmínka:

$$\lambda \geq \lfloor \beta * length_{signals} + 0{,}5 \rfloor,$$

kde λ je počet výskytů časové hodnoty ve výsledcích jednotlivých signálů, $length_{signals}$ je množství signálů a β je konstanta pro sjednocení.

3.1.4 ZÁVĚR

Tento algoritmus používá základní techniky detekce střihu ve videu, které jsou nazývány pixel-to-pixel. To znamená, že v podstatě porovnává hodnoty pixelů mezi sebou. Je to nejjednodušší způsob zpracování signálů, je nedokonalý vůči šumu, blesků a pohybu kamery nebo objektů ve scéně. Ale je schopen detekovat správné přechody mezi scénami.

3.2 ALGORITMUS DETEKCE POMOCÍ JND HISTOGRAMŮ

Informace ohledně daného algoritmu byla nalezena ve článku [17]. Přehled existujících řešení ukázal, že většina úspěšných algoritmů využívá porovnání barevných histogramů, detekce a porovnání hran a strojové učení jako moduly pro SBD. Proto byl zvolen tento algoritmus, který je založen na porovnání barevných histogramů.

3.2.1 JND HISTOGRAM

Jádrem algoritmu je tzn. JND histogram. To je histogram, který je v JND barevném modelu. JND zkrátka znamená Just Noticeable Difference, což doslovně je Jen Znatelný Rozdíl.

JND barevný model je model, který je v barevném prostoru RGB a je založen na omezeních vnímání lidského oka. Lidská sítnice obsahuje dva typy světločivých buněk: tyčinky a čípky. Tyčinky jsou zodpovědné za vnímání kontrastů (černobílé, mlhavé vidění při nízkém osvětlení). Čípky umožňují barevné vnímání a mají tři základní typy podle určitého rozmezí vlnových délek: červený, zelený a modrý. Podle teorie Thomasa Younga se všechny barvy dají rozložit na lineární kombinace těchto třech barev. Průměrné lidské oko dokáže vnímat 17 000 barev při maximální intenzitě bez nasycení oka. Jinými slovy, pokud velký barevný prostor je vzorkován jenom do 17 000 barev, můžeme dosáhnout stejného výkonu jako lidský zrak při normálním osvětlení. Člověk je schopen rozlišit dvě barvy, pokud tyto barvy mají alespoň jeden „jen znatelný rozdíl“. Tím vzniká pojem JND a používá se jako jednotka měřící rozdíl mezi barvami.

Pokud se rozhodneme rozdělit 17 000 barev do červené, zelené a modré, budeme potřebovat přibližně 26 hodnot pro každou osu. Ale z fyziologického hlediska zelené čípky lidské sítnice jsou nejcitlivější, pak jdou modré čípky a červené čípky. Proto červená osa barevného modelu je rozdělena do 24 úrovni, zelená je rozdělena do 28 úrovní a modrá je rozdělena do 26 úrovní, což celkově vychází 17 472 barvy. Tím pádem, JNG histogram má 17 472 řádků.

Pro výpočet JND histogramu je nutné vědět jak ukládat hodnoty do histogramu. Pro tento účel je daná rovnice výpočtu indexu v histogramu pomocí hodnot RGB barevného prostoru, které máme na vstupu:

$$Color\ index = \left(R_{JND} * G * B + G_{JND} * B + B_{JND}\right) + 1$$

kde R_{JND}, G_{JND} a B_{JND} jsou kvantované R, G a B hodnoty pixelu ve videu. Tabulka **3.1** demonstruje, jak vypadá struktura výsledného JND histogramu.

Struktura JND histogramu		Hodnoty barevného prostoru RGB		
Index	Počet pixelů	R	G	B
1	34	0	0	0
2	453	0	0	1
...	...	...	...	...
26	349	0	0	25
...	...	...	...	...
17471	120	23	27	25

Tabulka 3.1 Struktura JND histogramu

Pomocí tohoto vzorkování můžeme přenést obrázek z RGB prostoru do barevného prostoru JND, který pak používáme pro extrakce charakteristik.

3.2.2 EXTRAKCE CHARAKTERISTIK

Podobnost dvou snímků může být spočítána pomocí hodnot barevných binů:

