VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INFORMAČNÍCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INFORMATION SYSTEMS

DIFF PRO MULTIMEDIÁLNÍ DOKUMENTY

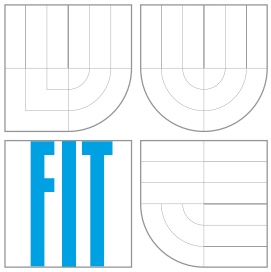
DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE
AUTHOR

Bc. JOZEF LANG

BRNO 2012

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INFORMAČNÍCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INFORMATION SYSTEMS

DIFF PRO MULTIMEDIÁLNÍ DOKUMENTY

MULTIMEDIA DOCUMENT TYPE DIFF

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE
AUTHOR

Bc. JOZEF LANG

VEDOUCÍ PRÁCE
SUPERVISOR

Ing. PETR CHMELAŘ

BRNO 2012

Abstrakt

Rozvoj internetu a jeho masové rozšíření přineslo zvýšení objemu multimediálnich dat. S nárůstem množství dat stoupá potřeba efektivního zjišťování podobnosti mezi různymi multimediálními soubory za účelem prevence a odhalování porušování autorských licencí nebo jen v rámci zjišťování duplikátních nebo jinak podobných souborů. Tato práce se zabývá současnými možnostmi v oblasti porovnávaní obrazových dat na základě jejich obsahu. Zaměřuje se na techniky extrakce rysů, vzdálenostní metriky, návrh a implementaci modulu aplikace mediaDiff, který je schopen porovnávat videa na základě jejich obsahu.

Abstract

Development of Internet and its massive spread resulted in increased volume of multimedia data. The increase in the amount of multimedia data raises the need for efficient similarity detection between multimedia files for the purpose of preventing and detecting violations of copyright licenses or for detection of similar or duplicate files. This thesis discusses the current options in the field of the content-based image and video comparison and focuses on the feature extraction techniques, distance metrics, design and implementation of the mediaDiff application module for the content-based comparison of video files.

Klíčová slova

Diff, video, MPEG, extrakce rysů, vzdálenostní metriky, porovnávaní na základě obsahu, detekce hraníc záběrů, VideoDiff, mediaDiff

Keywords

Diff, video, MPEG, feature extraction, distance metrics, content-based comparison, shot boundary detection, VideoDiff, mediaDiff

Citace

Jozef Lang: Diff pro multimediální dokumenty, diplomová práce, Brno, FIT VUT v Brně, 2012

Diff pro multimediální dokumenty

Prohlášení

Prohlašuji, že jsem tuto diplomovou práci vypracoval samostatně pod vedením pana Ing. Petra Chmelaře.

.....
Jozef Lang
22. května 2012

Obsah

Kapitola 1

Úvod

S rozvojom internetu a jeho rozšírením medzi širokú verejnosť sa stal v posledných rokoch neoddeliteľnou súčasťou ľudského života. S nárastom počtu používateľov rastie aj objem multimediálnych dát prenášaných a dostupných na internete. Video sa v posledných rokoch stálo neoddetiteľnou súčasťou Internetu. Portály ako Youtube alebo Vimeo ponúkajú svojím používateľom možnosť zverejniť rôzne videá zdarma a bez poplatkov. Toto masové rozšírenie znižuje schopnosť jednotlivca vyhľadávať a triediť multimediálne informácie. Systémy ako Google Images sú schopné vyhľadávať podobné obrázky na internete, ale podobná funkcionalita určená pre video nie je k dispozícii.

Táto práca sa zaoberá problematikou získavania informácii o obsahu multimediálnych dát a to predovšetkým videa. Získané informácie sú následne využité pri zisťovaní rozdielov medzi videami. Cieľom práce je navrhnúť aplikáciu schopnú porovnať dve rôzne videá a určiť miesta v ktorých sa líšia. Tieto typy aplikácií sa súhrne anglicky nazývajú *diff* nástroje. Výsledná aplikácia má byť modulom aplikácie mediaDiff vyvýjanej na fakulte informačných technológií Vysokého Učení Technického v Brne v spolupráci s firmou Red Hat.

Práca je rozdelená do viacerích kapitol pojednávajúcich o jednotlivých aspektoch riešenej problematiky.

Kapitola 2 je venovaná multimédiam. Definuje pojem multimédium a pojednáva o digitálnej reprezentácii obrázkov a videa. Samostatná podkapitola je venovaná metadátam, ich účelu a veľký priestor je venovaný norme MPEG-7, ktorá definuje štandardizovaný popis audiovizuálnych dát.

Kapitola 3 pojednáva o problematike porovnávania súborov. V rámci tejto kapitoly sú načtnuté základné algoritmy pre zisťovanie rozdielnosti súborov. Prvá časť je venovaná textovým súborom a skupine nástrojov DiffUtils. Druhá časť je venovaná porovnávaniu súborov na základe obsahu, extrakcii rysov z obrazových súborov a videa. Posledná časť je venovaná algoritmom na určenie rozdielnosti a zároveň sú predstavené viaceré vzdialenostné metriky pre textové a obrazové dáta.

Kapitola 4 je venovaná predstaveniu aplikácie mediaDiff ktorej modul na porovnávanie videí na základe obsahu je v tejto práci navrhnutý a implementovaný. Priestor je venovaný histórii aplikácie a predstaveniu jednotlivých už existujúcich modulov.

Kapitola 5 sa venuje návrhu aplikácie a jej implementácií. Kapitola pojednáva o návrhových a implementačných detailoch aplikácie, predstavuje základnú implementovanú funkčnosť a stanovené návrhové a implementačné ciele.

Kapitola 6 sa venuje testovaniu výslednej aplikácie. Posledná kapitola je venovaná zhrnutiu obsahu práce a načrtáva možné smery pokračovania práce.

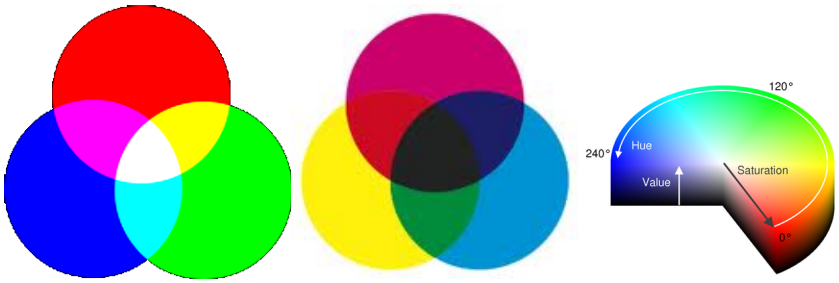
Kapitola 2

Multimédia

Podľa [1] sa pod pojmom multimédia rozumie kombinácia viacerích typov dát. Dáta môžu byť statické – v čase nemenné, alebo dynamické – v čase premenné. V oblasti informačných technológií je dôležitá digitálna reprezentácia preto v tejto kapitole sú popísané spôsoby digitalizácie obrázkov a videa a uvedené princípy získavania metadát.

2.1 Obrázky

Digitalizácia obrázkov funguje na princípe aproximácie analogového obrázku pomocou obrazových bodov takzvaných pixelov. Obrazové body sú usporiadané do mriežky a pre každy bod je nutné uchovávať informácie o farbe. V súčastnosti existuje viacero farebných modelov[26]. Medzi najpoužívanejšie patrí aditívny model RGB, skladajúci sa z troch zložiek – červenej, zelenej a modrej. Tieto tri farby sa nazývaju aj základné farby lebo ich kombináciou je možné popísať ostatné farby. Pridaním informácie o priehľadnosti, takzvaný alfa kanál k RGB modelu vznikne RGBA farebný model. Medzi subtraktívne farebné modely patrí model CMY(K), ktorý sa skladá zo štyroch zložiek – azúrovej, purpurovej, žltej a čiernej. Ďalším používaným modelom je model HSV skladajúci sa z troch zložiek – odtieňu, sýtosti a hodnoty farby.



(a) RGB (b) CMY(K) (c) HSV

Obrázek 2.1: Farebné modely. Prevzaté a upravené z [30, 3].

Celková velkosť obrázka sa dá vypočítať podľa vzťahu (2.1), kde S je celková velkosť obrázka, b je počet bajtov potrebných na uchovanie informácií o farbe, x a y predstavujú rozmery obrázka v pixeloch.

$$S = bxy \tag{2.1}$$

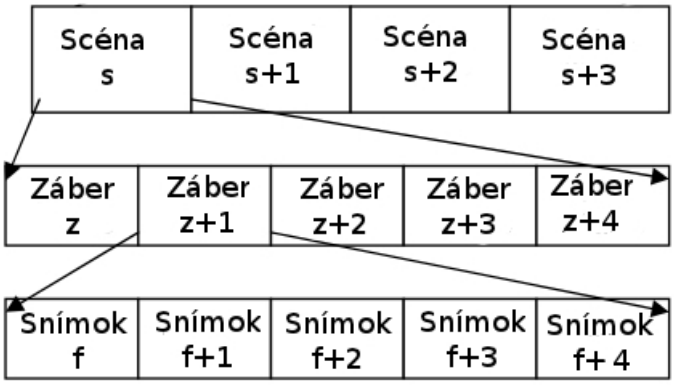
2.2 Video

Video podľa [34] je nahrávka ktorá vznikla nahrávacím zariadením zaznamenávajúcim pohyb. Video pozostáva zo snímkov. Medzi jednotlivými snímkami existuje určitá závislosť a video je možné rozdeliť do nasledujúcich hierarchických úrovní:

Snímok je braný samostatne, neexistuje žiadna časová závislosť.

Záber je spojitá sekvencia snímkov zosnímaných na jedno neprerušené natáčanie kamerov. Existuje časová závislosť.

Scéna je spojitá sekvencia záberov majúca rovnakú sémanticku významnosť. Časová závislosť vo videu má hierarchickú štruktúru. Grafické znázornenie hierarchickej štruktúry je na obrázku 2.2.



Obrázek 2.2: Hierarchické usporiadanie videa. Obrázok prevzatý a upravený z [5].

Ako bolo už spomenuté video scéna je tvorená zábermi. Jednotlivé zábery nasledujú postupne za sebou a zmena jedného záberu za iný je možné podľa [5] uskutočniť buď strihom, rozplynutím alebo takzvaným rozotretím. Strih je ostrý prechod medzi zábermi. Prejavuje sa náhlou zmenou rysou hraničných snímkov. Rozplynutie je prechod pri ktorom sa obsah posledných snímkov povodného záberu zmieša s obsahom prvých snímkov nasledujúceho záberu. Rozotretie je nahradnie povodného záberu tvarovaným prechodom. Prechod medzi zábermi môže byť uskutočnený viacerými spôsobmi.

Existujú situácie kde je potrebné jednotlivé zábery spojiť. Medzi techniky spájania patrí jednoduché spájanie pri ktorom sa dva zábery spoja bez akýchkoľvek prechodových funkcií, priestorová kompozícia pri ktorej dochádza k postupnému prekrytiu pôvodného záberu novým a chromatická kompozícia pri ktorej dochádza k postupnému nahradzovaniu pixelov prvého záberu pixelami druhého záberu.

Vo videu taktiež dochádza k časovým a priestorovým redundanciám. Priestorová redundancia vzniká z dôvodu podobnosti určitých častí jedného snímku. Časová redundancia vzníká kvôli časovými závislostiam a dochádza k podobnosti po sebe nasledujúcich snímkov.

2.2.1 MPEG

MPEG – Moving Picture Coding Expert Group je názov skupiny vytvorenej medzinárodnou organizáciu pre štandardy. Účel MPEG je vytvoriť štandard pre popis a kódovanie

audiovizuálnych dát. Účel vzniku MPEG bolo štandardizovanie kódovanie a kompresia audiovizuálnych dát na digitálnych médiach [12]. Prvé stretnutie MPEG sa konalo v roku 1988 a zúčastnilo sa ho 25 expertov z rôznych svetových spoločností. V súčastnosti má skupina viac ako 300 členov. Počas svojho pôsobenie skupina štandardizovala viacero štandardov. Prvým bol štandard MPEG-1 prijatý v roku 1992. Štandard pozostával zo štyroch častí. Návrh štandardu bol cielený na šírku pásma CD-ROM-ov a Video-CD, ale používa sa napríklad aj pri technológiach VOD – Video on Demand alebo prenosu videa cez internet [14].

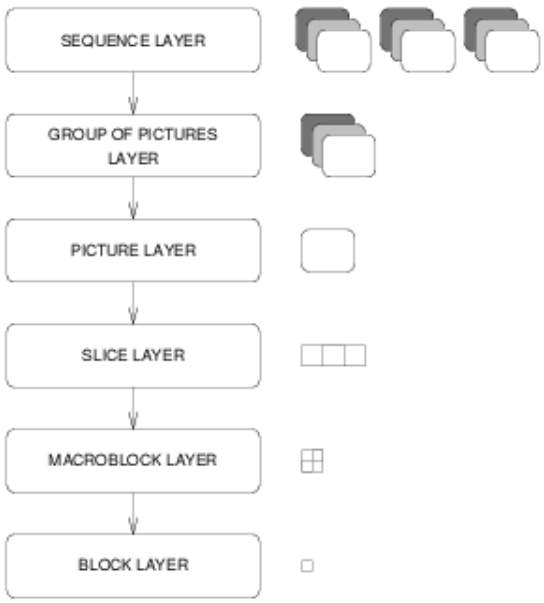
V roku 1992 bol prijatý štandard MPEG-2. Účelom štandardu bolo priniesť lepšiu kvalitu obrazu. Tento štandard sa stal štandardom pre kompresiu audiovizuálnych dát na DVD [14].

Práca na štandarde MPEG-3 začala v roku 1992 a cieľom bolo vyvinúť štandard pre HDTV technológiu. Avšak po vytvorení prvého draftu štandardu MPEG-2 sa rozhodlo, že štandard MPEG-3 nie je potrebný a požiadavky stanovené na tento štandard je možné pokryť aj štandardom MPEG-2. Práce na štandarde MPEG-3 boli preto ukončené [12].

Ďalším prijatým štandardom je MPEG-4. Práce na tomto štandarde začali v roku 1995 a štandard bol prijatý v roku 1998. Štandard vychadza zo štandardov MPEG-1 a MPEG-2 z ktorých prebral mnoho vlastností. Tento štandard je určený pre interaktívne prostredie a použitie v prostredí internetu [12].

Štandard MPEG-7 je štandardom pre popis audiovizuálnych dát. Tomuto štandardu je venovaná samostatná podkapitola 2.3.1.

Video kódované MPEG je rozdelené do viacerích vrstiev. Toto rozdelenie do viacerích vrstiev slúži k lepšej obsluhe chýb a synchronizácií so zvukovou stopou. Prvá vrstva obsahuje jednotlivé snímky videa. Druhá vrstva obsahuje informácie o skupinách snímkov – Group of Pictures. Trietia vrstva obsahuje informácie o jedntolivých zakódovaných snímkoch ich typoch. Nižšie vrstvy obsahujú informácie o konkrétnych rastroch. Poradie jednotlivých vrstiev je zobrazené na obrázku 2.3 [9].



Obrázek 2.3: Typy vrstiev v MPEG. Obrázok prevzatý z [9].

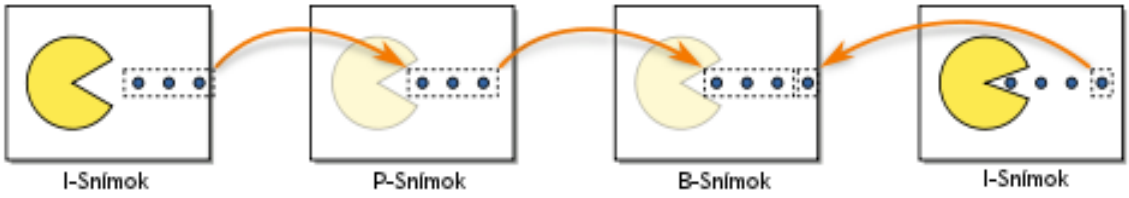
Ako už bolo načrtnuté, štandard MPEG definuje tri typy snímkov. Na obrázku 2.4 sú

znázornené jednotlivé typy snímkov podľa [32]:

I-snímky takzvané intra-snímky. Tieto snímky sú kódované samostatne bez väzby na okolité snímky. Kompresia spočíva v odstraňovaní priestorových redundancií. Časové redundancie nemožno odstrániť nakoľko sa snímok kóduje samostatne.

P-snímky takzvané predictive-snímky alebo inter-snímky. Tieto snímky dosahujú lepšiu mieru kompresie nakoľko je možné komprimovať ako aj priestorové tak aj časové redundancie. P-snímky môžú využívať informácie s predchádzajúcich I-snímkov alebo P-snímkov ku predikcii pohybu.

B-snímky takzvané bidirectionally-predictive alebo taktiež inter-snímky. Pri odstraňovaní časovej redundancie a predikcii pohybu môžu využiť predchádzajúce snímky, podobne ako P-snímky, ale navyše môžu využiť informácie z nadchádzajúcich snímkov. B-snínmky dosahujú najväčšiu mieru kompresie.



Obrázek 2.4: Typy snímkov v MPEG. Obrázok prevzatý a upravený z [21].

Jednotlivé po sebe nasledujúce snímky je možné zlučovať do skupín, takzvaných Group of Pictures. Snímky si v rámci GoP zachovávajú časovú postupnosť. Priemerná dĺžka GoP je približne 15 snímkov a je ju možné popísať dvoma veličinami:

- Vzdialenosťou medzi dvoma nasledovnými I-snímkami
- Vzdialenosťou medzi dvoma nasledovnými P-snímkami

Pri použití kódovania pomocou MPEG dochádza k takzvanej stratovej kompresii. Niektoré informácie sú z videa odstránené čo má za následok menšiu velkosť výsledného videosúboru. Tieto odstránené informácie sú avšak pri prehrávaní videa znova vložené čím vzniká len malý rozdiel oproti originálnemu videu. Kódovanie MPEG okrem už spomenutej časovej a priestorovej redundancie oodstraňuje aj psychovizuálnu redundanciu [11]. Odstraňovanie psychovizuálnej redundancie spočíva v odstraňovaní častí videosúboru ktoré su pre ľudské oko nepostrehnuteľné. V rámci MPEG je pre dosiahnutie kompresie výsledného videosúboru používaná diskrétna kosínusová transformácia – DCT a takzvané pohybové (motion) vektory. DCT transformácia je použitá pri I-snímkoch a pohybový vektor je použitý pri kompresii P a B snímkov.

Všeobecná DCT transformácia, rovnica (2.2), je aplikovaná na bloky o velkosti 8x8 pixelov. Vstupom je blok o velkosti MxN kde funkcia $f(x, y)$ predstavuje intenzitu obrazového bodu na pozícii x, y . Výstupom je matica DCT koeficientov. Získané koeficienty sú kvantované a je aplikovaný zig-zag skenovací vzor ktorého výstupom je zoznam kvantifikovaných koeficientov zoradených podľa skenovacieho vzoru. Následne je na dané koeficienty aplikovaná VLC – Variable-length code kompresia [11].

$$f(x,y) = \left(\frac{2}{N}\right)^{\frac{1}{2}} \left(\frac{2}{M}\right)^{\frac{1}{2}} \sum_{i=0}^{N-1} \sum_{j=0}^{M-1} \Lambda(i).\Lambda(j). \cos\left[\frac{\pi x}{2N}(2i+1)\right] \cos\left[\frac{\pi y}{2M}(2j+1)\right] f(i,j) \tag{2.2}$$

kde

$$\Lambda(i) = \begin{cases} \frac{1}{\sqrt{2}} & : \quad x,y = 0 \\ 1 & : \quad inak \end{cases} \tag{2.3}$$

Pohybové vektory v MPEG slúžia na určenie posunu objektov v jednotlivých snímkoch odstraňujúc časovú redundanciu. Snímok je rozdelený do makroblokov o veľkosti $16x16$ pixelov a je určený referenčný blok vo východzom snímku a pohybový vektor. Tieto získané informácie sú následne zakódované [11].

2.3 Metadata

Neštruktúrované dáta ako sú zvuk, obrázky a video obsahuje veľké množstvo informácií a preto je v mnohých prípadoch vhodné nepracovať priamo s týmito neštrukturovanými dátami, ale je lepšie získať s týchto neštrukturovaných dát popisné dáta. Tieto popisné dáta sa nazývajú metadata. Podľa [1] metadata by mali spĺňať nasledujúce podmienky:

- popisné informácie by mali byť čo najviac ucelené.
- nemali by byť objemné vzhľadom na objem pôvodných dát.
- mali by byť rýchlo porovnateľné.

Existujú rôzne typy metadát. Dajú sa rozdeliť do týchto skupín:

Popisy Popisy sú druh metadát, ktoré poskytujú popisné informácie o multimediálnych dátach ako celku.

Anotácie Anotácia je textový popis obsahu multimediálnych dát.

Rysy Rys je určitá charakteristika multimediálnych dát. Proces pri ktorom sa získavajú informácie o rysoch multímediálnych dát sa nazýva *extrakcia rysov*, anglicky *feature extraction*. Rysy sa dajú rozdeliť do dvoch hlavných skupín:

Nízkoúrovňové rysy zahŕňajú rysy úzko späté s konkrétnym typom multimediálnych dát. Pri obrázkok sem patria rysy spojené s farbou, rôzne farebné histogrami a priestorové závislosti. Tieto rysy je možné získavať automatizovane.

Vysokoúrovňové rysy sú rysy ktoré majú význam pre koncového používateľa [1]. Nízkoúrovňové rysy narozdiel od vysokoúrovňových rysov neposkytujú používateľovi žiadnu konkrétnu informáciu o tom, čo sa na danom multimédiu nachádza. Negatívom vysokoúrovňových rysov je, že je často nutná spolupráca s používateľom.

Pre popis metadát multimediálnych dát vznikol štandard MPEG-7 vyvinutý skupinou MPEG. Cieľom štandardizácie bolo štandardizovať popis metadát audiovizuálnycd dát. Skupina MPEG vyvinula viacero štandardov, niektoré z nich boli už spomenuté v podkapitole 2.2.1. Vzhľadom na povahu a ciele tejto práce bude v nasledujúcej kapitole predstavená len jedna časť zo štandardu MPEG-7 a to časť Visual, ktorá sa zaoberá popisom vizuálnych rysov.

2.3.1 MPEG-7

Vývoj na štandarde MPEG-7 [27, 17] začal v roku 1996 pod záštitou MPEG. Cieľom bolo vytvoriť štandardizovaný spôsob popisu audiovizuálnych dát v multimediálnom prostredí. Na uchovávanie metadát je použitý formát XML. Štandard bol navrhnutý tak, aby bol nezávislý na použitom kódovaní popisovaných dát. Hlavným cieľom je štandardizácia:

Popisovačov ktoré definujú syntax a sémantiku každého rysu.

Popisovacích schém ktoré definujú štruktúru a sémantiku vzťahov medzi komponentami, ktorými môžu byť popisovače, alebo popisovacie schémy.

Popisovacieho jazyka ktorým sa definuje jazyk MPEG-7 štandardu za účelom možnosti vytvárania nových popisovačov a popisovacích schém, alebo rozšírenia už existujúcich.

Ako už bolo spomenuté MPEG-7 sa skladá zo 7 častí. Kompletný výpis jednotlivý častí je dispozícii v [27]. Táto podkapitola sa venuje popisovačom definovaných v časti MPEG-7 Visual konkrétne popisovače farieb, textúr a útvarov.

Popisovače farby

Popisovače farby ako už názov napovedá popisujú farbu. MPEG- ponúka 7 druhov popisovačov farby. Popisovač *Color space* slúži na popísanie farebného modelu použitého v jednotlivých popisovačoch farby. V rámci MPEG-7 je definovaných 6 druhov farebných modelov a to konkrétne RGB, YCbCr, HSV, HMMD, monochromatický farebný model a model lineárnej transformačnej matice s referenciou na RGB. Popisovač *Color quantization* popisuje spôsoby akými kvantovať spojité hodnoty farebných modelov do diskrétnych hodnôt – košov. Počet košov do ktorých kvantovať je možné konfigurovať. Informácie z tejto podkapitoly boli čerpané z [27, 17, 18]

Popisovač *Dominant color* slúži na popísanie dominantných farieb v učitom výseku alebo celom snímku. Je ho možné aplikovať v aplikáciach zameraných na získavanie a porovnávanie snímkov na základe obahu. *Dominant color* je definovaný vzťahom (2.4). Hodnota N vyjadruje počet dominantných farieb. Hodnota c_i predstavuje dominantnú farbu reprezentovanú vo zvolenom farebnom modely. Hodnota p_i predstavuje percentuálne zastúpenie dominantnej farby v danom úseku alebo snímku. Hodnoty v_i a s popisujú variabilitu farieb v okolí dominantnej farby a priestorovú koherenciu. Počet získaných dominantných farieb môže byť rôzny.

$$F = \{\{c_i, p_i, v_i\}, s\}, i = (1, 2, \ldots, N) \tag{2.4}$$

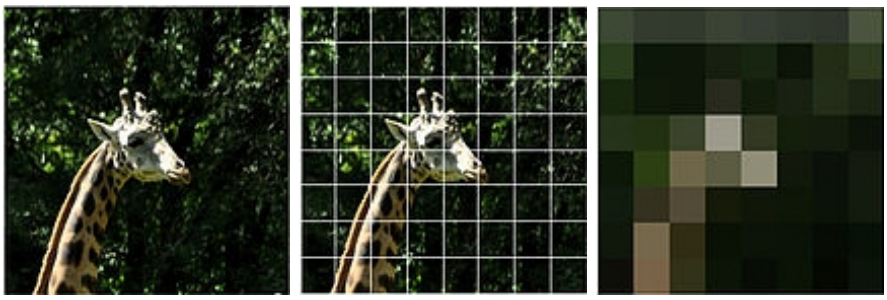
Popisovač *Scalable color* je aplikáciou Haarovej transformácie na hodnoty farebného histogramu v HSV farebnom modely. Jednotlivé hodnoty HSV farebného histogramu sú získané zo snímku a následne normalizované a nelineárne rozdelené do štyroch diskrétnych hodnôt. Na diskrétne hodnoty je následne aplikovaná Haarova transformácia

Group of Pictures je popisovač popisujúci skupinu snímkov združených do jedného GoP. Hodnota popisovača je vypočítaná z histogramov jednotlivých snímkov. Histogramy pre jednotlivé snímky sú vyjadrené v HSV farebnom modely. Konečný výpočet hodnoty popisovača môže byť založený na nájdení minima, priemeru alebo sumy jednotlivých združených snímkov.

Ďalším popisovačom definovaným v MPEG-7 Visual je *Color structure*. Tento popisovač popisuje zastúpenie jednotlivých farieb a ich priestorovú štruktúru. Popis priestorovej štruktúry umožnuje lepšie popísať obsah ktorý sa z pohľadu zastúpenia farieb javí ako rovnaký.

Počet hodnôt ktorými bude obsah snímku popísaný je možné upravovať. Pri získavaní tohto popisovača je najprv získaný histogram kvantovaný do 256 diskrétnych hodnôt v HMMD farebnom modele. Pokiaľ je počet hodnôt ktorými popísať snímok menši ako 256 sú získané hodnoty prepočítané tak aby bolo získaných N diskrétnych hodnôt.

Popisovač *Color layout* je veľmi rýchly popisovač vhodný na použitie v aplikáciach zaoberajúcich sa podobnostným vyhľadávaním. Je invariantný voči rozmerom spracovávaného snímku a navrhnutý na efektívne popísanie rozloženia farieb. Proces získavania tohto popisovača začína rozdelením spracovávaného snímku na 64 blokov. Následne je v každom bloku určená dominantná farba. Na určenie dominantnej farby je možné použiť rôzne algoritmy na určenie priemeru, medianu alebo iné. Následne je z jednotlivých reprezentatívnych farieb vytvorený obrázok o veľkosti 8x8 obrazových bodov. Z vytvoreného obrázka je pomocou diskrétnej kosínusovej transformácie získaných množina 64 DCT koeficientov. Tieto koeficienty sú pomocou zig-zag skenera zoradené a kvantované do zvoleného počtu košov. Reprezentácia získanych údajov je vo forme 3 vektorov – 1 vektor pre každú zložku farebného modelu so zvoleným počtom hodnôt.



(a) (b) (c)

Obrázek 2.5: Rozklad obrázka (a) pôvodný snímok, (b) rozdelenie do blokov, (c) obrázok z dominantných farieb jednotlivých blokov. Prevzaté z [22].

Popisovače textúr

Textúra je vlastnosť výseku obrázka tvorná priestorovým rozložením rôznych geometrických primitív. Popisovače textúr poskytuju informácie na prechádzanie, vyhľadávanie na základe texturných vlastností. Popisovač *Homogeneous texture* chápe obrázok ako množinu textúr na základe ktorých môže byť obrázok indexovaný. Popisovač reprezentuje textúru pomocou 62 čísel. Hodnoty sa získavajú aplikovaním Gáborových filtrov. *Homogeneous texture* je vhodný pre popisovanie homogénnych štruktúr. Zameriava sa na vlastnosti blízke ľudskému vnímaniu ako sú napríklad pravideľnosť a orientácia. Výpočet tohto popisovača začína vyfiltrovaním obrázka pomocou Gáborových filtrov a následne sú identifikované dve dominantné orientovania a pozdĺž identifikovaných orientovaní je zisťovaná pravidelnosť textúry. Popisovač *Edge histogram* popisuje priestorové rozloženie jednotlivých typov hrán. Nakoľko popisom hrán je možné identifikovať snímky s rovnakým sémantickym obsahom je tento popisovač cielený na použitie v aplikáciach na porovnávanie na základe obsahu. Informácie boli čerpané z [27, 17].

Popisovače útvarov

Popisovače útvarov popisujú priestorové rozloženie obrazových bodov patriacich do rôznych útvarov. Tieto popisovače sa dajú rozdeliť do dvoch hlavných skupín. Na dvojdimenzionálne a trojdimenzionálne. Medzi dvojdimenzionálne patrí *Region Shape* ktorý popisuje rozloženie obrazových bodov v rámci určitého útvaru. Pod pojmom útvar rozumie zoskúpenie obrazových bodov a to či už jednoliate zložené z jedného homogenného celku alebo z v rámci útvaru sa môžu vykytovať aj miesta ktoré do útvaru nepatria – diery. Ďalším popisovačom je *Contour Shape* a zachytáva viditeľné charakteristiky útvarov. Medzi 3D popisovače útvarov patrí *Shape 3D* ktorý využíva 3D mriežku. Informácie boli čerpané z [27, 17].

Kapitola 3

Techniky porovnávania súborov

S nárastom rozšírenia výpočtovej techniky rastie aj množstvo dát, ktoré sú digitálne prenášané a uchovávané. S narastajúcim množstvom rastie taktiež aj náročnosť zisťovania rozdielov a zmien medzi jednotlivými súbormi. Nástroje, ktoré rieša problematiku zisťovania zmien medzi súbormi sa súhrne nazývaju ako nástroje na porovnávanie obsahu. Táto kapitola pojednáva o problematike zisťovania zmien v textových súboroch, obrázkoch a videu.

3.1 Porovnávanie textových súborov

V tejto podkapitole je predstavených niekoľko všeobecných algoritmov na porovnávanie dát. Nakoľko tieto algoritmy boli pôvodne navrhnuté na porovnávanie textových súborov a pracujúc nad množinou riadkov alebo znakov avšak táto skutočnosť nezabraňuje použitiu uvedených algoritmov aj na porovnávanie iných typov dát. V nasledujúcom texte budú preto ukážky ilustrované na textových dátach.

Problém porovnávania akýchkoľvek sa dá aproximovať na problém hľadania najdlhšieho spoločného podreťazca, anglicky Longest Common Subsequence [19]. Podľa [8] problém *najdlhšieho spoločného reťazca* je definovaný nasledovne.

Reťazec $C = c_1c_2 \dots c_p$ je podreťazcom reťazca $A = a_1a_2 \dots a_m$ ak existuje funkcia $F : \{1, 2 \dots p\} \to \{1, 2 \dots m\}$ taká, že $LCS(i) = k \longleftrightarrow c_i = a_k$ a zároveň LCS je monotónna a striktne rastúca a reťazec C vznikne odobraním $m - p$ znakov z reťazca A.

Reťazec $C = c_1c_2 \dots c_p$ je podreťazcom reťazca $A = a_1a_2 \dots a_m$ ak existuje funkcia $F : \{1, 2 \dots p\} \to \{1, 2 \dots m\}$ taká, že $LCS(i) = k \longleftrightarrow c_i = a_k$ a zároveň LCS je monotónna a striktne rastúca a reťazec C vznikne odobraním $m - p$ znakov z reťazca A.

Rekurzívny matematický zápis funkcie LCS podľa [19] je zapísaný rovnicou (3.1).

$$LCS(C_i, A_k) = \begin{cases} \emptyset & \text{ak} \quad i = 0 \vee k = 0 \\ (LCS(C_{i-1}, A_{k-1}), C_i) + 1 & \text{ak} \quad C[i] = A[k] \\ max(LCS(C_{i-1}, A_k), LCS(C_i, A_{k-1})) & \text{ak} \quad C[i] \neq A[k] \end{cases} \tag{3.1}$$

Priebeh výpočtu LCS algoritmu sa dá zobraziť v tabuľke. V tabuľke 3.1 je zobrazený priebeh LCS algoritmu pre reťazce *XMJYAUZ* a *MZJAWXU*.

		0	1 M	2 Z	3 J	4 A	5 W	6 X	7 U
0		0	0	0	0	0	0	0	0
1	X	0	0	0	0	0	0	1	1
2	M	0	1	1	1	1	1	1	1
3	J	0	1	1	2	2	2	2	2
4	Y	0	1	1	2	2	2	2	2
5	A	0	1	1	2	3	3	3	3
6	U	0	1	1	2	3	3	3	4
7	Z	0	1	2	2	3	3	3	4

Tabulka 3.1: Priebeh algoritmu LCS

3.1.1 Diff

Algoritmus diff slúži na nájdenie zmien medzi dvoma textovými súbormi. Optimálnym výstupom je minimálny zoznam zmien ktoré je nutné vykonať aby porovnávané súbory boli zhodné. Jadro algoritmu diff rieši problém najdlhšej spoločnej postupnosti a hľadá minimálny editovací skript. Na riešenie najdlhšej spoločnej postupnosti nie je k dispozícií všeobecne optímalny algoritmus. Najjednoduchším spôsobom je prechádzať jednotlivé súbory riadok po riadku a hľadať zmenu. Existuje avšak jednoduchý algoritmus na báze dynamického programovania na riešenie problému najdlhšej spoločnej postupnosti. Majme dva súbory s riadkami $A = 0 \ldots m$ a $B = 0 \ldots n$. Algoritmus porovná prvých i riadkov z prvého súboru a prvých j riadkov a P_{ij} je najdlhšia spoločná postupnosť prvých i riadkov z prvého súboru a prvých j riadkov z druhého súboru. Zo spomenuté algoritmu vyplýva a platí:

$$P_{j0} = 0 \quad i = 0, \ldots, m$$
$$P_{0j} = 0 \quad j = 0, \ldots, n$$

$$P_{ij} = \begin{cases} 1 + P_{i-1,j-1} & A_i = B_j \\ \max(P_{i-1,j}, P_{i,j-1}) & A_i \neq B_j \end{cases} \quad 1 \leq i \leq m, 1 \leq j \leq n \tag{3.2}$$

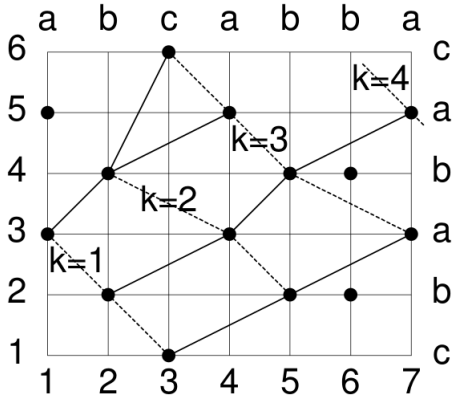
P_{mn} je najdlhšia spoločná postupnosť porovnávaných reťazcov navyše z jednotlivých čiastkových hodnôt z ktorých bola hodnota P_{mn} vypočítaná je jednoduché zistiť jednotlivé čiastkové hodnoty najdlhšej spoločnej postupnosti. Časová zložitosť ako aj priestorová zložitosť tohto algoritmu je v najhoršom prípade $O(mn)$ [16]. Hirschberg neskôr vylepšil tento algoritmus a dosiahol lineárnu časovú zložitosť.