$$Simi_{JND}(X,Y) = \sum_{i=1}^{C} Min(h_x(i), h_y(i))$$

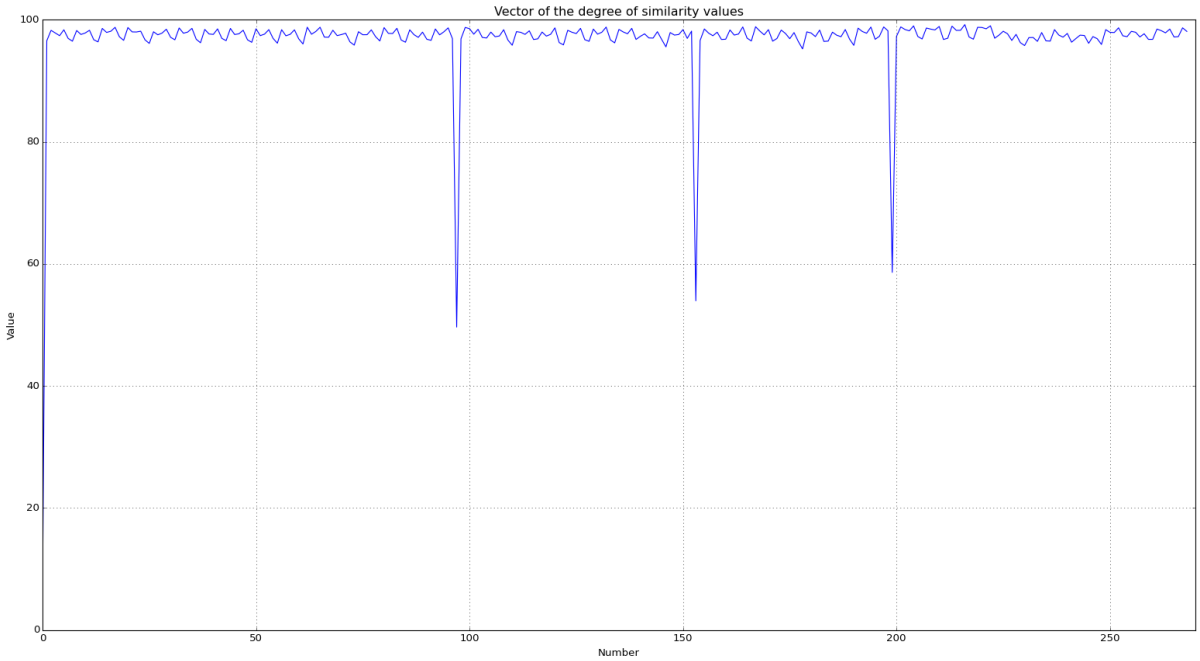
Kde h_x a h_y jsou JND histogramy snímků X a Y a C je počet barevných binů v histogramu. Tato podobnost poskytuje celkový počet pixelů, které jsou přítomné v obou snímcích podle porovnání.

Míra podobnosti se dá spočítat pomocí rovnice:

$$DS_{JND}(i,j) = Simi_{JND}(i,j)/((m*n)/100)$$

Kde $(m*n)$ je rozměr snímku. Míra podobnosti nám poskytuje procentuálně vyjádřenou podobnost snímků i a j.

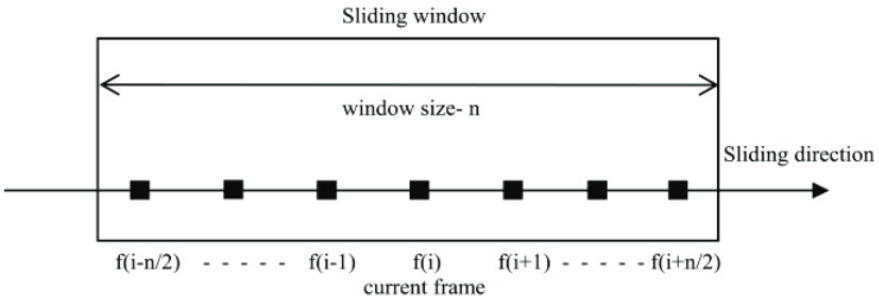
V posledním kroku extrakce máme jednorozměrný vektor charakteristik velikosti N-1, obsahující míry podobnosti spočítané pro N snímků ve videu. **OBRÁZEK 3.5** ukazuje, jak vypadá tento vektor.



Obrázek 3.5 Výstupný vektor charakteristik, obsahující míry podobnosti snímků

3.2.3 ADAPTIVNÍ PRAHOVÁ HODNOTA

V algoritmu JND histogramů je zvolena adaptivní prahová hodnota, která je založena na metodě spočítání prahové hodnoty pomocí lokálního okna. Obrázek **3.6** znázorňuje posuvné okno se středem v aktuálním snímku.



Obrázek 3.6 Posuvné okno, zdroj [17]

Jak již bylo napsáno výše, vstupem pro další krok je jednorozměrný vektor velikosti N-1. Pomocí něho počítáme střední hodnotu míry podobnosti pro snímky, které jsou uvnitř okna. Kromě toho je nutné najít minimální hodnotu a druhou minimální hodnotu v posuvném okně. Minimální míra podobnosti vyjadřuje největší rozdíl mezi snímky podle porovnání.