Hunt a McIlroy v práci [16] predstavili vylepšenie pôvodného algoritmu a to také, že sa do úvahy berú len podstatné zhody ktorých porušenie by zmenilo výsledok výpočtu najdlhšej spoločnej postupnosti. Hirschberg definoval podstatné zhody ako k-kandidátov ktorí nastatnú keď $A_i = B_j$ and $P_{ij} > \max(P_{i-1,j}, P_{i,j-1})$. K-kandidát je dvojica indexov (i, j) takých že platí $A_i = B_j$ a zároveň najdlhšia spoločná postupnosť dlžky k leží medzi prvými i prvkami prvého súboru a prvými j prvkami druhého súboru a zároveň neexistuje najdlhšia spoločná postupnosť dĺžky k pokiaľ je i alebo j vynechané. Je zjavné, že najdlhšia spoločná postupnosť leží v kompletnom zozname k-kandidátov.

Ak (i_1, j_1) a (i_2, j_2) a $i_i < i_2$ a obe páry sú k-kandidáti potom $(j_1 > j_2)$. Pokiaľ $j_1 = j_2, (i_2, j_2)$ a porušovalo by to podmienku o neexistencii najdlhšej spoločnej postupnosti o dĺžke k pokiaľ i alebo j vynechané a pokiaľ $j_1 < j_2$ potom najdlhšia spoločná postupnosť

dĺžky k končiaca (i_1, j_1) môže byť rozšírená na najdlhšiu spoločnú postupnosť dĺžky $k+1$ končiac (i_2, j_2).

Metóda $k$kandidátov sa dá jednoducho znázorniť. Na obrázku 3.1 body označujú dvojice (i, j) pre ktoré platí, že $A_i = B_j$. Spoločné postupnosti sú množina bodov ktoré sú pospájané striktne rastúcou krivkou. Na obrázku sú znázornené štyri také krivky ktoré spájajú body ktoré sú kandidátmi. Hodnoty jednotlivých kandidátov sú zobrazené pri každej spoločnej postupnosti. Bodkovanou čiarou sú pospájané postupnosti rovnakej dĺžky. Tieto postupnosti rovnakej dĺžky musia monotónne klesať. Kandidátov je zreteľne menej ako mn a v reaálnych dátach ich je ešte menej [16].



Obrázek 3.1: k-kandidáti a spoločné postupnosti. Prevzaté z [16].

Body v uvedenej ukážke sú uložené v lineárnom priestore nasledovne:

1. Zostroja sa zoznamy tried ekvivalencie jednotlivých prvkov v druhom súbore. Tieto zoznamy budú zaberať $O(n)$ priestoru.

2. Ku každému prvku z prvého súboru sa priradí trieda ekvivalencie. Informácie o priradniach budú zaberať $O(m)$ priestoru.

Generovanie k-kandidátov prebieha zľava do prava. Nech K je pole určujúce najpravejších k-kandidátov pre každé k. Pre zjednodušenie výpočtu sa pridáme formálneho 0-kandidáta $(0, 0)$. Pre každé k ktoré nemajú kandidátov pridáme formálneho kandidáta na pozíciu $(m+1, n+1)$. K-pole je na začiatku prázdne až na výplň a je aktualizované pri pohybe doprava. Ostatné časti výpočtu minimálneho editačného skriptu a najdlhšej spoločnej postupnosti zostávajú nezmenené [16].

Pre zrýchlenie výpočtu pri veľkých vstupných súborov diff hašuje každý prvok súboru do dlžky jedného slova. Táto operácia môže spôsobiť, že prvky ktoré nie sú rôzne budú vyhodnotené ako zhodné. Pokiaľ je použitá vhodná hašovacia funkcia pravdepodobnosť že nastane chybné porovnanie je $1/M$ kde $1 \ldots M$ sú hodnoty ktoré môže hašovacia funkcia nadobúdať [16].

Zložitosť algoritmu v najhoršom možnom prípade nie je oveľa lepšia ako klasický prístup dynamického programovania. Najhoršia časová zložitosť je $O(mn \log m)$ avšak v praxi táto situácia nastane len málokedy [16].

3.1.2 Heckelov algoritmus

Paul Heckel v [7] predstavil algoritmus na určovanie zmien v súboroch. Podľa jeho slov algoritmus pracuje so zažitou predstavou o zmenách, je ľahko implementovateľný a výpočtovo

efektívny s lineárnou zložitosťou. Heckelov algoritmus je založený na dvoch predpokladoch:

1. Riadok ktorý sa nachádza raz a len raz v každom súbore musí byť ten istý riadok, nezmenený ale možno premiestnený. Takéto riadky sú nájdené a vylúčené z ďalších výpočtov.

2. Riadky ktoré sú priľahlé už nájdeným riadkom v bode 1 musia byť takisto nezmenené a teda opakovanou aplikáciou tohto pravidla sú nájdené sekvencie nezmenených riadkov.

Heckelov algoritmus pracuje nad dvoma súbormi nazvanými ako starý, ktorý je označený písmenom O a novým označeným písmenom N. Algoritmus vychádza z predpokladu, že súbor O bol upravený tak, aby vznikol súbor N. Algoritmus pracuje nad troma dátovými štruktúrami – tabuľkou symbolov a dvoma vektormi označenými ako OA a NA. Jednotlivé riadky slúžia ako kľúče v tabuľke symbolov. Každý záznam v tabuľke symbolov dve počítadla, OC a NC ktoré určujú počet kópií daného riadku v súboroch O respektíve N. Tieto počítadla nadobúdaju len hodnoty 0,1 a hodnotu veľa. Tabuľka symbolov taktiež obsahuje hodnotu OLNO ktorá uchováva číslo riadku v starom súbore. Táto hodnota je zaujímava iba ak hodnota OC je rovná 1 [7].

Vektory OA respektíve NA obsahujú jeden záznam pre každý riadok v danom súbore. Riadok môže byť identifikovaný smerníkom do tabuľky symbolov alebo pomocou čísla riadku na ktorom sa v danom súbore nachádza [7].

Samotný algoritmus pozostáva zo šiestich krokov. V prvom kroku sa inicializujú dátove šturktúry. Súbor N je sekvenčne prečítaný riadok po riadku a pre každý načítaný riadok je vytvorený záznam v tabuľke symbolov. Pokiaľ je to možné tak je inkrementovaná hodnota počítadla NC a hodnota vektora NA$[i]$ je nastavená tak, aby ukazovala na záznam pre riadok i v tabuľke symbolov [7].

Druhý krok je totožný s krokom jedna s tým rozdielom, že sa spracuje súbor O. V treťom kroku sa aplikuje predpoklad číslo 1 a spracujú sa len riadky pre ktoré platí NC $=$ OC $= 1$ nakoľko je predpoklad že predstavujú nezmenené riadky. Pre každý takýto riadok sú smerníky vo vektoroch OA a NA nastavené na číslo daného riadku v druhom súbore. Napríklad ak je riadok NA$[i]$ tak je vyhľadaný v tabuľke symbolov a následne je hodnota NA$[i]$ nastavená na hodnotu OLNO a OA[OLNO] na hodnotu i [7].

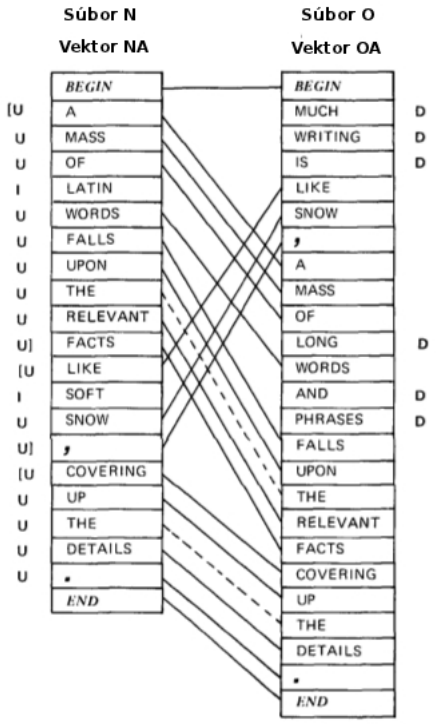
Vo štvrtom kroku je aplikovaný predpoklad čislo 2. Vektor NA sa spracuje vo vzostupnom poradí a ak NA$[i]$ ukazuje na OA$[j]$ a zároveň NA$[i+1]$ a OA$[j+1]$ ukazujú na rovnaký záznam v tabuľke symbolov potom OA$[j+1]$ je nastavená na riadok $i+1$ a NA$[i+1]$ je nastavená na riadok $j+1$ [7].

V piatom kroku sa taktiež aplikuje predpoklad číslo 2 a vektor NA sa spracuje v zostupnom poradí a ak NA$[i]$ ukazuje na OA$[j]$ a zároveň NA$[i-1]$ a OA$[j-1]$ ukazujú na rovnaký záznam v tabuľke symbolov potom OA$[j-1]$ je nastavená na riadok $i-1$ a NA$[i-1]$ je nastavená na riadok $j-1$ [7].

V tomto momente vektor NA obsahuje informácie potrebné k vygenerovaniu editačného skriptu. Pokiaľ NA$[i]$ ukazuje do tabuľky symbolov potom možme predpokladať, že ide o vloženie nového riadku i, respektíve ak NA$[i]$ ukazuje na OA$[j]$ ale NA$[i+1]$ neukazuje na OA$[j+i]$ môžme predpokladať že riadok i je hraničný riadok zmazaného alebo presunutého bloku [7]. V šiestom kroku sú jednotlivé zmeny poslané na výstup v príslušnej forme podľa aplikácie.

Na obrázku 3.2 je zobrazený stav vektorov NA o OA po piatom kroku. Plnými čiarami sú spojené všetky unikátne a rovnaké riadky ktoré sú identifikované v treťom kroku. Vo štvrtom

a piatom kroku sú identifikované identické ale nie unikatne riadky. Na obrázku označené prerušovanou čiarou. Jednotlivé zmenené riadky – vložené (I), zmazané (D), nezmenené (U) alebo presunuté ([,]) sa dajú identifikovať a vypísať používateľovi.



Obrázek 3.2: Výsledný stav vektorov NA a OA. Prevzaté a upravené z [7].

3.1.3 Dynamické ohýbanie času

Dynamické ohýbanie času (dynamic time warping - DTW) je známa technika porovnávania používaná na nájdenie optímalného zarovnania dvoch sekvencií ktoré sa môžu líšiť v čase a rýchlost. Dynamické ohýbanie času bolo pôvodne vytvorené pre potreby rozpoznávania hlasu ale dá sa úspešne aplikovať aj na iné problémy. Príkladom typického použitia je porovnanie dvoch úsekov reči.

3.1.4 Existujúce nástroje

GNU DiffUtils [15] vydaný pod licenciou GNU a dostupný na Unix systémoch obsahuje nástroje na porovnávanie obsahu textových súborov. V tejto podkapitole sú popísané jednotlivé nástroje ktoré združuje.

Diff

Nástroj *diff* textový súbor chápe ako sériu riadkov textu, kde zmena je chápaná ako vloženie alebo odstránenie riadku, respektíve zmena jeho obsahu. Postupne číta vstupné súbory riadok po riadku a pokiaľ narazí na zmenu reportuje nájdenú zmenu vo zvolenom formáte, ktoré je možné meniť podľa účelu. Zároveň umožnuje ignorovať určité zmeny ako napríklad počet bielych znakov. Základná syntax spustenia programu *diff* je na ukážke 3.1.

Najjednoduchším spôsobom spustenia je nad dvoma súbormi. V takom prípade porovná

Ukázka 3.1: Spustenie programu *diff*

```
diff [prepináče]... súbory
```

obsah oboch súborov. Pokiaľ sa ako vstupný súbor použije znak -, porovná obsah druhého súboru s obsahom štandardného vstupu. Spustenie programu s dvoma parametrami -, spôsobí, že sa medzi sebou porovnajú dve kópie obsahu štandardného vstupu.

Ak je program *diff* spustený nad súbormi, ktoré sú adresármi potom sa porovnajú súbory obsiahnuté v danych adresároch. Súbory sa porovnávajú v abecednom poradí. Pokiaľ sa pri spustení nad adresármi použitý prepínač *–recursive* alebo *-r* sú rekurzívne porovnávané aj súbory nachádzajúce sa v podadresároch. Ukážka 3.2 zobrazuje výstup programu *diff*. Program bol spustený v základnom nastavení.

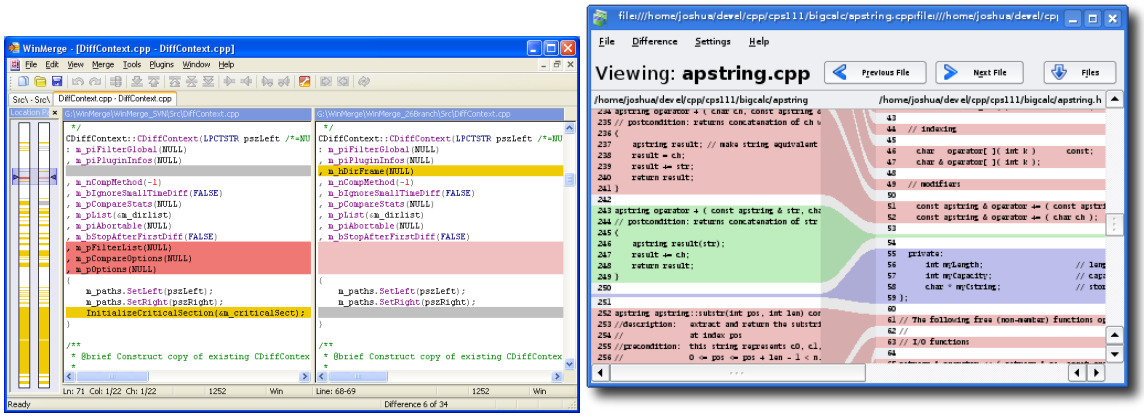
Ukázka 3.2: Výstup programu *diff*

```
1c1
< Lorem ipsum dolor sit amet
---
> Lorem ipsum dolor sit Amet
```

Ďalšie implementácie

V súčastnosti existuje mnoho implementácii *diff* dostupných pre rôzne platformy. Program *WinMerge* [28] je implementácia *diff* dostupná na platforme Windows a vydávaná pod licenciou GPL. *WinMerge* obsahuje funkcionalitu porovnávania súborov a adresárov a vizualizáciu rozdielov viď obrázok 3.3a.

Príkladom implementácie *diff* na platforme Linux a užívateľským rozhraním KDE je program *Kompare* [25], ktorá obsahuje funkcionalitu porovnávania dvoch súborov či adresárov viď obrázok 3.3b.



(a) WinMerge (b) Kompare

Obrázek 3.3: Výstupy rôznych *diff* nástrojov. Prevzaté z [28, 25].

Patch

Program *patch* je program, ktorý na vstupe dostane výstup z programu *diff* a zoznam súborov na ktoré aplikovať daný súbor zmien, takzvaný patchfile. Výsledkom sú nové súbory v ktorých sú aplikované dané zmeny. Syntax spustenia programu patch je na ukážke 3.3.

Ukázka 3.3: Spustenie programu *patch*

```
patch [prepínače] [originálny-súbor [zmenový-súbor]]
```

Cmp

Ďalším spôsobom ako je možné nazerať na textový súbor je nazerať na súbor ako na sekvenciu za sebou nasledujúcich bajtov. Nástroj *cmp* porovnáva dva súbory bajt po bajte a pokiaľ narazí na zmenu vypíše pozíciu a riadok na ktorom sa nachádza prvý rozdielný bajt, alebo oznámi, že jeden súbor je prefixom druhého. Číslovanie riadkov a bajtov začína od čísla 1. Ukážka zobrazuje syntax spustenia programu *cmp*. Pokiaľ sa niektorí so

Ukázka 3.4: Spustenie programu *cmp*

```
cmp [prepináče] ... súbor1 [súbor2 [vynechaj1 [vynechaj2]]]
```

vstupných súborov vynechá potom sa ako súbor použije obsah štandardného vstupu. Nepovinnými parametrami *Vynechaj1* a *Vynechaj2* sa dá špecifikovať počet začiatočných bajtov v jednotlivých súboroch, ktoré sa majú ingorovať pri porovnávaní.

Pri spustení programu *cmp* v základnej konfigurácii bez akýchkoľvek parametrov a v prípade že súbory sú obsahovo zhodné program nevypíše žiadny výstup, v prípade, že súbory sa líšia, alebo je jeden prefixom druhého program vypíše danú informáciu na štandardný výstup.

3.2 Porovnávanie na základe obsahu

3.2.1 Obrázky

Problematika porovnávania obrázkov na základe obsahu je založená na problematike získavanie informácií z obrázkov na základe obsahu anglicky nazývané *Content-based image retrieval* a podľa [4] vychádza z oboru počítačového videnia a je jeho podmnožinou. Rozdiel je badať v zameraní. Získavanie informácií z obrázkov na základe obsahu sa zameriava na selekciu množiny obrázkov na základe podobnosti charakteristík k cieľovému obrázku. [4] uvádza dve hlavné typy využitia v praxi:

- Určenie podobnosti dvoch obrázkov
- Nájdenie podobných obrázkov z množiny obrázkov

Určovanie podobnosti obrázkov prebieha na porovnávaní takzvaných rysov, alebo pospisovačov - anglicky features alebo descriptors. Tieto rysy sú z obrázka získané a následne sú uložené v dátovom úložisku pre následné použitie. Medzi používané rysy patria informácie o farbe, textúrach a útvaroch nachádzajúcich sa na obrázku. Jednotlivé typy rysov boli definované normou MPEG-7 Visual boli spomenuté v podkapitole 2.3.1.

3.2.2 Video

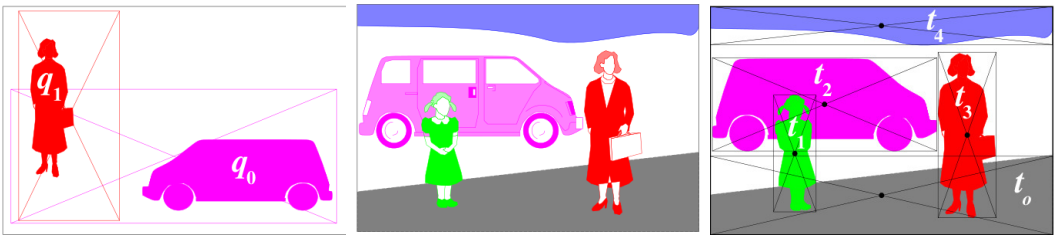
Ako bolo spomenuté v podkapitole 2.2, video sa skladá so sekvencie za sebou nasledujúcich snímkov. Pri porovnávaní obsahu videa je nutné z daných snímkov získať informáciu o obsahu daného snímku. Podľa [5] je získavanie informácií z videa na základe obsahu prirodzené rozšírenie problematiky získavania informácií z obrázkov na základe obsahu, ktoré bolo rozoberané v podkapitole 3.2.1. V podkapitole 2.2.1 boli spomenuté rôzne typy snímkov videa kodóvaného pomocou MPEG kompresie. Niektoré snímky slúžia ako referenčný snímok pre predikciu a odhad zmeny polohy objektov pri ich pohybe. Táto závislosť umožnuje zjednodušiť výpočet porovnávania a to tak, že nie je potrebné porovnávať všetky snímky ale postačí ak sa porovnájú snímky ktoré sú referenčné pre pohybujúce sa objekty v danom zábere. Tieto referenčné snímky sa môžu nazývať aj ako snímky kľúčové. Na obrázku 3.4 je ukážka kľúčových snímkov anotovaných MPEG.

Obrázek 3.4: Ukážka kľúčových snímkov.

3.2.3 Existujúce nástroje

VisualSEEk

VisualSEEk [31] je sýstém na získavanie informácii z obrázka na základe obsahu. Zisťovanie podobností prebieha na základe rozdelenia obrázka na regióny a následne sú z jednotlivých regiónov získane rysy a informácie o priestorovom rozložení a vzťahoch viď obrázok 3.5.



(a) (b) (c)

Obrázek 3.5: Rozklad obrázka (a) pôvodný obrázok, (b) cieľový obrázok, (c) cieľový obrázok rozdelený na regióny. Prevzaté z [31].

VisualSEEk bol vyvinutý v roku 1996 na Kolumbíjskej univerzite v New Yorku. Je vyvinutý v Jave a pozostáva s viacerích komponent – nástrojov, servera s obrázkami a videom, databázy metadát a indexovacích súborov.

Google Images

Google Images[1] ponúka možnosť vyhľadávania obrázkov na základe obrázka ktorý mu užívateľ zadá. Aktuálne podporuje možnnosť zadať obrázok nahraním, alebo zadaním URL adresy na ktorej sa obrázok nachádza. Po zadaní obrázku Google zobrazí zoznam podobných obrázkov. Na obrázku 3.6 je zobrazený výstup po zadaní hľadania obrázkov podobných k logu fakulty.

Obrázek 3.6: Výsek výstupu Google Images vyhľadávania na základe zadaného obrázka.

3.3 Vzdialenostné metriky

Na určenie či sú dva súbory zhodné alebo rozdielné je nutné určiť ich vzdialenosť. V tejto podkapitole sú predstavené niektoré z dostupných metrík.

Podľa [29] je metrický priestor definovaný ako ľubovolná neprázdna mnnožina X. Zobrazenie $d : X \times X \to \mathbb{R}$ ktoré spĺňa nasledujúce vlastnosti:

$$\rho(x, y) = 0 \longleftrightarrow x = y \tag{3.3}$$

$$\rho(x, y) = \rho(y, x) \tag{3.4}$$

$$\rho(x, y) + \rho(y, z) \geq \rho(x, z), \forall x, y, z \in X \tag{3.5}$$

Podmienka (3.4) sa nazýva symetria a (3.5) je zápis trojuholníkovej nerovnosti. Usporiadanú dvojicu (x, D) nazývame metrickým priestorom a zobrazenie d metrikou a X základnou množinou.

3.3.1 Vzdialenostné metriky textových dát

Výpočet vzdialenosti textových dát je založený na počítaní minimálneho počtu zmien ktoré je nutné urobiť v jednom reťazci tak, aby po aplikovaní týchto zmien boli porovnávané reťazce zhodné. V tejto podkapitole sú spomenuté dva algoritmy na určenie vzdialenosti textových súborov a to konkrétne Hammingová vzdialenosť a všeobecná Levenhsteinová vzdialenosť.

[1] Dostupné na stránke http://images.google.com/

Hammingová vzdialenosť

Hammingová vzdialenosť [20] dvoch rovnako dlhých textových reťazcov sa dá chápať ako počet zmien, ktoré je nutné vykonať aby tieto dva reťazce boli zhodné. Táto vzdialenosť je pomenovaná po americkom vedcovi Richardovi Hammingovi. Pôvodne bola Hammingová vzdialenosť určená pre zisťovanie chýb sposobených prenosom signálu. Pri určovaní Hammingovej vzdialenosti sa vychádza z predpokladu, že dané reťazce je možné transformovať na reťazec binárných hodnôt a to tak, že každý znak vo vstupných reťazcoch bude reprezentovaný binárnou hodnotou podľa nasledujúceho pravidla. Na danej pozícii bude hodnota 0 pokiaľ sa dané reťazce v danom znaku zhodujú, v opačnom prípade nadobudne hodnotu 1 viz. 3.5. na ktorom je vidieť, že dané reťazce majú Hammingovú vzdialenosť rovnú 2.

Ukázka 3.5: Príklad výpočtu Hammingovho binárneho reťazca

```
Lorem ipsum dolor sit amet
Lorem iqsum dolor sit Amet
00000010000000000000001000
```

Levenhsteinová vzdialenosť

Levenhsteinova vzdialenosť je podľa [35] definovaná nasledovne:

Abeceda Σ a Σ^* je množina reťazcov generovaných abecedou Σ. $\lambda \notin \Sigma$ je prázdny reťazec. Reťazec $X = \Sigma^*$ naznačený ako $X = x_1x_2 \dots x_3$ kde x_n je i-tý symbol reťazca X. $X_{i\dots j}$ je podreťazec X obsahujúci symboly $x_i \dots x_j$, $1 \leq i \leq j \leq n$ a ktorého dĺžka je daná ako $|X_{i\dots j}| = j - i + 1$ a je prázdny reťazec $\lambda(|\lambda| = 0)$ pokiaľ $i > j$. Základná editačná operácia je definovaná ako pár $(a, b) \neq (\lambda, \lambda)$ často značený ako $a \to b$, kde oba reťazce a a b sú dĺžky 1 alebo 0. Zápisy $\lambda \to a$, $a \to b$, $b \to \lambda$ značia tri základné eidtačné operácie – vloženie, nahradenie alebo zmazanie. $T_{X..Y} = T_1T_2 \dots T_l$ označuje editačnú transformáciu X na Y definovanú ako sekvenciu základných editačných operácií transformujúcich X na Y. Pokiaľ váhova funkcia γ priradí $a \to b$ kladné reálne číslo $\gamma(a \to b)$ potom váha editačnej transformácie $T_{X..Y}$ je možné vypočítať ako $\gamma(T_{X..Y}) = \sum_{i=1}^{l} \gamma(T_i)$.

Pri daných reťazcoch $X, Y \in \Sigma$ je všeobecná Levenhsteinová vzdialenosť, anglicky nazývaná GLD definovaná ako

$$GLD(X, Y) = min\{\gamma(T_{X,Y})\} \tag{3.6}$$

GLD tvorí metriku nad Σ^* pokiaľ sú splnené nasledovné podmienky:

$$\forall a, b \in \Sigma \cup \{\lambda\} \tag{3.7}$$

$$\gamma(a \to a) = 0 \tag{3.8}$$

$$\gamma(a \to b) > 0, a \neq b \tag{3.9}$$

$$\gamma(a \to b) = \gamma(b \to a) \tag{3.10}$$

3.3.2 Vzdialenostné metriky obrazových dát

Vzdialenostné metriky obrazových dát patria metriky založené na porovnávaní obsahu jednotlivých obrázkov. Informácie ktoré popisujú obsah obsah sa nazývajú popisovače alebo

taktiež rysy. Príklady rysov definovaných normou MPEG-7 Visual boli uvedené v podkapitole 2.3.1. Či už je porovnávanie založené na získaných rysoch alebo na priamočiarom porovnávaní zhody jednotlivých obrazových bodov k dispozícií je široké spektrum metrík. Porovnanie vybraných metrík je možné nájsť v práci Hampapura a Bolleho [6].

Rozdielnosť obrázkov

Najjednoduchším spôsobom ako určiť rozdielnosť dvoch obrázkov je porovnať a zistiť rozdielnosť jednotlivých obrazových bodov. Metrika *Rozdielnosť obrázkov* je založená na sumovaní absolútnej hodnoty rozdielov obrazových bodov naprieč farbenými kanálmi. Získaná hodnota je následne normalizovaná počtom obrazových bodov krát počet farebných kanálov. Rovnica (3.11) vyjadruje vzťah na výpočet vzdialenosti dvoch obrázkov.

$$D_{id} = \frac{\sum_{x=0}^{W-1}\sum_{y=0}^{H-1}\sum_{c=0}^{ch-1}|I_1(x,y,c) - I_2(x,y,c)|}{ch \times W \times H} \tag{3.11}$$

Porovnávanie vektorov

Metriky na porovnávanie vektorov je možné použiť k porovnávaniu napríklad rôznych histogramov alebo aj rysov ktorých výstup má formu vektora, prípadne vektorov.

Výpočet prebieha nad n-dimenzionálnym vektorom $F = \{f_1, f_2, \dots f_n\}$ ktorý je možné transformovať na bod $P(f_1, f_2, \dots, f_n)$ v n-dimenzionálnom priestore [33]. Po tejto transformácií je možné pri porovnávaní použiť klasické priestorové metriky ako manhattanskú metriku, Euklidovú metriku alebo chi-square metriku. Pre dva body $a = (a_1, a_2, a_3, \dots, a_n)$ a $b = (b_1, b_2, b_3, \dots, b_n)$ v n-dimenzionálnom priestore je Manhattanská vzdialenosť definovaná rovnicou (3.12), Euklidová metrika rovnicou (3.13) a chi-square metrika rovnicou (3.14).

$$d_M(a,b) = |a_1 - b_1| + |a_2 - b_2| + \ldots + |a_n - b_n| = \sum_{i=1}^{n}|a_i - b_i| \tag{3.12}$$

$$d_E(a,b) = \sqrt{(a_1 - b_1)^2 + (a_2 - b_2)^2 + \ldots + (a_n - b_n)^2} = \sqrt{\sum_{i=1}^{n}(a_i - b_i)^2} \tag{3.13}$$

$$d_\Xi(a,b) = \sum_{i=1}^{n}\frac{(a_i - b_i)^2}{a_i + b_i} \tag{3.14}$$

Hausdorffova vzdialenosť

Metrika založená na princípe porovnávania reprezentácie hrán jednotlivých obrazkov. Obrázky sú prevedené do monochromatického farebného modelu a následne sú z nich zístené informácie o hranách objektov nachádzajúcich sa na obrázku. Rozdielnosť hranových reprezentácií je vypočítana rovnicou (3.15), kde $h_p(I_1, I_2)$ a $h_p(I_2, I_1)$ predstavujú čiastočné Hausdorffove vzdialenosti.

$$D_{haus} = max(h_p(I_1, I_2), h_p(I_2, I_1)) \tag{3.15}$$

Metrika reprezntácie lokálnych hrán

Princípom tejto metriky je prevod obrázkov do čierno-bielej farebného modelu a následnom vyextrahovaní hraničných bodov, ktoré sú rozdelené do $n_1 x n_2$ okien. I-té okno nadobúda hodnotu c_i, získanú kvantovaním pozícií centroidu hraničných bodov tohto okna. Rozdielnosť je daná rovnicou (3.16).

$$D_{led} = \frac{\sum_{i=0}^{n_1 \times n_2} \begin{cases} 0 & c_i(I_1) = c_i(I_2) \\ 1 & c_i(I_1) \neq c_i(I_2) \end{cases}}{n_1 \times n_2} \tag{3.16}$$

Kapitola 4

MediaDiff

MediaDiff je opensource aplikácia na porovnávanie viacerých druhov dokumentov a formátov na základe ich obsahu. MediaDiff[1] je v súčastnosti šíreny pod licenciou GPLv3. MediaDiff je vyvýjaný na fakulte informačných technológií Vysokého Účení Technického v Brne v spolupráci s firmou Red Hat. Cieľom projektu vytvoriť modulárny nástroj schopný porovnávať viacero druhov dokumentov a formátov. Jadro aplikácie je implementované v jazyku Python ale niektoré moduly aplikácie sú implmementované v jazyku C++. Z knižníc implementovaných v jazyku C++ sú následne nástrojmi na tvorbu modulov do programovacieho jazyka Python vytvorené moduly použiteľné v prostredí jazyka Python.

Jadrom aplikácie je modul ktorý tvorí vstupný bod do aplikácie a podľa potreby spúšťa dostupné moduly. Modulov na porovnávanie textových súborov je viacero. Modul *TextDiff* obsahuje funkčnosť porovnávania obyčajných textových súborov. Podporuje rôznu doplnkovú funkčnosť ako ignorovanie bielych a prázdnych riadkov. Existuje možnosť selekcie riadkov alebo slov definovaním regulárneho výrazu. Ďalšími modulmi na porovnávanie textových súborov sú moduly *LatexDiff*, *ConfigDiff* a *SemanticDiff*. Modul *LatexDiff* je schopný porovnať zdrojový text sádzacieho systému LaTex. Funguje na princípe očistenia zdrojového kódu na úroveň čistého textu. Podobne ako modul *TextDiff* podporuje nastavenie doplnkových parametrov. Modul *ConfigDiff* slúži na porovnávanie konfiguračných súborov založených na formáte používanom v prostredí operačného systému UNIX. Porovnávanie sa sústredí na sémanticku informáciu a nie je citlivé na poradie jednotlivých riadkov v porovnávaných súboroch. Na porovnávanie zdrojových kódov programovacích jazykov slúži modul *SemanticDiff*. V súčastnosti tento modul podporuje jazyky C/C++, Java a Python. Modul *FileInfoDiff* slúži na porovnávanie súborových informácií ako napríklad názov súboru, typ súboru alebo prístupových práv k porovnávaným súborom.

Porovnávanie multimédií v súčastnosti ponúkajú moduly *SoundDiff* a *ImageMagick*. Modul *SoundDiff* slúži na porovnanie dvoch zvukových záznamov a nájdenie rozdielov počuteľných ľudským uchom. Na prácu so zvukom využíva knižnicu Kaldi a porovnávanie je založené na skrytých Markovových modeloch. Modul *ImageMagick* porovnáva statické obrázky.

Ďalšie informácie k jednotlivým modulom aplikácie mediaDiff je možné nájsť na domovskej stránke projektu alebo v prácach [36, 37, 10, 2].

[1] Domovskú stránku projektu je možné nájsť na adrese *http://gitorious.org/mediadiff*

Kapitola 5

Návrh a implementácia

Táto kapitola sa zaoberá návrhom a implementáciou aplikácie ktorá je predmetom tejto práce. Pri návrhu a implementácii boli využité teoretické základy problematiky opísanej v predchádzajúcich kapitolách. Prvá časť kapitoly sa venuje analýze požiadaviek a následnému návrhu aplikácie. Zvyšná časť kapitoly je venovaná použitým technológiam a pojednáva o implementácii navrhovanej aplikácie.

5.1 Návrh aplikácie

Táto podkapitola je venovaná návrhu aplikácie ktorá je predmetom tejto práce. V úvode sú uvedené požiadavky ktoré sú stanovené na navrhovanú aplikáciu. Zbytok kapitoly je venovaný návrhom jednotlivých častí aplikácie.

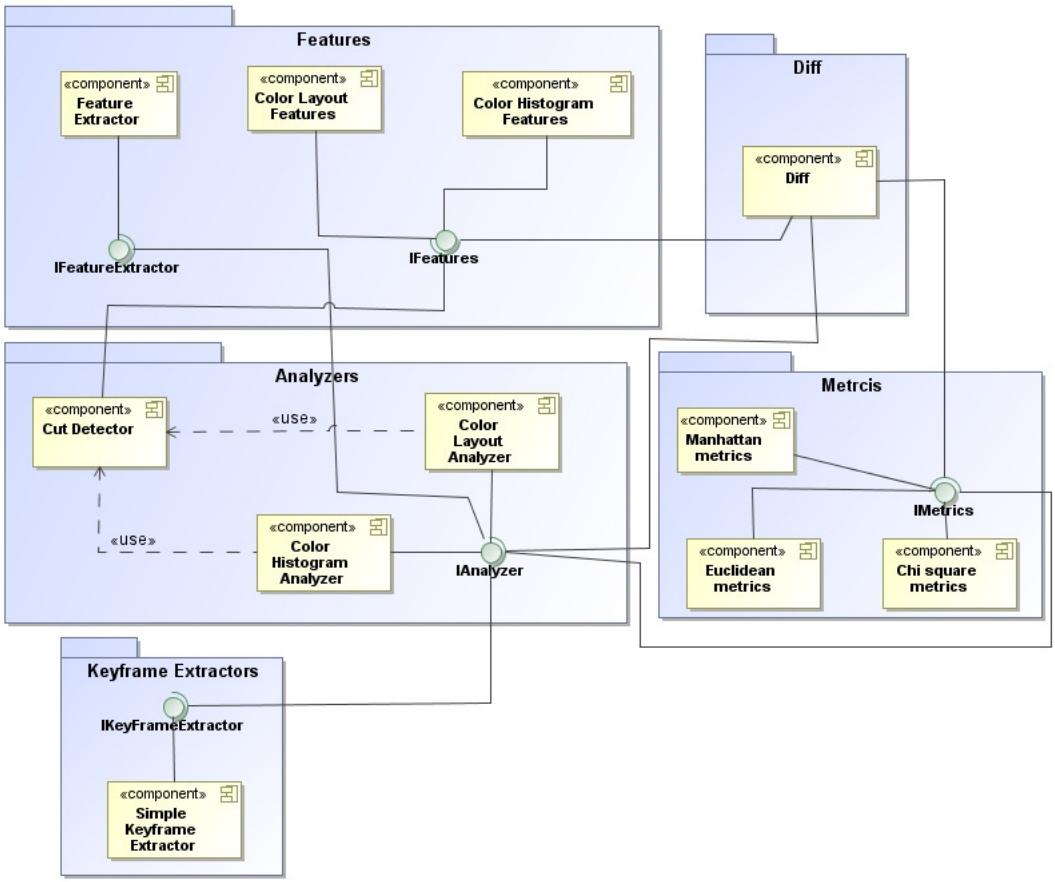
Na navrhovanú aplikáciu okrem funkčných požiadaviek vyplyvajúcich zo zadnia sú stanovené aj rôzne iné funkčné a nefunkčné požiadavky ktoré priamo nevyplývajú zo zadania tejto práce. Základnou požiadavkou je implementovať funkčnosť porovnávania videí na základe obsahu a to zisťovaním rozdielnych snímkov v jednotlivých záberoch. Medzi požiadavky definované zadaním práce patrí aj nutnosť integrácie vytvorenej aplikácie s aplikáciou mediaDiff . Medzi požiadavky ktoré nevvyplyvajú so zadania práce súvisia s požiadavkami na ľahké rozširovanie funkčnosti a jednoduchá udržovateľnosť. Ďalšou požiadavkou je možnosť integrácie vzniknutej aplikácie s rôznymi inými aplikáciami.