Úplný algoritmus používá dvě prahové hodnoty: horní a spodní. Horní prahová hodnota slouží pro detekci ostrých přechodů a spodní je určena pro detekci postupných přechodů. V rámci této bakalářské práce je použita jenom horní prahová hodnota, protože ve funkčních požadavcích není možnost detekce postupných přechodů. Postup detekce náhlých přechodů je inspirován článkem B. L. Yeo a B. Liu [18]. Pro každou pozici v okně ověřujeme přítomnost přechodu podle daného kritéria:

$$Pokud\ DS(i,i+1) = \max_{k=-\frac{n}{2},\dots,\frac{n}{2}} \left(\forall DS(i+k,i+1+k)\right) \wedge DS(i,i+1) \leq \alpha S_{low}$$

$$\mu \geq \alpha S_{low} \ \gg \text{ostrý přechod}$$

kde α je konstanta, která typicky má hodnotu v intervalu od 1.1 do 2.0. Jinak řečeno, mezi snímky i a i+1 je ostrý přechod, pokud je splněna každá z těchto podmínek:

1) míra podobnosti je nejmenší v okně
2) míra podobnosti je menší nebo se rovná součinu α a druhé minimální hodnoty
3) střední hodnota posuvného okna je větší nebo se rovná součinu α a minimální míry podobnosti.

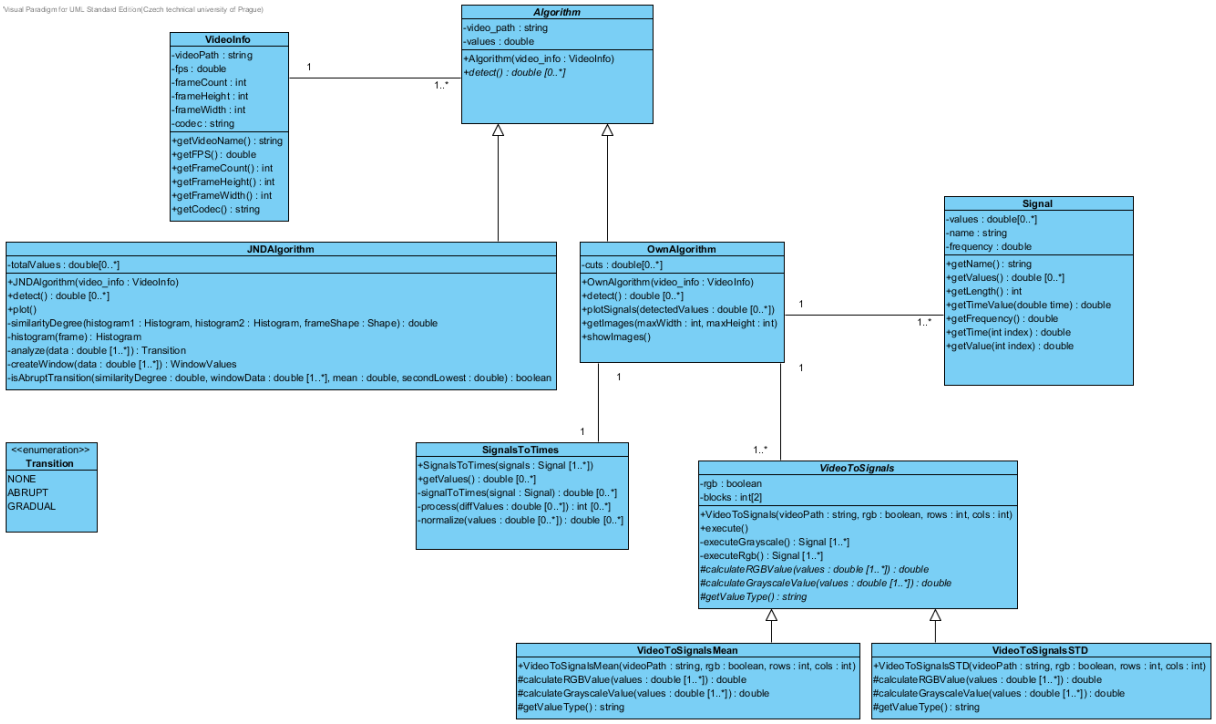
3.2.4 ZÁVĚR

Algoritmus JND histogramů používá málo známý barevný prostor JND, který ale šetří místo v paměti a umožnuje rychlejší zpracování histogramů pro snímky. Tento algoritmus byl zvolen kvůli použití porovnání histogramů a autoři tvrdí, že má perfektní výsledky. Výkon algoritmu je silně závislý na rozměru posuvného okna a konstantě α.

3.3 NÁVRH ARCHITEKTURY

Výsledek této bakalářské práce by měl být použit jako modul již navržené aplikace. Proto bylo zvoleno modulární programovací paradigma, jehož základním prvkem je modul. Cílem tohoto paradigmatu je zjednodušení projektování a rozdělení procesu vývoje mezi skupinami vývojářů.

Uvnitř modulu se používá objektově orientované programovací paradigma. Modul je rozdělen do několika balíčků, kde hlavním je balíček obsahující algoritmy. Obrázek **3.7** demonstruje diagram tříd, kde můžeme vidět celkovou architekturu tohoto balíčku.



Obrázek 3.7 Diagram Tříd

Pro spojení dvou algoritmu se používá návrhový vzor Strategie. Strategie je návrhovým vzorem, tykajícím se chování, a slouží k vyměňování různých implementací algoritmu za běhu programu. Tato záměna může proběhnout buď explicitně (na žádost klienta) nebo implicitně (na základě nastavení ovladače). Jádrem tohoto vzoru je třída Context, která slouží jako kontejner na jednotlivé postupy. V momentě, kdy dojde k potřebě změny chování, Context prohodí objekty reprezentující dané algoritmy. V daném případu, Context je hlavní soubor, který spouští detekce přechodů, třída Algorithm je Strategií a záměna algoritmů probíhá explicitně.

Navíc, Strategie se používá u vlastního algoritmu, který má dva hlavní typy zpracování signálů: pomocí střední hodnoty nebo pomocí směrodatné odchylky. Třída OwnAlgoritm je Contextem a abstraktní třída VideoToSignals je Strategií. Záměna typu zpracování signálů probíhá implicitně podle nastavení ve třídě OwnAlgorithm.

Celkový návrh architektury se snažil dodržet Funkcionální design, který garantuje, že každý modul aplikace má jenom jednu zodpovědnost a nízkou závislost na ostatních modulech aplikace.

Kapitola 4

IMPLEMENTACE

Na začátku práce byl pro implementaci zvolen programovací jazyk C++, později byl zvolen jazyk Python. Hlavním důvodem změny byla integrita systému a možnost jednoduché údržby kódu, protože celý systém byl napsán v jazyce Python.

Základní knihovnou pro práci s obrazy a videi je OpenCV, která je podrobně popsána v analýze této bakalářské práce. OpenCV má závislost na modulech Numpy a Matplotlib, které jsou taký použity v implementaci. Kvůli tomu, že OpenCV podporuje Python jenom verzí 2.7.x, byla zvolena verze Pythonu 2.7.8.

OpenCV nemá vestavěný kodek v instalačním balíčku, ale používá knihovnu kodeků libavcodec z knihovny FFmpeg. Pro správný běh aplikace byl naistalován K-LIte Codec Pack, který taký používá libavcodec a má ji uvnitř.

Pro čtení a zápis do souborů pomocí xml (eXtensible Markup Language) jazyka byla použita standartní knihovna Pythonu etree.

Kapitola 5

TESTOVÁNÍ

Testování algoritmů probíhalo na vybrané množině videí. Tato množina se podle zadání skládá ze třech kategorií:

- video pořízené amatérskou kamerou v exteriéru
- video obsahující reklamní TV vstup
- video zachycující sportovní přenos

Každá kategorie obsahuje tři videa, která se liší kodeky, kontejnery, rozlišeními, délkou, povětrnostními podmínkami nebo jinými faktory, které mají vliv na detekci přechodů, jako, například, blesky, rychlý pohyb kamery nebo objektu atd. Krátkou informaci ohledně každého videa popisuje Tabulka **5.1**.

Název	Kategorie	Kodek	Kontejner	Rozlíšení	FPS
cars	amateur	mp4v	3GP	352x288	15
night		avc1	MP4	1080x1920	29.97
signboards		avc1	MOV	720x576	29.97
edison	TV	avc1	MP4	1280x720	25
genertel		MP42	AVI	1280x720	25
nike		H.264	MKV	1280x720	23.976
football	sport	WMV3	WMV	1280x720	25
hockey		mp4v	3GP	352x288	15
volleyball		avc1	MP4	1280x720	29.97

Tabulka 5.1 Popis množiny videí

Některá videa byla uměle exportována do jiného kontejneru pomocí Adobe Premiere Pro CC 2014 a webové služby http://video.online-convert.com/ru , abych mohl otestovat různé případy kontejnerů. Maximální délka videa je cca 10 minut, což je požadavkem systému. Výběr videí ještě byl ovlivněn množstvím ostrých přechodů, ale některá obsahují postupné přechody, aby bylo možné zjistit, jak na to reagují algoritmy. Tabulka **5.2** demonstruje množství přechodů každého typu ve videích.

Název	Snímky	Ostré přechody	Postupné přechody
cars	9036	17	0
night	1979	16	0
signboards	1660	24	0
edison	1500	29	0
genertel	750	10	0
nike	1956	48	2
football	6355	62	3
hockey	8473	100	0
volleyball	17737	63	1

Tabulka **5.2** Charakteristika videí

5.1 HODNOTY MĚŘENÍ

Základními hodnotami pro měření výsledků byly zvoleny hodnoty, které se typicky používají pro hodnocení výsledků metod detekce přechodů: přesnost (P) a zisk (R).