V kapitole 3 boli spomenuté rôzne techniky na porovnávanie súborov na základe obsahu. Základným predpokladom je práca so sekvenčnými dátami, teda dátami ktoré sa dajú sekvenčne zoradiť. Pri porovnávaní videí je táto vlastnosť zaručená z princípu na akom video funguje. Ako bolo spomenuté v kapitole 2.2 video sa skladá zo snímkov. Je preto potrebné aby navrhovaný algoritmus pracoval s týmito snímkami na úrovni záberov. Pre dosiahnutie požadovanej funkčnosti je nutné najprv detekovať hranice jednotlivých záberov a následne z každého záberu vybrať množinu snímkov na základe ktorých bude samotné porovnávanie prebiehať. Je dôležité aby algoritmus výberu snímkov zo záberu vybral pri záberoch ktoré sú zhodné vždy zhodné snímky.

Samotné porovnávanie musí byť založené na porovnávaní obsahu jednotlivých snímkov. Pre splnenie tejto podmienky je nutné porovnávať obsah jednotlivých snímkov. Pre zefektívnenie algoritmu budú do porovnávania vstupovať metadata popisujúce obsah jednotlivých snímkov – rysy alebo popisovače. Metadatám bola venovaná kapitola 2.3. Použitý algoritmus na nájdenie najdlhšej spoločnej postupnosti musí byť schopný porovnávať dáta ktoré sa môžu líšiť v čase a rýchlosti. Zároveň je potrebné aby výstup z použitého algo-

ritmu bolo možné zobraziť vo forme štandardného výstupu pre programy typu *Diff*. Taktiež je nutné aby jednotlivé parametre vstupujúce do algoritmu boli používateľsky jednoducho upraviteľné.

Na obrázku 5.1 je zobrazený komponentový návrh navrhovanej aplikácie. Vychádzajúc z požiadaviek na návrh bolo rozhodnuté, že navrhovaná aplikácia bude mať formu knižnice a prepojenie s aplikáciou mediaDiff bude riešené pomocou samostatného rozhrania. Zvolený spôsob implementácie zaručuje, že navrhovaný modul má s aplikáciou mediaDiff len slabú väzbu a komunikácie prebieha len cez definované rozhranie. Zároveň je splnená požiadavka na prípadne použitie vytvorenej aplikácie v komibnácii s inými programovacím prostredím alebo aplikáciou.



Obrázek 5.1: Komponentový návrh základnej knižnice.

Ako je možné vidieť na obrázku 5.1 základná knižnica navrhovaného modulu je rozdelená do viacerích celkov. Základným stavebným prvkom je súbor komponent *Analyzers* ktoré zapúzdrujú základnú sadu operácií. Sada komponent *Features* je zodpovedná za extrakciu rysov zo zvoleného videa. Pri návrhu boli uvažované dva základné typy rysov *Color layout* a *Color histogram*.

Rysy *Color layout* sú definované normou MPEG-7. Tieto rysy sú invariantné voči rozmerom spracovávaného snímok. Táto vlastnosť je zabezpečená viackrokovým spracovávaním. V prvom kroku sa snímok rozdelí na 64 rovnakých blokov. Následne je v každom bloku určená reprezentatívna farba a vytvorí sa obrázok o veľkosti 8x8 pozostávajúci s týchto reprezentatívnych farieb. Následne je na vzniknutý obrázok aplikovaná DCT transformácia

a získané hodnoty sú zoradené podľa daného vzoru.

Rysy *Color histogram* sú rysy popisujúce zastúpenie farby v snímku. V rámci návrhu sa počíta z jednoduchou implementáciou kvantujúcov farby do vopred daného počtu košov.

Po získaní rysov z videa sú následne pomocou komponenty *CutDetector* detekované jednotlivé strihy a tým hranice jednotlivých záberov. Pre každý nájdený záber je určená množína snímkov z tohto záberu ktorá vstúpi do algoritmu porovnávania. Výber jednotlivých snímok má na starosti komponenta *Keyframe Extractor*. Zvolené snímky sú následne v komponente *Diff* porovnané a výsledky sú zobrazené užívateľovi. Ako algoritmus implementovaný v rámci komponenty *Diff* bol zvolený algoritmus dynamického ohýbania času (DTW) nakoľko spĺňa všetky definované podmienky.

5.2 Technológie

Ako bolo spomenuté v podkapitole navrhovaná aplikácia bude mať formu modulu v rámci aplikácie mediaDiff. Navhrnutý modul pozostáva z viacerích častí na ktoré sú kladené rôzne požiadavky. V tejto podkapitole sú diskutované technológie ktoré sú použité pri implementácií.

5.2.1 FFmpeg

Nakoľko hlavná funkčnosť navrhovanej aplikácie súvisí so spracovaním videa bolo rozhodnuté, že práca s videom bude zastrešená funkcionalitou knižnice FFmpeg.

FFmpeg [24] je multimediálny framework určený na prácu s multimédiami šírený pod licenciou GPL. Projekt bol založený Fabrice Bellardom v roku 2000. Podporuje široké spektrum formátov od historických až po najnovšie. Cieľom projektu je poskytovať vývojárom a bežným koncovým užívateľom najlepšie technické riešenie. FFmpeg sa skladá z viacerích knižníc a programov. Programy ktoré sú súčasťou frameworku sú ffmpeg, ffserver, ffplay a ffprobe. Okrem týchto programov obsahuje aj súbor knižníc kde patrí knižnica *libavcodec* obsahujúca množstvo kóderov a dekóderov audiovizuálnych dát. Množstvo funkcií a dátových štruktúr vhodných na prácu s multimédiami ktoré zlepšujú a zjednodušujú prácu obsahuje knižnica *libavutil*. Široké spektrum multiplexorov a demultimplexorov pre rôzne multimediálne kontajnery obsahuje knižnica *libavformat*. Knižnica *libavdevice* obsahuje množstvo vstúpno-výstupných zariadení pre prehrávanie a zobrazovanie vo rôznych multimediálnych frameworkoch. Knižnica *libavfilter* obsahuje množstvo filtrov pre rôzne média, *libswscale* slúži na efektívne prevody medzi rôznymi pixel formátmi a knižnica *libswresample* slúži na rýchle prevzorkovávanie audio súborov.

Knižnica FFmpeg bude v implementovanej aplikácii použitá na prácu s videosúbormi a to konkrétne na extrakciu snímkov z videosúborov a ich prípravu na ďalšie spracovanie.

5.2.2 Programovacie jazyky

Navrhovaná aplikácia pozostáva z dvoch častí. Prvú časť predstavuje knižnica ktorá bude zapúzdrovať všetku funkčnosť potrebnú k porovnávaniu videa. Na túto časť sú kladené nároky na efektívnosť a rýchlosť. Zároveň je dôležité aby bolo možné vo zvolenom programovacom jazyku použiť knižnicu FFmpeg. Po zvážení jednotlivých požiadavaiek bolo rozhodnuté, že knižnica bude napísaná v jazyku C++.

Jazyk C++ je objektovo orientovaný jazyk ktorý bol vytvorený v roku 1979 Bjarne Stroustrupom. Jazyk C++ vznikol ako objektovo orientovaná nadstavba jazyka C [13] s

podporou šablón. Kompilátory jazyka C++ sú k dispozícií pre veľke množstvo platforiem.

Druhou časťou aplikácie je napojenie knižnice obsahujúcej funkčnosť do aplikácie mediaDiff. Jadro aplikácie mediaDiff je napísané v jazyku Python. Preto bolo rozhodnuté, že sa pre vzniknutú knižnicu vytvorí wrapper pre jazyk Python.

Jazyk Python bol vytvorený začiatkom 90-tych rokoch Guido van Rossum. Python je moderný objektovo orientovaný interpretovaný jazyk s otvoreným kódom ktorý je nezávisly na platforme. Je význačný jednoduchou syntaxou a rýchlemu vývoju aplikácií. V porovnaní s kompilovanými jazykmi C/C++ je pomalši, ale umožňuje použitie kódu napísaného v C/C++. Súčasťou jazyka Python je aj automatická správa pamäte [23].

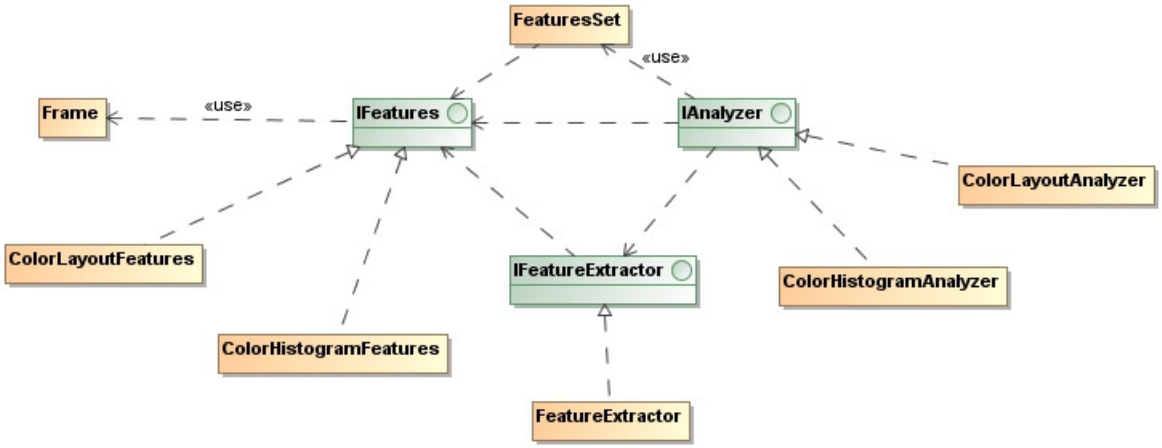
5.3 Implementácia

Táto podkapitola sa venuje imlplementácií navrhovanej aplikácie. V rámci tejto podkapitoly budú jednotlivé komponenty rozobraté do vačšej hĺbky a pozornosť bude venovaná jednotlivým rozhodnutiam ktoré boli urobené počas fázy implementácie.

Základom porovnávania na základe obsahu je získanie informácii o obsahu snímkov – rysov. Získavanie jednotlivých snímkov videa je implementované pomocou knižnice FFmpeg. Ako už bolo spomenuté táto knižnica podporuje prácu so širokým spektrom formátov. Počas implementácie boli implementované dva druhy rysov – *Color layout* a *Color histogram*. Tieto rysy boli zvolené kvôli svojej rýchlosti a predpokladaným dobrým výsledkom.

Funkčnosť extrakcie rysov je implementovaná v triedach ktoré implementujú rozhranie *IFeatures*. Toto rozhranie obsahuje základnú sadu metód potrebných pre prácu so získanými rysmi z jednotlivých snímkov. Jednotlivé extrahované rysy sú vyjadrené samostatnými triedami a v týchto triedach sa uchovávajú zároveň získané rysy. Identifikácia o snímku z ktorého boli rysy získané je vyjadrená poradovým číslom snímku. Z extrahovaných rysov už okrem čísla snímku neexistuje na žiadna väzba na snímok. Nemožnosť dohľadať celý snímok je daná z princípu na akom FFmpeg pracuje. Jednotlivé načítané snímky zdieľajú rovnaké pamäťové miesto a v počas posunu sa menia. Extrakcia rysov je obsluhovaná cez továrenskú triedu implementujúcu rohranie *IFeatureExtractor*. Triedy a rohrania *IFeatures* a *IFeatureExtractor* slúžia na prácu s jedným osamostatneným snímkom. Pre prácu z videom ako celkom boli implementované triedy implementujúce rohranie *IAnalyzer*. Rozhranie *IAnalyzer* poskytuje kompletnú sadu funkcií ktoré slúžia na spracovanie videa. Nakoľko sa práca s jednotlivými typmi získavaných rysov môže líšiť pre jednotlivé typy implementovaných rysov musí byť implementovaná trieda implementujúca rozhranie *IAnalyzer* a obsluhujúca prácu s daným typom rysov. Použitie správneho analyzátora je implementované použitím návrhového vzoru továreň. Nakoľko triedy implementujúce rozhranie *IAnalyzer* pracujú s videom ako celkom je nutné získané rysy uložiť ako celok aby mohli byť spracovávané naraz. Na uloženie získaných rysov slúži trieda *FeatureSet* ktorá je nezávisla na použitých rysoch. Namodelovaný vzťah je možné vidieť na obrázku 5.2.

Spracovanie snímku začína transformovaním snímkového formátu do formátu RGB a následne je aplikovaná transformácia za účelom zníženia rozmerov snímku. Túto transformáciu je možné parametricky upravovať a jej účel je zníženie doby spracovania snímku nakoľko ten je závislí od veľkosti snímku. Pri extrakcii rysov *Color histogram* je získaný histogram pre jednotlivé farebné zložky. Implementácia kvantuje získané hodnoty do 128 košov ktoré následne normalizuje do intervalu $< 0; 100 >$. Normalizáciou sa získa pomerové zastúpenie farieb v snímku čo zabezpečuje invariantnosť voči rozlíšeniu snímku. Implementácia získavania rysov *Color layout* je v súlade s normou MPEG-7 ktorá bola popísaná v podkapitole 2.3.1. Prvým krokom je načítanie každej zložky RGB modelu v celom obrázku a následne sú



Obrázek 5.2: Diagram hlavných tried vstupujúcich do extrakcie rysov.

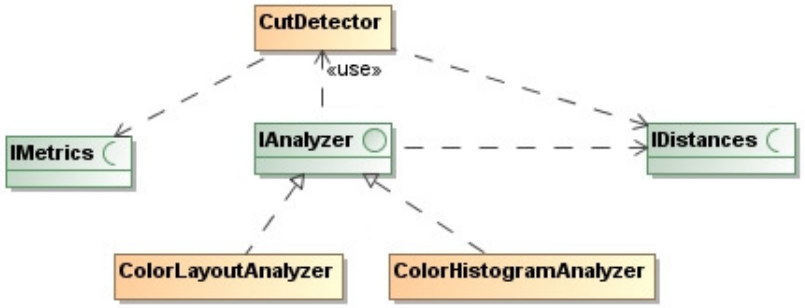
tieto zložky rozdelené na 64 blokov čo zabezpečuje invariantnosť voči rozlíšeniu. Druhým krokom je získanie reprezentatívnej a dominantnej farby v každom jednom bloku. Zistenie dominantnej farby je založené na získaní priemeru hodnôt farieb v danom bloku. Táto metóda je jednoduchá a postačuje na účely cieľovej aplikácie. Získané dominantné farby vytvárajú obrázok o veľkosti 64 obrazových bodov. Tento obrázok je transformovaný do YCbCr farebného modelu a je naň aplikovaná diskrétna kosínusová transformácia. Po aplikovaní diskrétnej kosínusovej transformácie je získaných 64 koeficientov v každej farebnej zložke modelu YCbCr. Tieto koeficienty sú zoskenované pomocou zig-zag skenera ktorého definiícia je v tabuľke 5.1. Na výstupe po aplikácii zig-zag skenera sú 3 vektory DCT koeficientov zoradené podľa zig-zag predpisu. Z týchto koeficientov je následne vybraných prvých 20 pre farebnú zložku Y a po 15 pre farebné zložky Cb a Cr.

0	1	5	6	14	15	27	28
2	4	7	13	16	26	29	42
3	8	12	17	25	30	41	43
9	11	18	24	31	40	44	53
10	19	23	32	39	45	52	54
20	22	33	38	46	51	55	60
21	34	37	47	50	56	59	61
35	36	48	49	47	58	62	63

Tabulka 5.1: Koeficienty zig-zag skenera.

Ďalším krokom je detekcia záberov nachádzajúcich sa vo videu. Pre určenie záberov je potrebné nájsť miesta strihu medzi jednotlivými zábermi. Na tento účel slúži trieda *CutDetector* ktorá zapúzdruje celú funkcionalitu zisťovania miest strihov. Strih je náhly prechod medzi dvoma snímkami. Použitý algoritmus vychádza z tejto definície a zisťuje vzdialenosť susedných snímkov. Výpočet prebieha nad získanými rysmi zo všektých snímkov videa. Zisťuje vzdialenosť každej dvojice susedných snímkov. Pre určenie vzdialenosti sú použité vzdialenostné metriky. Bola implementovaná Euklidová metrika (3.13), chi-square metrika (3.14) a Manhattanská metrika (3.12). Jednotlivé vzdialenosti sú uchovavávané v triede implementujúce rozhranie *IDistance* ktoré je definované v rámci rozhrania *IDistances*. Po získaní vzdialeností všetkých susedných snímkov sú získané vzdialenosti normalizované do

intervalu $< 0; 1 >$ pre získanie percentuálnej vzdialenosti a tým lepšej výpovednej hodnoty. Následne je zo všetkých snímkov vypočítaná priemerná vzdialenosť a smerodajná odchylka. Diagram tried predstavujúci túto časť aplikácie je na obrázku 5.3.

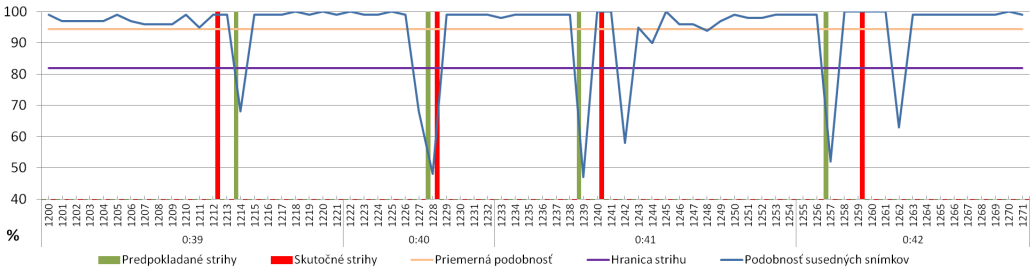


Obrázek 5.3: Diagram tried vstupujúcich do detekcie hraníc záberov.