Nález (N) je správně detekovaný přechod, zmeškaný nález (Z) je přechod, který nebyl detekován algoritmem, a chybný nález (C) je špatně detekovaný přechod. Pomocí těchto nálezů můžeme pak spočítat přesnost a zisk. Pro hodnocení výsledků byla navíc použita míra F1, která je kombinací přesnosti a zisku.

Přesnost je podílem správně detekovaných přechodů na celkové množství detekovaných přechodů:

$$P = N/(N + C)$$

Přesnost popisuje proporce správných nálezů z celkového množství nálezů.

Zisk určuje proporce správných nálezu z celkového množství přechodů:

$$R = N/(N + Z)$$

Obecně, přesnost a zisk kompletně popisují výkon algoritmů. Ale parametry algoritmu mohou být změněny, aby algoritmus měl vysokou přesnost, což snižuje zisk. Na druhé stráně, můžeme změnit parametry pro dosažení vysokého zisku, ale to často snižuje přesnost. Cílem je najít zlatou střední cestu mezi dobrou přesností a dobrým ziskem. Existuje mnoho způsobů, jak sjednotit zisk a přesnost do jedné hodnoty. Jednou z těchto možností je míra F1, která je harmonickým průměrem a přikládá stejný význam jak pro přesnost, tak i pro zisk:

$$F1 = 2PR/(P + R)$$

5.2 VÝSLEDKY

Pro dosažení dobrých výsledků byly vygenerovány vhodné konstanty pro každý algoritmus. Mírou pro hodnocení konstanty byla F1. Pro každou hodnotu konstanty probíhalo testování celé kolekce. Pokud F1 pro danou hodnotu byla větší než předchozí F1, pak tato hodnota je kandidátem.

Pro porovnání rychlosti dvou algoritmů v tabulkách výsledků je uveden čas. Čas byl naměřen ve stejných podmínkách na zdroje, majícím 64 bitový operační systém Windows 8.1 Pro, procesor Intel Core i5-4570 3.20 GHz, operační paměť 8 GB a videokartu GeForce GTX 760. Samozřejmě, tato hodnota je orientační a slouží pouze pro náhled rychlosti algoritmů.

5.2.1 JND ALGORITMUS

Pro JND algoritmus byly testovány konstanty α a rozměr posuvného okna. Nejlepšími pro danou množinu videí byly zvoleny hodnoty 1.3 pro α a 5 pro rozměr posuvného okna. Tabulka **5.3** ukazuje výsledky testování s použitím těchto hodnot.

Video	N	Z	C	P (%)	R (%)	F1 (%)	Čas (s)
cars	17	0	41	29,31	100	45,33	145
night	7	9	0	100	43,75	60,87	52
signboards	24	0	0	100	100	100	30
edison	27	2	0	100	93,10	96,43	32
genertel	9	1	0	100	90	94,74	16
nike	44	6	0	100	88	93,62	40
football	60	5	2	96,77	92,31	94,49	163
hockey	74	26	0	100	74	85,06	133
volleyball	62	2	0	100	96,88	98,41	357
Celkově	324	51	43	91,79	86,45	85,44	968

Tabulka 5.3 Výsledky testování JND algoritmu

5.2.2 VLASTNÍ ALGORITMUS

Pro vlastní algoritmus byla testována jenom jedna konstanta P. Zvolil jsem manuálně 4 bloky pro každý signál a množina signálů se skládala ze dvou signálů střední hodnoty a směrodatné odchylky v černobílém barevným prostoru a z jednoho signálu směrodatné odchylky v barevném prostoru RGB. Pro danou množinu byla zvolena hodnota 0,05. Tabulka **5.4** demonstruje výsledky testování vlastního algoritmu.

Video	N	Z	C	P (%)	R (%)	F1 (%)	Čas (s)
cars	11	6	8	57,89	64,71	61,11	23
night	14	2	1	93,33	87,5	90,32	145
signboards	17	7	1	94,44	70,83	80,95	21
edison	28	1	6	82,35	96,55	88,89	47
genertel	8	2	1	88,89	80	84,21	25
nike	46	4	2	95,83	92	93,88	66
football	53	12	25	67,95	81,54	74,13	316
hockey	89	11	12	88,12	89	88,56	36
volleyball	62	2	20	75,61	96,88	84,93	594
Celkově	328	47	76	82,71	84,33	83	1273

Tabulka 5.4 Výsledky testování vlastního algoritmu

5.3 ZÁVĚR

Algoritmus JND ukázal lepší výsledky pro celou množinu videí. Navíc je rychlejší než můj vlastní algoritmus. Největší problémy se projevily hlavně v detekcích přechodů u videí, pořízených amatérskou kamerou v exteriéru: cars a night videa. Video cars bylo natočeno při jasném počasí, což způsobilo stálé blikání slunce. Tímto může být ovlivněn velký počet chybných nálezů ve výsledcích, protože algoritmus detekoval blikání jako nový přechod. Video night bylo natočeno, když byl večer a při malém osvětlení. Proto, algoritmus nedetekoval všechny nálezy kvůli malému rozdílu mezi barvami snímků. V kategorii sport JND měl největší potíže s detekcí přechodů ve videu hockey, které bylo natočeno jenom z jedné pozice. Je tam stejný problém jako u night, malý rozdíl mezi barvami neumožnil najít všechny ostré přechody.

Vlastní algoritmus měl taký dobré výsledky, avšak chybných nálezů bylo skoro o dvakrát víc, než u JND algoritmu. Algoritmus měl největší potíže s kategorií sport. Je to ovlivněno rychlým pohybem kamery nebo objektů ve videu, které algoritmus rozpoznává jako ostré přechody.

Oba algoritmy nedetekovaly postupné přechody a nedekovaly kvůli nim chybné nálezy.

Kapitola 6

ZÁVĚR

6.1 Dosažené výsledky

V rámci této bakalářské práce byl nastudován přehled existujících kodeků, kompresních schémat a formátů kontejnerů, který pak sloužil pro studium knihoven, které umějí zpracovávat různé formáty videa. Přehled nástrojů, které používají metody detekce přechodů, ukázal, jak rychle a přesně dnešní aplikace mohou rozpoznávat jednotlivé přechody a jakým způsobem to dělají. Pro hlubší analýzu byly nastudovány nejefektivnější metody v oblasti detekce přechodů, které jsem v této práci nemohl použit z různých důvodů.

Byly vyzkoušeny dvě metody detekce: vlastní algoritmus a algoritmus JND histogramů. Během testování se zjistilo, že JND algoritmus je lepší než vlastní a je rychlejší. Avšak vlastní algoritmus ukázal také dobré výsledky a může být použit v komerčním prostředí.

Všechny funkční a nefunkční požadavky byly splněny a aplikace se dnes používá pro účely policie.

6.2 Možná budoucí rozšíření

Aplikace může být rozšířena v různých směrech. Na jedné stráně může být přidána detekce postupných přechodů jako jednotlivý modul, který bude používat jinou metodu detekce přechodů nebo může být přidána detekce přechodů ze článku o JND algoritmu [17]. Druhá možnost je zlepšit již existující detekce ostrých přechodů pomocí přidání detekce blikání, detekce změny hran nebo modulu, který bude používat strojové učení.

Architektura aplikace umožnuje jednoduše přidávat další algoritmy, proto mohou být například, realizovány nejlepší algoritmy detekce z TRECVIDU jako metoda univerzity Tsinghua nebo jiné algoritmy.

Kapitola 7

LITERATURA

[1] HandySaw DS description. http://www.davisr.com/en/products/handysaw/description.htm, stav ze 10.11.2014

[2] The Digital Format. http://www.digitalpreservation.gov, stav ze 10.11.2014

[3] Výzkum společnosti Sorenson Media. http://www.sorensonmedia.com/sorenson-media-survey-reveals-mp4-leads-wide-range-of-formats-used-for-web-and-mobile-viewing-increasingly-hands-on-role-of-video-professionals-worldwide/, stav ze 10.11.2014

[4] AVCutty description: http://www.avcutty.de/english/index_info.htm, stav ze 20.11.2014

[5] Wondershare Video Editor description: http://www.wondershare.com/video-editor/, stav ze 20.11.2014

[6] Limitations of the trial version for Video Editor: http://support.wondershare.com/how-tos/what-are-the-limitations-of-the-trial-version-for-video-editor.html, stav ze 20.11.2014

[7] History of WonderShare Video Editor. http://support.wondershare.com/video-editor/history.html, stav ze 20.11.2014

[8] TRECVID. http://trecvid.nist.gov/, stav ze 25.11.2014

[9] Alan F. Smeaton, Paul Over and Aiden R. Doherty. *Video Shot Boundary Detection: Seven Years of TRECVid Activity*. In: Computer Vision and Image Understanding. March 2009, Elsevier, Netherlands 2009, vol. 114, issue 4, pp. 411-418

[10] Alan F. Smeaton, Paul Over. TRECVID-2007:Shot Boundary Detection Task Summary [online]. 2007. [cit. 28. 11. 2014]. Dostupné z: http://www-nlpir.nist.gov/projects/tvpubs/tv7.slides/tv7.sb.slides.pdf