Hranica strihu je hodnota ktorá predstavuje deliacu hodnotu medzi miestami kde sa môže nachádzať strih a kde sa určite nenachádza. Táto hodnota sa vypočíta z priemernej podobnosti a smerodajnej odchylky podľa vzťahu (5.1). $\bar{d}$ je priemerná hodnota vzdialeností susedných snímkov, σ predstavuje smerodajnú odchylku a n je konštanta ktorú zadáva používateľ a tým ovplyvňuje priebeh algoritmu.

$$H_s = \bar{d} + n\sigma \tag{5.1}$$

Miesta kde vzdialenosť medzi susednými snímkami je vačšia ako vypočítaná hodnota H_s sú miesta pravdepodobného strihu. Pre určenie konkrétneho miesta strihu je použité plávajúce okno o dĺžke ktoru zadá používateľ a v rámci ktorého sa hľadajú miesta najvyššej vzdialenosti. Tieto miesta sú určené ako miesta strihu. Plávajúce okno prechádza získané hodnoty zľava do prava. Na obrázku 5.4 je znázornený výsledok algoritmu zisťovania miest strihov použitom na krátkej ukážke. Modrá čiara predstavuje podobnosť dvoch susedných snímkov normalizovanú do intervalu $< 0; 1 >$. Fialová čiara je hranica strihu. Pri výpočte bola použitý 1-násobok smerodajnej odchylky. Červené čiary sú miesta skutočných strihov anotovaných ručne. A zelené miesta ktoré označil algoritmus. Pri výpočte bola použitá Euklidová metrika a *Color layout* rysy s plávajúcim oknom o šírke 10 snímkov. Osa y predstavuje mieru podobnosti v percentách.



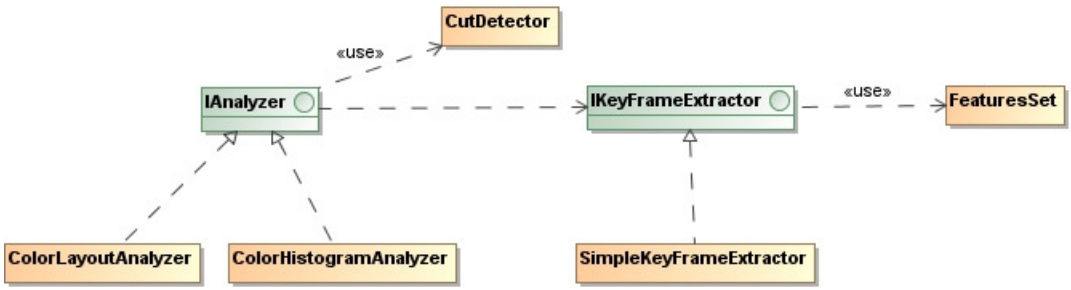
Obrázek 5.4: Detekované hranice záberov.

Po určení jednotlivých záberov je z každého záberu určená množina snímkov ktoré budú použité pri samotnom porovnávaní. Na algoritmus výberu týchto snímkov boli v podkapitole návrhu určené požiadavky. Je nutné aby algoritmus vybral pri rovnakých záberoch rovnaké

snímky bezohľadu na to, koľko snímkov tvorí daný záber. Táto vlastnosť je nutná z dôvodu porovnávania videí s rôznym počtom snímkov za sekundu nakoľko záberu v tomto prípade obsahujú rôzny počet snímkov. Implementovaný bol jednoduchý algoritmus pozostávajúci s nasledujúcich krokov:

1. Určenie dĺžky záberu
2. Výpočet dĺžky kroku medzi dvoma reprezentatívnymi snímkami
3. Výber snímkov

Algoritmus v prvom kroku určí dĺžku záberu, tj. zistí koľko snímkov sa v danom zábere nachádza. Následne tento záber rozdelí tak aby pri výbere reprezentatívnych snímkov pokryl celý záber. Rozdelenie záberu je závisle na počte snímkov ktoré chceme zo záberu získať. Počet snímkov je možné parametricky upraviť. Následne po rozdelení intervalu na časti o dĺžke medzi dvoma kľúčovými snímkami sú tieto hraničné snímky označené a zaradené do ďalšieho kroku výpočtu. Tento algoritmus je implementovaný v triede *SimpleKeyFrameExtractor* ktorá implementuje rozhranie *IKeyframeExtractor*. Toto rozhranie musia implementovať všetky dostupné algoritmy na výber snímkov. Použitie rozhrania taktiež znížilo väzbu a pracnosť prípadnej implementácie ďalšieho algoritmu na výber snímkov. Na obrázku 5.5 je zobrazený diagram tried vstupujúcich do tejto časti výpočtu.



Obrázek 5.5: Diagram tried vstupujúcich do výberu kľúčových snímkov.

Na určenie rozdielov je nutné nájsť najdlhšiu spoločnu postupnosť a to predovšetkým s ohľadom na prípadné zmeny v čase. Porovnávanie a určovanie rozdielov je implementované v komponentne *Diff* ktorý zapúzdruje túto funkčnosť. Implementovaný bol algoritmus dynamického ohýbania času ktorý bol spomenutý v podkapitole 3.1.3. Tento algoritmus ako už bolo spomenuté je určený na porovnávanie dát ktoré sa môžu líšiť v rýchlosti a čase. Pri porovnávaní berie v úvahu aj okolie porovnávaného prvku čo ma za následok zvýšenú presnosť ale na druhú stranu zvyšuje to časovú náročnosť algoritmu. Pri porovnávaní slúžia zvolené snímky z predchádzajúceho záberu ako jednotlivé prvky v porovnávanej sekvencii. Samotné prvky sú reprzentované triedou *Item* ktorá predstavuje jeden prvok a teda jeden snímok. Táto trieda obsahuje metódu ktorá preťažuje operátor == a v rámci ktorého je implementované porovnanie dvoch snímkov. Algoritmus porovnania je totožný s tými algoritmami čo boli použité v predchádzajúcich častiach. Vzdialenosť je vypočítaná so získaných rysov týchto snímkov použitím zvolenej vzdialenostnej metriky. Dva prvky sú označené ako zhodné pokiaľ ich vzdialenosť je väčšia ako zvolená presnosť – vzdialenosť ktorá je tolerovaná. Úroveň presnosti je možné používateľsky parametricky upravovať nakoľko jej hodnota dokáže ovplyvniť získané výsledky. Z vypočítaných hodnôt je následne

vytvorený editačný skript ktorý je zobrazený užívateľovi na štandardný výstup. Výstup obsahuje tri druhy informácie:

- Výstup 00:12.32,00:22.85d00:27.99 značí, že časť od 00:12.32 po 00:22.85 v prvom videu bola vymazaná
- Výstup 01:48.05,01:59.15c01:15.32,01:15.65 značí, časť od 01:48.05 až po 01:59.15 v prvom videu bola nahradená časťou 01:15.32 až 01:15.65 v druhom videu
- Výstup 02:09.09,02:13.45a01:35.90 značí, že časť 01:35.90 v druhom videu pridaj časť 02:09 až 09,02:13.45 s prvého videa

Pokiaľ sa na výstupe nachádza viac zmien sú tieto jednotlivé zmeny vypísane tak, že každá zmena je napísana na samostatnom riadku viď obrázok 5.1.

Ukázka 5.1: Príklad výstupu použitého algoritmu diff.

```
00:12.32,00:22.85d00:27.99
01:48.05,01:59.15c01:15.32,01:15.65
02:09.09,02:13.45a01:35.90
```

Implementovaná aplikácia obsahuje viacero voliteľných parametrov ktorými je možné ovplyvňovať priebeh algoritmu. Ukážka 5.2 obsahuje výpis dostupných parametrov. Celkovo aplikácia obsahuje 7 voliteľných parametrov. Parameter *features* slúži na určenie typu používaných rysov. Ako už bolo spomenuté implementované boli dva typy rysov – *Color layout* a *Color histogram*. Parameter *kfextractor* slúži na určenie typu algoritmu ktorý použiť pri výbere snímkov z každého záberu. V rámci implementácie bol implementovaný len jeden algoritmus. Na určenie typu použitej vzdialenostnej metriky slúži parameter *metrics*. Parameter *kfpershot* slúži na určenie počtu snímkov ktoré vybrať z každého nájdeného záberu. Parametre *windowsize* a *cutthreshold* slúžia na parametrizovanie algoritmu na určovanie hraníc záberov. Ako bolo spomenuté implementovaný algoritmus porovnávania je parametrizovateľný takzvanou citlivosťou. Parameter *similaritythreshold* slúži na nastavenie citlivosti. Aplikácia taktiež umožnuje zmenšiť velkost porovnávaných snímkov. K tomuto účelu slúží parameter *ratio*. Prepínač *positions* slúži na určenie údajov zobrazených v editačnom skripte. Pokiaľ je prepínač použitý sú na výstupe zobrazené pozície snímkov namiesto časov.

Ukázka 5.2: Dostupné aplikačné parametre.

```
videodiff.py [-h] [--features {1,2}] [--kfextractor {1}]
                  [--metrics {1,2,3}] [--kfpershot KFPERSHOT]
                  [--windowsize WINDOWSIZE] [--cutthreshold CUTTHRESHOLD]
                  [--similaritythreshold SIMILARITYTHRESHOLD]
                  [--ratio RATIO] [--positions]
                  file1 file2
```

Kapitola 6

Testovanie

Dôležitou časťou vývojového cyklu každého softvérového produktu je testovanie. Pri implementácií akejkoľvek aplikácie je nutné overiť jej funkčnosť. Táto kapitola je venovaná testovaniu vytvorenej aplikácie. Kapitola je rozdelené do troch častí. V prvej časti kapitoly sú predstavené testovacie vzorky a metriky ktoré budú následne overované a testované. Druhá časť testuje správnosť a presnosť implementovaných rysov a vzdialenostných metrík. V tejto časti sú taktiež diskutované parametre ktorými je možné aplikáciu parametrizovať ako aj ich vplyv na priebeh porovnávania. Posledná časť kapitoly je venovaná testovaniu samotnej funkcionality porovnávania dvoch videí na základe obsahu pri rôznych rysoch a vzdialenostných metrikách. V rámci tejto časti sú predstavené výsledky jednotlivých testovacích vzoriek s ich zhodnotením.

6.1 Nastavenie parametrov

Pokiaľ má akýkoľvek algoritmus správne pracovať je nutné aby boli vstupy správne a algoritmus ich správne interpretoval. Vzhľadom na povahu navrhnutej aplikácie boli niektoré časti navrhnuté ako parametrizovateľné. Táto podkapitola je venovaná nastaveniam týchto parametrov v rámci testovania.

Navrhnutú aplikáciu je možné parametrizovať rôznymi parametrami ktoré môžu dosť zásadne ovplyvniť priebeh algoritmu. Ukážka ... obsahuje množinu všetkých dostupných parametrov ktoré je možné v aplikácii nastaviť.

6.2 Testovacie dáta

Ako testovacie vzorky boli vybrané voľne dostupné ukážky k celovečerným filmom. Po výbere testovacích vzoriek boli z týchto vzoriek vytvorené ďalšie vzorky aplikovaním rôznych zmien aby bolo možné otestovať viaceré aspekty implementovanej aplikácie ktorá pozostáva z viacerích kľúčových častí. Pred samotným výberom testovacích vzoriek bolo rozhodnuté aké metriky budú testované. Bolo rozhodnuté, že testovanie bude testovať nasledovné tri metriky:

- Presnosť pri určovaní hraníc záberov – miest strihov
- Presnosť porovnávania rovnakých videosúborov s rozdielným kódovaním alebo počtom snímkov za sekundu

- Presnosť porovnávania zmenených a rozdielných videosúborov

Ako testovacie vzorky bolo pripravených 6 videosúborov ktorých základný popis je zobrazený v tabuľke 6.1. Prvá testovacia vzorka je upútavka k celovečernému filmu James Bond Casino Royale. Táto ukážka obsahuje veľké množstvo rýchlych strihov. Veľké množstvo strihov dobre poslôži na otestovanie úspešnosti určovania miest hraníc záberov. Druhá testovacia vzorka je taktiež upútavka k rovnakému filmu ale vznikla komprimovaním videa iným video komprimačným algoritmom a zároveň rozmery snímok sú odlišné. Táto ukážka poslúži na otestovania invariantnosti priebehu porovnávania vzhľadom na použitý video komprimačný algoritmus a velkosť snímkov. Tretia testovacia vzorka vznikla prevzorkovaním prvej testovacej vzorky na nižší počet snímkov za sekundu. Implementovaný algoritmus porovnávania by mal byť invariantný voči počtu snímkov za sekundu porovnávaných videosúborov. Štvrtá testovacia vzorka vznikla prekomprimovaním prvej testovacej vzorky pomocou iného video komprimačného algoritmu. Piata testovacia vzorka vznikla vynechaním asi 38 sekúnd z prvej testovacej vzorky. Šiesta vzorka je vzorka z inej upútavky k celovečernému filmu a poslúži na otestovanie funkčnosti porovnávania dvoch rôznych videosúborov. Vo vybraných videosúboroch boli taktiež ručne anotované pozície hraníc záberov.

#	Nazov	Komprimácia	Dĺžka	Snímkov za sekundu	Rozmery
1	1_crt.flv	h264	02:34.14	30	540x360
2	2_crt_vga.avi	mpeg4	02:34:10	30	640x480
3	3_crt_25fps.flv	mpeg4	02:34.14	25	540x360
4	4_crt_30fps.flv	mpeg4	02:34.14	30	540x360
5	5_crt_30fps_vs.flv	mpeg4	01:53.68	30	540x360
6	6_qst.flv	h264	01:58.08	30	640x360

Tabulka 6.1: Parametre vybraných testovacích vzoriek

6.3 Voľba testovacích parametrov

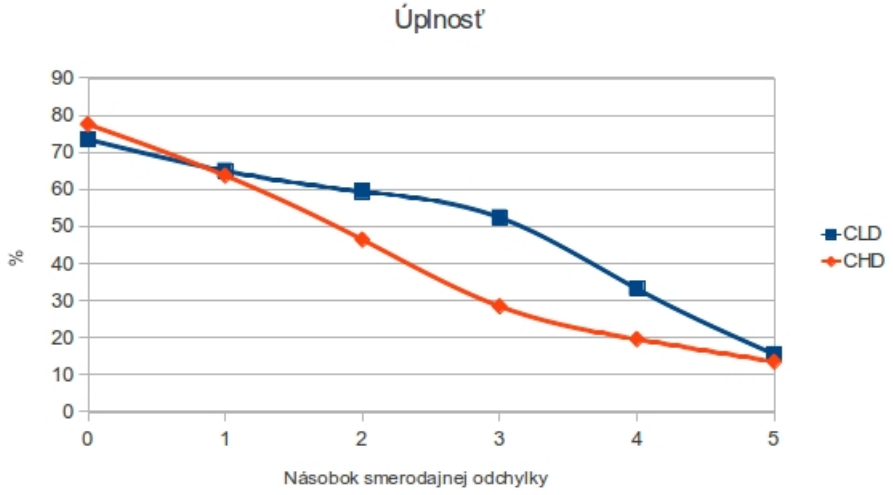
Je zrejmé, že aplikácie určené na porovnávanie dát sú závislé od toho ako dokážu detekovať jednotlivé zmeny v porovnávaných dátach. Presnosť implementovanej aplikácie je taktiež závislé na presnosti určenia porovnávaných prvkov. Implementovaná aplikácia obsahuje sadu parametrov ktorými je možné upravovať priebeh porovnávania. Ukážka 5.2 zobrazuje kompletnú sadu dostupných parametrov. Táto časť kapitoly je venovaná testom jednotlivých parametrov, ich presnosti a vplyvu na celkový výsledok porovnávania.

Výber použitých rysov spolu s výberom typu vzdialenostnej metriky patrí medzi parametre ktoré razantne ovplyvňujú priebeh porovnávania a celkový výstup aplikácie. Ďalšími dôležitými parametrami sú parametre ovplyvnujúce algoritmus určovania hraníc záberov. V rámci testovania boli otestovaná úspešnosť jednotlivých rysov a vzdialenostných metrík pri určovaní hraníc záberov. Merané boli dve hodnoty presnosť (6.1) a úplnosť (6.2). Testovanie prebiehalo pre hodnoty hraníc strihu pri 0 až 5 násobku smerodajnej odchylky. Hranica strihu bola vypočítana podľa rovnice (5.1). V rámci testov boli použité dva typy šírky plávajúceho okna – 10 a 15 snímkov. Hranica bola označená ako uspešne určená pokiaľ rozdiel medzi pozíciou skutočného strihu a vypočítaného miesta strihu bol menší ako polovica šírky použitého okna.

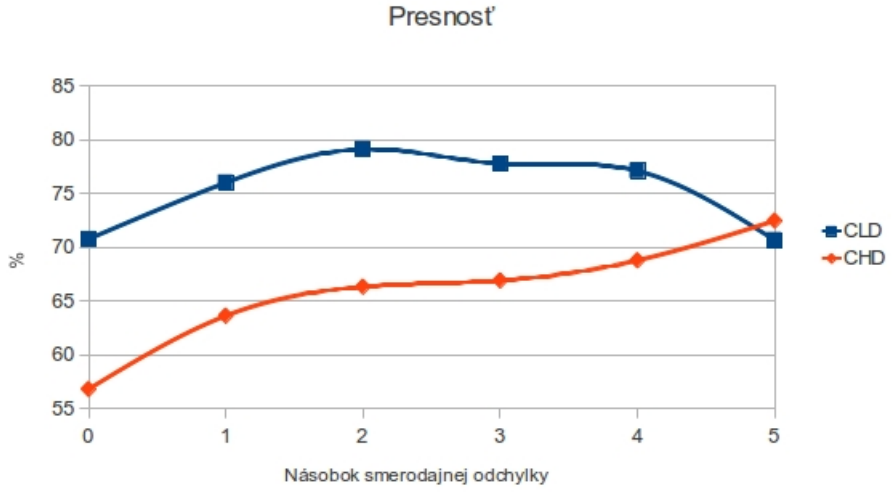
$$\text{Presnosť} = \frac{\text{Počet správne určených strihov}}{\text{Počet nájdených strihov}} \tag{6.1}$$

$$\text{Úplnosť} = \frac{\text{Počet správne určených strihov}}{\text{Počet ručne anotovaných strihov}} \tag{6.2}$$

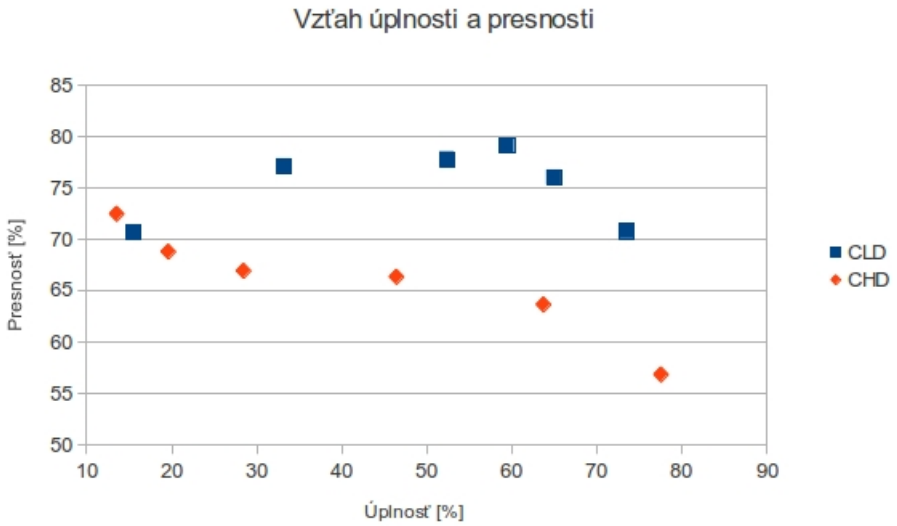
Získané výsledky boli spriemerované za všetky použité testované vzorky. Graf na obrázku 6.1 zobrazuje priebeh úplnosti a graf na obrázku 6.2 priebeh presnosti pre jednotlivé úrovne hraníc strihov. Hodnoty CLD predstavujú rysy *Color layout* a CLH rysy *Color histogram*. Z uvedných grafov je vidno, že rysy *Color layout* dosahujú v priemere lepších výsledkov. Dosiahnutie lepších výsledkov je spojené s prepracovanejším zisťovaním rysov. Ako bolo spomenuté v predchádzajúcich kapitolách rysy *Color layout* okrem farby berú v úvahu taktiež jej rozmiestnenie. Pre úplnosť je na obrázku 6.3 zobrazený graf zobrazujúci získané hodnoty v závislosti presnosti a úplnosti.