[11] Ali Amiri, Mahmood Fathy. *Video Shot Boundary Detection Using Generalized Eigenvalue Decomposition and Gaussian Transition Detection*. In: Computing and Informatics. [online]. 2011. [cit. 28. 11. 2014]. Dostupné z: http://www.cai.sk/ojs/index.php/cai/article/view/185/156

[12] Jinhui Yuan, Huiyi Wang, Lan Xiao, Dong Wang, Dayong Ding, Yuanyuan Zuo, Zijian Tong, Xiaobing Liu, Shuping Xu, Wujie Zheng, Xirong Li, Zhangzhang Si, Jianmin Li, Fuzong Lin, Bo Zhang. *Tsinghua University at TRECVID 2005* [online]. 2005. [cit. 30.11.2014]. Dostupné z: http://www-nlpir.nist.gov/projects/tvpubs/tv5.papers/tsinghua.pdf

[13] Jie Cao, Yanxiang Lan, Jianmin Li, Qiang Li, Xirong Li, Fuzong Lin, Xiaobing Liu, Linjie Luo, Wanli Peng, Dong Wang, Huiyi Wang, Zhikun Wang, Zhen Xiang, Jinhui Yuan, Bo Zhang, Jun Zhang, Leigang Zhang, Xiao Zhang, Wujie Zheng. *Intelligent Multimedia Group of Tsinghua University at TRECVID 2006* [online]. 2006. [cit. 30.11.2014]. Dostupné z: http://www-nlpir.nist.gov/projects/tvpubs/tv6.papers/tsinghua.pdf

[14] Jinhui Yuan, Zhishan Guo, Li Lv, Wei Wan, Teng Zhang, Dong Wang, Xiaobing Liu, Cailiang Liu, Shengqi Zhu, Duanpeng Wang, Yang Pang, Nan Ding, Ying Liu, Jiangping Wang, Xiujun Zhang, Xiaozheng Tie, Zhikun Wang, Huiyi Wang, Tongchun Xiao, Yinyu Liang, Jianmin Li, Fuzong Lin, Bo Zhang, JianGuo Li, WeiXin Wu, XiaoFeng Tong, DaYong Ding, YuRong Chen, Tao Wang, Yimin Zhang. *THU and ICRC at TRECVID 2007* [online]. 2007. [cit. 30.11.2014]. Dostupné z: http://www-nlpir.nist.gov/projects/tvpubs/tv7.papers/thu-icrc.pdf

[15] Jinchang Ren, Jianmin Jiang, Juan Chen. *Determination of Shot Boundary in MPEG Videos for TRECVID 2007* [online]. 2007. [cit. 30.11.2014]. Dostupné z: http://www-nlpir.nist.gov/projects/tvpubs/tv7.papers/bradford.pdf

[16] Rainer Lienhart. *Comparison of automatic shot boundary detection algorithms*. In: SPIE Image and Video Processing VII. Jan. 1999, vol. 3656, pp. 290-301.

[17] Nitin J. Janwe and Kishor K. Bhoyar. *Video shot boundary detection based on JND color histogram*. In: Image Information Processing (ICIIP), 2013 IEEE Second International Conference on. Dec. 2013, pp. 476-480.

[18] Boon-Lock Yeo and B. Liu. *Rapid scene analysis on compressed video*. In: Circuits and Systems for Video Technology, IEEE Transactions on. Dec. 1995, vol. 5, issue 6, pp. 533-544.

Příloha A

SEZNAM POUŽITÝCH ZKRATEK

TRECVID TREC Video Retrieval Evaluation

TREC Text REtrieval Conferrence

UML Unified Modeling Language

HVS Human Visual System

GNU GNU's Not Unix!

LGPL Lesser General Public License

GPL General Public License

BSD Berkeley Software Distribution

DCT Discrete cosine transform

MPEG Moving Picture Experts Group

3GPP Third Generation Partnership Project

AVI Audio Video Interleaved

EBML Extensible Binary Meta Language

DV Digital Video

FOI Fade Out/In

SIFT Scale invariant feature tranform

GED Generalized Eigenvalue Decomposition

SBD Shot Boundary Detection

XML eXtensible Markup Language

YAWL Yet Another Workflow Language

SVM Support Vector Machine

JND Just Noticeable Difference

OpenCV Open Computer Vision library

FPS Frames Per Second

Příloha B

SEZNAM TESTOVACÍCH VIDEÍ

V této příloze jsou podrobnější popsány charakteristiky testovacích videí a jejich konfigurace, pokud video bylo konvertováno do jiného kodeku či kontejneru.

B.1 KATEGORIE AMATEUR

Všechna videa v této kategorii byla natočena samostatně, proto se neuvádí zdroj.

B.1.1. CARS

Rozlišení: 352x288

Délka: 10m 2s

FPS: 15

Kontejner: 3GP

Kodek: MP4V

Obrázek:

Obrázek B.1 Cars

B.1.2. NIGHT

Rozlišení: 1080x1920

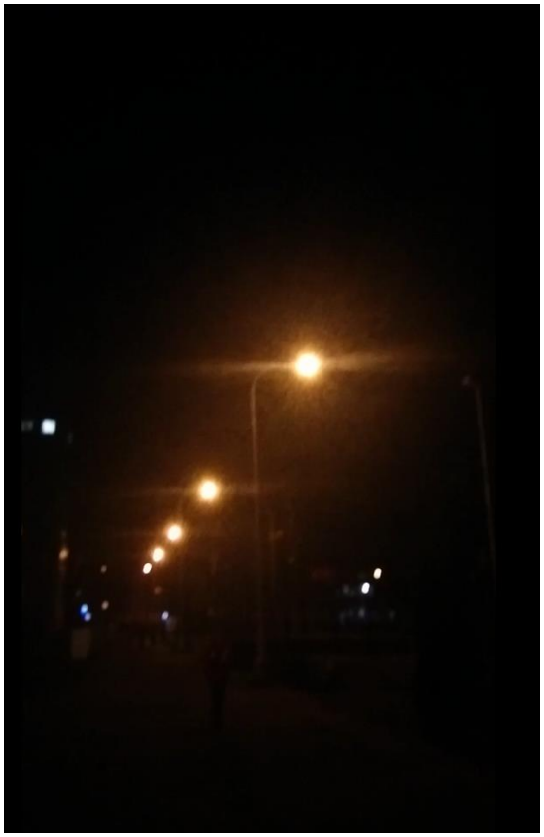
Délka: 1m 6s

FPS: 29.97

Kontejner: MP4

Kodek: AVC1

Obrázek:



Obrázek B.2 Night

B.1.3. SIGNBOARDS

Rozlišení: 720x576

Délka: 55s

FPS: 29.97

Kontejner: MOV

Kodek: AVC1

Obrázek:

Obrázek B.3 Signboards

Editováno pomocí: WonderShare Video Editor

B.2 Kategorie TV

B.2.1. Edison

Rozlišení: 1280x720

Délka: 1m

FPS: 25

Kontejner: MP4

Kodek: AVC1

Obrázek:

Obrázek B.4 Edison

Zdroj originálu: http://vimeo.com/79281950

B.2.2. GENERTEL

Rozlišení: 1280x720

Délka: 30s

FPS: 25

Kontejner: AVI

Kodek: MP42

Obrázek:

Obrázek B.5 Genertel

Editováno pomocí: http://video.online-convert.com/ru/convert-to-avi

Zdroj originálu: http://vimeo.com/79280937

B.2.3. NIKE

Rozlišení: 1280x720

Délka: 1m 21s

FPS: 23.976

Kontejner: MKV

Kodek: H.264

Obrázek:

Obrázek B.6 Nike

Editováno pomocí: http://video.online-convert.com/ru/convert-to-mkv

Zdroj originálu: http://vimeo.com/114272132

B.3 Kategorie sport

B.3.1. Fotball

Rozlišení: 1280x720

Délka: 4m 14s

FPS: 25

Kontejner: WMV

Kodek: WMV3

Obrázek:

Obrázek B.7 Football

Editováno pomocí: Adobe Premiere Pro CC 2014

Zdroj originálu: http://vimeo.com/67252467

B.3.2. HOCKEY

Rozlišení: 352x288

Délka: 9m 24s

FPS: 15

Kontejner: 3gp

Kodek: MP4V

Obrázek:

Obrázek B.8 Hockey

Editováno pomocí: Adobe Premiere Pro CC 2014

Zdroj originálu: http://vimeo.com/1847041

B.3.3. VOLLEYBALL

Rozlišení: 1280x720

Délka: 9m 51s

FPS: 29.97

Kontejner: mkv

Kodek: AVC1

Obrázek:

Obrázek B.9 Volleyball

Editováno pomocí: Adobe Premiere Pro CC 2014

Zdroj originálu: http://vimeo.com/89779901

Příloha C

OBSAH PŘILOŽENÉHO DVD

doc	
html	Dokumentace projektu v html
pdf	Dokumentace projektu v pdf
inst	
win32	Instalační balíček pro Windows 32bit
win64	Instalační balíček pro Windows 64bit
src	
SceneDetector	Zdrojový kód aplikace
test_files	Soubory xml, obashující manuálně detekované přechody
video	Kolekce videí pro testování
text	
bp_primaale_2015.docx	Text bakalářské práce ve Wordu
bp_primaale_2015.pdf	Text bakalářské práce
README.txt	Soubor s návodem k instalaci

Obrázek C.10 Obsah přiloženého DVD