Obrázek 6.1: Priebeh úplnosti v priemere za všetky testované vzorky.

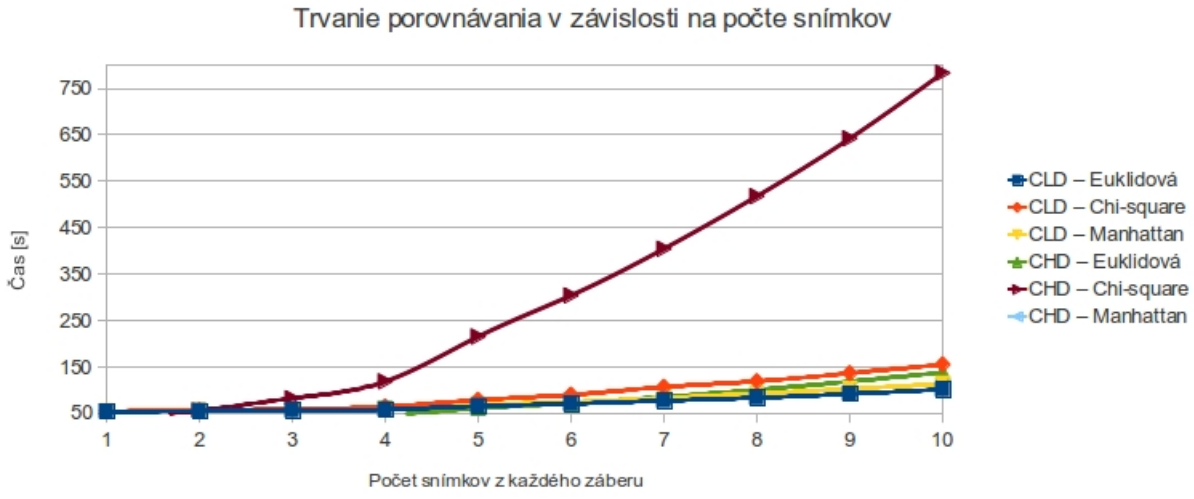


Obrázek 6.2: Priebeh presnosti v priemere za všetky testované vzorky.



Obrázek 6.3: Vzťah úplnosti a presnosti.

Jednotlivé rysy a vzdialenostné metriky sa líšia aj dobou, ktorá je potrebná na porovnanie dvoch videosúborov. Graf na obrázku 6.4 zobrazuje čas potrebný na porovnanie dvoch rôznych videosúborov o dĺžke dve a pol minúty vzhľadom na počet porovnávaných snímkov z každého záberu. Je zrejmé, že okrem chi-square metriky majú jednotlivé typy rysov a metrík majú podobnú časovú náročnosť. Jedine použitie chi-square metriky s *Color histogram* rysmi má pri vačšom počte snímkom omnoho vyššiu časovú zložitosť. Tento rozdiel je spôsobený veľkým množstvom matematických operácií s číslami s plávajúcou desatinnou čiarkou a aj tým, že počet hodnôt ktoré vstupujú do výpočtov je pri rysoch *Color histogram* približne dva krát vyšší ako pri rysoch *Color layout*.



Obrázek 6.4: Porovnávanie doby porovnávania dvoch videosúborov.

Ďalší parameter ktorý výrazne ovplyvnuje priebeh algoritmu je citlivosť. Citlivosť ako už bolo spomentué v kapitole 5 je hodnota vzdialenosti v rámci ktorej sa ešte dvojica snímkov považuje za zhodnú. Nakoľko typ použitých rysov a typ vzdialenostnej metriky

môže razantne ovplyvniť priebežné hodnoty a preto je to nutné brať v úvahu pri určovaní citlivosti. Na otestovanie citlivosti bola pripravená nasledujúca sada obrázkov:

Posuny túto sadu tvoria obrázky ktoré vo videu nasledujú za sebou a je medzi nimi malý rozdiel, Rozdiel je tvorený hlavne posunom objektov v snímku a to z dôsledku pohybu. Cieľom je otestovať citlivosť na mierne zmeny.

Väčšie túto sadu tvoria snímky ktoré sa líšia rozmermi. Cieľom je otestovať citlivosť na tento typ zmien.

Kompresia túto sadu tvoria rovnaké snímky ktoré boli získané s rovnakého videa komprimovaného použitím inej video kompresnej metódy. Cieľom je otestovať citlivosť na použitú video kompresiu.

Rôzne túto sadu tvoria obrázky ktoré sú navzájom rôzne. Cieľom je otestovať a určiť hraničnú hodnotu podobnosti.

Každá sada okrem sady rôzne obsahuje po 5 rôznych dvojíc snímkov. Kompletné namerané hodnoty sú k dispozícii v prílohe C. Priemerné hodnoty boli vypočítané ako priemer vypočítaných vzdialeností zo všetkých testovacích prípadov v rámci danej testovacej sady. Vypočítané priemerné hodnoty pre jednotlivé sady sú zobrazené v tabuľke 6.2.

	Color layout			Color histogram		
Sada	Euklidová	Chi-square	Manhattan	Euklidová	Chi-square	Manhattan
Posuny	7,82	0,29	4,82	2,03	2,54	3,74
Väčšie	11,97	0,85	5,37	21,59	34,61	7,50
Kompresia	20,94	2,49	7,39	27,94	38,52	6,33
Rôzne	157,76	124,79	44,08	10,3	38,72	7,86

Tabulka 6.2: Priemerné hodnoty vzdialenosti pre jednotlivé testovacie sady

Z hodnôt uvedených v tabuľke sa znova potvrdzuje, že rysy *Color histogram* dosahujú horšie výsledky. Zrejmé to je predovšetkým z malých rozdielov vzdialeností medzi rozdielnými snímkami a rovnakými snímkami rozdielne komprimovaného videa. Takto úzka hranica môže viesť v praxi k nesprávnym výsledkom. Naopak rysy *Color layout* potvrdili svoju výkonnosť a je možné vidieť markantný rozdiel medzi vzdialenosťami podobných snímkov a snímkov rozdielných kde sú vzdialenosti niekoľkonásobne vyššie.

6.4 Dosiahnuté výsledky

Predchádzajúca podkapitola sa zaoberala testami dostupných parametrov. Vychádzajúc z výsledkov získaných v predchádzajúcej kapitole sa táto kapitola venuje testovaniu kompletnej funkčnosti aplikácie.

Vzhľadom na merania z predošlej kapitoly bolo rozhodnuté, že testovanie jednotlivých testovacích vzoriek bude prebiehať pre každý typ implementovaných rysov a každú implementovanú vzdialenostnú metriku. Citlivosť v rámci jednotlivých testov bude závisieť od typu použitých rysov a vzdialenostnej metriky viď tabuľka 6.2 a bude v prípade rysov *Color layout* rovná polovičnej vzdialenosti získanej pre sadu rôznych snímkov. Pri použití rysov *Color histogram* bude vzhľadom na prekrývnie vypočítaných vzdialeností citlivosť rovná hodnote vzdialenosti získanej pre sadu rôznych snímkov. Parametre ktoré sú pre každý testovací prípad konštantné sú:

Hranica strihu 1 násobok smerodajnej odchylky

Šírka okna 15 snímkov

Počet snímkov na záber 5 snímkov

Testovavanie vzoriek s rozdielnym počtom snímkov za sekundu K tomuto testu boli použité testovacie vzorky číslo 3 a 4. Tieto vzorky sa líšia len v počte snímkov za sekundu. Nakoľko zmena počtu snímkov za sekundu nemení obsah jednotlivých snímkov očakávaný výstup je, že aplikácia nedetekuje žiadnu zmenu.

Avšak získané výstupy, ktoré sú dostupné v prílohe D.1 ukazujú, že aplikácia nejaké zmeny detekovala. V prípade použitia rysov *Color layout* a Euklidovej metriky aplikácia detekovala veľke mnnožstvo krátkych zmenených úsekov viď D.1. Pri hlbšej analýze medzivýsledkov vyšlo najavo, že hranice ktoré aplikácia detekovala neboli v oboch testovacích ukážkach a dokonca nesúhlasil ani počet detekovaných strihov. Pokiaľ sa pozrieme na výsledky ďalších meraní tak pri použití chi-square metriky aplikácia detekovala štyri zmeny a spolu nesprávne detekovala 7 sekúnd viď ukážka D.2. Príčinou nesprávneho výstupu boli opäť rozdiely v detekovaných hraniciach záberov. Výstup testu Manhattan metriky obsahuje jednu detekovanú zmenu viď D.3 a aj v tomto prípade nesúhlasili detekované hranice záberov a preto aplikácia určila nesprávne približne 20 sekúnd obsahu testovacej vzorky.

Pokiaľ sa pozrieme na výstupy rysov *Color histogram* na ukážkach D.4, D.5 a D.6 je zrejmé, že aplikácie taktiež nesprávne detekovala zmeny a aj tu boli dôvodom nesúhlasne detekované hranice záberov.

Zo získaných výsledkov je zrejmé, že aplikácia má problém správne detekovať zmeny v prípade, že porovnávané videosúbory majú odlišný počet snímkov za sekundu. Tabuľka 6.3 zobrazuje súhrny prehľad výsledkov.

Typ rysov	Metrika	Citlivosť	Výstup
Color layout	Euklidova	75	Chybne
	Chi-square	60	Chybne
	Manhattan	22	Chybne
Color histogram	Euklidova	10	Chybne
	Chi-square	38	Chybne
	Manhattan	8	Chybne

Tabulka 6.3: Výsledky testovania vzoriek s rozdielnym počtom snímkov za sekundu.

Testovavanie vzoriek kódovaných rôznym video komprimačným algoritmom V rámci tohto testu bola testovaná citlivosť porovnávania na videá komprimované rôznym video komprimačným algoritmom. Test prebiehal na vzorke číslo 1, ktorá je kódovaná algoritmom h264 a na vzorke 4, ktorá je kódovaná algoritmom mpeg-4. Rovnako ako v predchádzajúcom teste očakávaný výstup je, že testované vzorky sú zhodné.

Pri pohľade na výstupy testu je zrejmé, že v prípade použitia rysov *Color layout* a metrík Euklidovej a chi-square aplikácia správne nedetekovala žiadnu zmenu a tým potvrdila, že vzorky sú zhodné. Výsledok testu z Manhattan metrikou ukazuje, že aplikácia nesprávne detekovala 7 sekúnd viď ukážku D.9. Po bližšej analýze sa potvrdilo, že detekované zábery v jednotlivých videách neboli zhodné čo malo za následok nesprávne detekovanie zmeny

nakoľko daných detekovaných 7 sekúnd predstavuje záber ktorý tvorí rozdiel medzi detekovanými zábermi v jednotlivých vzorkách.

Pri pohľade na výstupy testov rysov *Color histogram* je zrejmé že aplikácia opäť nesprávne detekovala veľké množstvo zmien. Výstupy sú na ukážkach D.7, D.8 a D.10. Sumárny prehľad výsledkov je v tabuľke 6.4.

Typ rysov	Metrika	Citlivosť	Výstup
Color layout	Euklidova	75	Správne
	Chi-square	60	Správne
	Manhattan	22	Chybne
Color histogram	Euklidova	10	Chybne
	Chi-square	38	Chybne
	Manhattan	8	Chybne

Tabulka 6.4: Výsledky testovania vzoriek vytvorených rozdielným video komprimačným algoritmom.

Testovavanie vzoriek s rozdielnými rozmermi Na otestovanie citlivosti porovnávania na rozdiely v rozmeroch porovnávaných videí boli zvolené testovacie vzorky číslo 1 a 2. Tieto vzorky sa okrem rozmerov líšia aj v použitom video komprimačnom algoritme. Aj v tomto prípade je očakávaným výstupom to, že aplikácia nedetekuje žiadnu zmenu. Tabuľka 6.5 obsahuje prehľad získaných výsledkov.

Pri pohľade na získané výsledky z testov rysov *Color layout* je zrejmé, že aplikácia v prípade použitia Euklidovej metriky nesprávne detekovala úsek o dĺžke približne 1 sekunda viď ukážka D.11. V prípade chi-square metriky, ukážka D.12, aplikácia nesprávne detekovala dva úseky spolu o dĺžke 2 sekundy. Výstup testu z Manhattan metrikou na ukážke D.13 ukazuje, že aplikácia nesprávne detekovala dva úseky spolu o dĺžke tiež 2 sekundy. Pri bližšej analýze sa zistilo, že v prípade Euklidovej metriky aplikácia nedetekovala jeden a v prípade chi-square a Manhattan metriky dva zábery čo zapríčinilo nesprávny výstup.

Výstup testov *Color histogram* ukazuje, že aplikácia pri použití ktorej koľvek metriky detekovala veľké množstvo neplatných zmien. Príčinou je opäť nesúhlasný počet detekovaných záberov. Výstupy sú dostupné na ukážkach D.14, D.15 a D.16.

Typ rysov	Metrika	Citlivosť	Výstup
Color layout	Euklidova	75	Chybne
	Chi-square	60	Chybne
	Manhattan	22	Chybne
Color histogram	Euklidova	10	Chybne
	Chi-square	38	Chybne
	Manhattan	8	Chybne

Tabulka 6.5: Výsledky testovania vzoriek s rozdielnými rozmermi.

Testovavanie vzoriek s vynechanými zábermi V tomto teste boli testované vzorky číslo 4 a 5. Tieto vzorky sa líšia v tom, že zo vzorky číslo 5 bolo vystrihnutých približne 38 sekúnd videa v čase od 38 sekundy. Očakávaný výstup tohto testu teda je, že aplikácia korektne detekuje dĺžku a miesto vystrihnutej časti.

Typ rysov	Metrika	Citlivosť	Výstup
Color layout	Euklidova	75	Správne
	Chi-square	60	Chybne
	Manhattan	22	Chybne
Color histogram	Euklidova	10	Chybne
	Chi-square	38	Chybne
	Manhattan	8	Chybne

Tabulka 6.6: Výsledky testovania vzoriek s vynechanými zábermi.

Pri pohľade na výstupy testov v prílohe D.4 je zrejmé, že jediný správny výstup dosiahli rysy *Color layout* spolu s Euklidovou metrikou viď ukážka D.17. Z ukážky je názorne vidieť kedy vystrihnutá časť začína a kde končí. Pri použití chi-square metriky bol výstup, viď D.18 nesprávny nakoľko aplikácia určila síce, že v sa dané testovacie vzorky líšia ale nedetekovala celú vystrihnutú časť a taktiež detekovala zmenu v miestach kde nenastala. Pri použití Manhattan metriky aplikácie správne detekovala vystrihnutú časť, ale nie celú a na nesprávnom miesta viď ukážka D.19.

Výstup testov rysov *Color histogram* z akovkoľvek metrikou bol opäť nesprávny a aplikácia detekovala zmeny aj tam kde nenastali. Pre jednotlivé výstupy viď D.20, D.21 a D.22.

Typ rysov	Metrika	Citlivosť	Výstup
Color layout	Euklidova	75	Chybne
	Chi-square	60	Chybne
	Manhattan	22	Chybne
Color histogram	Euklidova	10	Chybne
	Chi-square	38	Chybne
	Manhattan	8	Chybne

Tabulka 6.7: Výsledky testovania rôznych vzoriek.

Testovavanie navzájom rôznych vzoriek V teste boli použité vzorky číslo 1 a 6. Tieto testovacie vzorky sú odseba odlišné nakoľko ide o upútavky k rozličným celovečerným filmom. Očakávaný výstup je označenie vzoriek ako úplne odlišných.

Zhodnotenie týchto výsledkov je náročné. Ani v jednom testovacom prípade neboli dosiahnuté spravne výsledky. Avšak v každom testovacom prípade aplikácia detekovala, že sa testovacie ukážky líšia avšak výstup bol nebol správny. Nepresnosť bola spôsobená nízkou citlivosťou. Po zvýšení citlivosti sa výsledky zlepšili. Výstupy sú dostupné v prílohe D.5.

6.5 Zhodnotenie testov

Výsledky testov ukázali, že rysy *Color layout* podľa očakávania dosahujú vyššiu presnosť ako rysy *Color histogram*. Zároveň sa ukázalo, že implementovaný algoritmus detekcie hraníc záberov je citlivý na rôzne transformácie vstupných dát. Predovšetkým pri videách s rôznym počtom snímok za sekundu dochádzalo k veľkým rozdielom medzi počtom detekovaných záberov v jednotlivých videách. Týmto zmenám je ale náročné predchádzať nakoľko pri úpravách videa môže dochádzať k výrazným zmenám. Testy taktiež preukázali dôležitosť

správnej voľby vstupných parametrov. Hlavne citlivosti, ktorá dosť razantne ovplyvňuje výstup aplikácie.

Ďalším zisteným faktom je, že rysy *Color histogram* sa ukázali ako veľmi nepresné nakoľko dochádzalo k veľkým rozdielom vo výstupoch jednotlivých testovacích prípadov ako aj k rozdielom v počtoch detekovaných záberov. Vzhľadom ale na spôsob získavania týchto rysov a ich nízku schopnosť rolíšiť rôzne snímky s rovnakým farebným zložením bol tento výsledok očakávaný.

Kapitola 7

Záver

Táto práca bola zameraná na návrh a implementáciu aplikácie diff pre video súbory ktorá porovnáva videá na základe obsahu. Úvod práce bol venovaný úvodu do problematiky získavania informácií o obsahu z obrázkov a videa a štandardu MPEG-7 Visual pre popis nízkoúrovňových rysov. Práca sa taktiež zaoberala tradičnými algoritmami na porovnávanie textových súborov a rôznym vzdialenostným metrikám.

V rámci práce bola navrhnutá a implementovaná aplikácia na porovnávanie videa na základe obsahu. Implementovaná aplikácia využíva nízkoúrovňové rysy Color layout a Color histogram k určovaniu podobnosti medzi videosúbormi. V rámci priebehu porovnávania sú vo videách detekované hranice záberov. Algoritmus detekcie hraníc záberov je založený na porovnávaní vzdialeností susedných snímkov. Implementovaný algoritmus vychádza z definície strihu ako miesta kde sa obsah susedných snímkov razantne zmení. Samotné porovnávanie je založené na porovnávaní zvolených snímkov z každého detekovaného záberu. Ako algoritmus porovnávania bol implementovaný algoritmus dynamického ohýbania času, ktorý je schopný porovnávať súbory líšiace sa v rýchlosti a čase. V práci boli popísané použité externé knižnice a nástroje. Po ukončení fáze implementácie boli vykonané testy overujúce správnu funkčnosť implementovanej aplikácie.

V rámci testovania boli testované nízkoúrovňové rysy Color histogram a Color layout definované normou MPEG-7. Jednotlivé rysy boli testované pri použití rôznych vzdialenostných metrík. Medzi testované vzdialenostné metriky patrila Euklidová metrika, chi-square metrika a Manhattan metrika. Získané výsledky ukázali, že rysy Color layout dosahujú lepšie výsledky. Testovanie taktiež ukázalo, že implementovaný algoritmus detekcie hraníc záberov je citlivý na zmeny vo vstupných dátach. Avšak aj napriek tomu je možné implementovanú aplikáciu hodnotiť ako úspešnú.

Prípadný ďalší vývoj aplikácie by mohol byť zameraný na zlepšenie prenosti a úplnosti algoritmu na detekciu hraníc záberov ako aj zníženie citlivosti na zmeny vo vstupných dátach. V neposlednom rade je možné aplikáciu rozširovať pomocou implementácie nových nízkoúrovňových rysov, vzdialenostných metrík poprípade algoritmov na výber snímkov. Nemenej dôležité je prípadné celkové zeefektívnenie celej aplikácie za účelom skrátenia doby potrebnej na porovnanie súborov.

Navrhnutá aplikácia je súčasťou väčšieho nástroja s názvom mediaDiff ktorý je vyvýjaný na fakulte informačných technológií Vysokého Učení Technického v Brne v spolupráci s firmou Red Hat. Tento nástroj v súčastnosti ponúka široké spektrum modulov na porovnávanie rôznych typov dát.

Literatura

[1] Blanken, Henk M.: *Multimedia retrieval*. Springer, 2007, ISBN 978-3540728948.

[2] Brothánek, J.: Mediadiff - diff pro statické obrázky. Bakalářská práce, FIT VUT v Brně, Brno, 2011.

[3] Correia, H.: Adaptive Coloring for Syntax Highlighting [online]. `http://doc.qt.nokia.com/qq/qq26adaptivecoloring.html`, [cit. 2012-01-06].

[4] Eakins, J.; Graham, M.: Content-based Image Retrieval. Technická zpráva, JTAP, 1999.

[5] Ghodeswar, S.; Meshram, B.: Content-Based Video Retrieval. In *Proceedings of ISCET 2010*, Advanced Computing, 2010, ISBN 978-81-910304-0-2.

[6] Hampapur, A.; Bolle, R.: Comparison of distance measures for video copy detection. In *Multimedia and Expo, 2001. ICME 2001. IEEE International Conference on*, 2001, s. 737 – 740.

[7] Heckel, P.: A technique for isolating differences between files. *Commun. ACM*, ročník 21, č. 4, Duben 1978: s. 264–268, ISSN 0001-0782, doi:10.1145/359460.359467.
URL http://doi.acm.org/10.1145/359460.359467

[8] Hirschberg, D. S.: Algorithms for the Longest Common Subsequence Problem. *J. ACM*, ročník 24, 1977: s. 664–675, ISSN 0004-5411.

[9] Kingsbury, N.: The MPEG Standard [online]. `http://cnx.org/content/m11144/latest/`, [cit. 2012-05-04].

[10] Komadel, M.: *Multimediální diff - audio dokumenty*. Diplomová práce, FIT VUT v Brně, Brno, 2011.

[11] Le Gall, D.: MPEG: a video compression standard for multimedia applications. *Commun. ACM*, ročník 34, č. 4, Duben 1991: s. 46–58, ISSN 0001-0782, doi:10.1145/103085.103090.
URL http://doi.acm.org/10.1145/103085.103090

[12] Mitchell, J.; Pennebaker, W.; Fogg, C.; aj.: MPEG History. In *MPEG Video Compression Standard*, Springer US, 2002, ISBN 978-0-306-46983-1, s. 383–393, 10.1007/0-306-46983-9_18.
URL http://dx.doi.org/10.1007/0-306-46983-9_18

[13] Amaya S.: History of C++ - C++ information [online]. `http://www.cplusplus.com/info/history/`, [cit. 2012-04-29].

[14] Berkeley University: History of MPEG [online].
http://www2.sims.berkeley.edu/courses/is224/s99/GroupG/report1.html, [cit. 2012-05-03].

[15] Free Software Foundation: Comparing and Merging Files [online].
http://www.gnu.org/software/diffutils/manual/html_mono/diff.html, [cit. 2011-12-23].

[16] Hunt, J.; McIlroy, M.: An Algorithm for Differential File Comparison. Technická zpráva, Bell Laboratories, 1976.

[17] INTERNATIONAL ORGANISATION FOR STANDARDISATION: MPEG-7 Overview (version 10) [online].
http://mpeg.chiariglione.org/standards/mpeg-7/mpeg-7.htm, [cit. 2011-12-31].

[18] Ohm J.; Cieplinski L; Kim H. J.; Krishnamachari S.; Manjunath B. S.: The MPEG-7 Colour Descriptor. *Circuits and Systems for Video Technolog, IEEE Transactions on*, 2001.

[19] Prispievatelia portálu Wikipedia: Longest common subsequence problem - Wikipedia, the free encyclopedia [online].
http://en.wikipedia.org/wiki/Longest_common_subsequence_problem, [cit. 2011-12-26].

[20] Prispievatelia portálu Wikipedia: Hamming Distance - Wikipedia, the free encyclopedia [online]. http://en.wikipedia.org/wiki/Hamming_distance, [cit. 2011-12-30].

[21] Prispievatelia portálu Wikipedia: Video compression picture types - Wikipedia, the free encyclopedia [online].
http://en.wikipedia.org/wiki/Video_compression_picture_types, [cit. 2012-05-03].

[22] Prispievatelia portálu Wikipedia: Color Layout Descriptor - Wikipedia, the free encyclopedia [online]. http://en.wikipedia.org/wiki/Color_layout_descriptor, [cit. 2012-05-05].

[23] Python Software Foundation: History and License - Python v2.7.3 documentation [online]. http://docs.python.org/license.html, [cit. 2012-04-29].

[24] The FFmpeg team: About FFmpeg [online]. http://ffmpeg.org/about.html, [cit. 2012-04-29].

[25] The Kompare Team: Kompare - Different from rest [online].
http://www.caffeinated.me.uk/kompare/, [cit. 2012-01-06].

[26] Umeå University: Computer Graphics and Visualization Course - Lecture 3 [online].
http://www8.cs.umu.se/kurser/TDBC07/HT04/handouts/H0-lecture3.pdf, [cit. 2012-01-02].

[27] Video Group: *Text of ISO/IEC 15938-3/FCD Information Technology – Multimedia Content Description Interface – Part 3 Visual*. INTERNATIONAL ORGANIZATION FOR STANDARDIZATION, 2001.

[28] WinMerge Development Team: WinMerge [online]. http://winmerge.org/, [cit. 2012-01-06].

[29] Rostás, K.: METRICKÉ PRIESTORY, 2010.

[30] Saenz, J.: Jefferson Saenz Blog [online]. http://jeffersonsaenz.blogspot.com/, [cit. 2012-01-06].

[31] Smith, J. R.; Chang, S.-F.: VisualSEEk: a fully automated content-based image query system. In *Proceedings of the fourth ACM international conference on Multimedia*, MULTIMEDIA '96, New York, NY, USA: ACM, 1996, ISBN 0-89791-871-1, s. 87–98.

[32] Tudor, P.: MPEG-2 VIDEO COMPRESSION [online]. http://www.bbc.co.uk/rd/pubs/papers/paper_14/paper_14.shtml, [cit. 2012-05-03].

[33] Vadivel, A.; Majumdar, A. K.; Sural, S.: Performance Comparison of distance metrics in content-based image retrieval applications. In *International Conference on Information Technology*, 1999.

[34] Webopedia: Video - A word definition from the Webopedia computer dictionary [online]. http://www.webopedia.com/TERM/V/video.html, [cit. 2012-01-06].

[35] Yujian, L.; Bo, L.: A Normalized Levenshtein Distance Metric. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, ročník 29, č. 6, 2007: s. 1091 –1095, ISSN 0162-8828.

[36] Zemko, M.: Diff pro různé typy dokumentů. Bakalářská práce, FIT VUT v Brně, Brno, 2009.

[37] Zemko, M.: *Diff pro různé typy dokumentů*. Diplomová práce, FIT VUT v Brně, Brno, 2011.

Příloha A

Návod na použitie a obsah CD

A.1 Návod na použitie

Ku správnemu používaniu aplikácie VideoDiff je nutné mať nainštalovaný Python interprét verzie 2.7 a nainštalovanú knižnicu FFmpeg. Aplikácia VideoDiff bola testovaná s verziou FFmpeg číslo 0.10.3 preto sa odporúča mať nainštalovanú práve túto verziu. Zdrojové kódy knižnice FFmpeg verzie 0.10.3 sú k dispozícií aj na CD.

Aplikácia VideoDiff sa ovláda z príkazového riadka ukážka A.1 zobrazuje prehľad parametrov príkazového riadku. Význam jednotlivých parametrov je nasledovný:

features typ použitých rysov. Podporované rysy:

1. Color layout
2. Color histogram

kfextractor typ algoritmu na výber snímkov zo záberov.

metrics typ použitej vzdialenostnej metriky

1. Euklidová metrika
2. Chi-square metrika
3. Manhattan metrika

kfpershot počet snímkov vybratých z každého záberu.

windowsize šírka plávajúceho okna pri zisťovaní hraníc záberov. Udáva sa počtom snímkov.

cutthreshold parameter určujúci násobok smerodajnej odchylky vzdialeností susedných snímkov použitý pri určovaní hranice strihu.

similaritythreshold citlivosť porovnávania. Udáva sa vzdialenosťou, čím vačšie číslo tým je citlivosť menšia.

ratio určuje pomer v ktorom porovnávať získané snímky. Pokiaľ je pomer 1.00 tak sa snímky porovnávajú v pôvodnej veľkosti. Pokiaľ napríklad 0.5 tak sú snímky zmenšené na polovicu pôvodnej veľkosti.

positions prepínač určujúci či na výstupe zobrazovať pozície snímkov namiesto času.

Ukázka A.1: Parametre príkazového riadku

```
videodiff.py [-h] [--features {1,2}] [--kfextractor {1}]
                  [--metrics {1,2,3}] [--kfpershot KFPERSHOT]
                  [--windowsize WINDOWSIZE] [--cutthreshold CUTTHRESHOLD]
                  [--similaritythreshold SIMILARITYTHRESHOLD]
                  [--ratio RATIO] [--positions]
                  file1 file2
```

A.2 Obsah CD

Na priloženom CD sú dostupné zdrojové kódy vytvorenej aplikácie s potrebnými knižnicami, text práce, testovacie vzorky a text práce vo formáte PDF. V koreňovom adresári je súbor README, ktorý obsahuje kompletné informácie o obsahu CD.

V koreňovom adresári je taktiež skript control-panel.sh ktorý slúži na spustenie inštalácie a testov. Skript control-panel.sh je možné ovládať troma prepínačmi:

–**install-ffmpeg** pokúsi sa nainštalovať knižnicu FFmpeg verzie 0.10.3 ktorá je dostupná na CD. Počas inštalácia môže vyžadovať administrátorské práva.

–**install-videodiff** pokúsi sa nainštalovať knižnicu libvideodiff potrebnú k spusteniu aplikácie VideoDiff. Počas inštalácie môže vyžadovať administrátorské práva.

–**run-test** spustí zvolený test

Príloha B

Výsledky testov detekcie hraníc záberov

	Color layout			Color histogram		
	Strihy: 137	Priem.: 8,30	Odch.:21,58	Strihy: 137	Priem.: 2,06	Odch.: 6,95
	Nájdené	Úplnosť	Presnosť	Nájdené	Úplnosť	Presnosť
0	172	84,17%	67,76 %	175	86,86%	68,00 %
1	139	79,85 %	79,85 %	95	52,55%	75,78 %
2	128	74,82%	81,25 %	67	36,49%	74,62 %
3	113	65,69 %	79,65 %	58	30,65%	72,41 %
4	90	51,09%	77,78 %	43	23,35 %	74,41 %
5	52	28,77%	76,92 %	39	20,43%	71,79 %

Tabulka B.1: Výsledky meraní určenia pozícií strihov v ukážke číslo 1 pri šírke okna 10 snímkov použitím Euklidovej metriky.

	Color layout			Color histogram		
	Strihy: 137	Priem.: 8,30	Odch.: 21,58	Strihy: 137	Priem.: 2,06	Odch.: 6,95
	Nájdené	Úplnosť	Presnosť	Nájdené	Úplnosť	Presnosť
0	139	84,17 %	84,18 %	141	83,94 %	81,56 %
1	122	80,29 %	90,16 %	81	50,36%	85,18 %
2	113	72,27 %	88,03 %	56	34,30%	83,92 %
3	105	68,61%	89,52 %	48	29,19%	83,33 %
4	83	54,01%	89,15 %	39	24,08 %	84,61 %
5	49	31,38%	87,75 %	35	21,16 %	82,85 %

Tabulka B.2: Výsledky meraní určenia pozícií strihov v ukážke číslo 1 pri šírke okna 15 snímkov použitím Euklidovej metriky.

	Color layout			Color histogram		
	Strihy: 137	Priem.: 2,60	Odch.: 16,29	Strihy: 137	Priem.: 2,52	Odch.: 10,75
	Nájdené	Úplnosť	Presnosť	Nájdené	Úplnosť	Presnosť
0	143	81,75 %	78,32 %	164	88,32 %	73,78 %
1	123	71,52 %	79,67 %	114	64,23 %	77,19 %
2	96	54,74%	78,12 %	84	48,17 %	78,57 %
3	71	39,41%	76,05 %	54	29,92%	75,92 %
4	45	24,08 %	73,33 %	45	24,08%	73,33 %
5	26	11,67%	61,53 %	38	20,43%	73,68 %

Tabulka B.3: Výsledky meraní určenia pozícií strihov v ukážke číslo 1 pri šírke okna 10 snímkov použitím chi-square metriky.

	Color layout			Color histogram		
	Strihy: 137	Priem.: 2,60	Odch.: 16,29	Strihy: 137	Priem.: 2,52	Odch.: 10,75
	Nájdené	Úplnosť	Presnosť	Nájdené	Úplnosť	Presnosť
0	124	81,01%	89,51 %	138	86,13 %	85,50 %
1	109	71,52 %	89,90 %	98	62,77 %	87,75 %
2	88	57,66 %	89,77 %	76	48,90%	88,15 %
3	67	42,92 %	89,56 %	49	31,38 %	87,75 %
4	43	27,00 %	86,04 %	41	25,54 %	85,36 %
5	24	13,13%	75,00 %	34	20,43 %	82,35 %

Tabulka B.4: Výsledky meraní určenia pozícií strihov v ukážke číslo 1 pri šírke okna 15 snímkov použitím chi-square metriky.

	Color layout			Color histogram		
	Strihy: 137	Priem.: 3,25	Odch.: 3,76	Strihy: 137	Priem.: 2,09	Odch.: 1,73
	Nájdené	Úplnosť	Presnosť	Nájdené	Úplnosť	Presnosť
0	195	86,12%	60,51 %	219	89,05 %	55,70 %
1	148	83,93%	77,70 %	162	84,67%	71,60 %
2	135	79,56%	80,74 %	126	70,80%	76,98 %
3	124	73,72%	81,45 %	72	37,95%	72,22 %
4	63	36,48%	79,35 %	40	21,89%	75,00 %
5	13	5,83%	61,53 %	9	4,37 %	66,66 %

Tabulka B.5: Výsledky meraní určenia pozícií strihov v ukážke číslo 1 pri šírke okna 10 snímkov použitím Manhattan metriky.

	Color layout			Color histogram		
	Strihy: 137	Priem.: 3,25	Odch.: 3,76	Strihy: 137	Priem.: 2,09	Odch.: 1,73
	Nájdené	Úplnosť	Presnosť	Nájdené	Úplnosť	Presnosť
0	153	85,40 %	76,47 %	159	83,94 %	72,32 %
1	130	83,20 %	87,69 %	132	81,02%	84,09 %
2	117	77,36 %	90,59 %	111	70,07%	86,48 %
3	109	72,25 %	90,82 %	61	37,22%	83,60 %
4	59	38,69 %	89,84 %	36	22,62%	86,11 %
5	13	7,29 %	76,92 %	9	5,10 %	77,77 %

Tabulka B.6: Výsledky meraní určenia pozícií strihov v ukážke číslo 1 pri šírke okna 15 snímkov použitím Manhattan metriky.

	Color layout			Color histogram		
	Strihy: 79	Priem.: 13,00	Odch.:29,86	Strihy: 79	Priem.: 5,01	Odch.: 13,59
	Nájdené	Úplnosť	Presnosť	Nájdené	Úplnosť	Presnosť
0	84	68,34 %	64,28 %	102	78,48 %	60,78 %
1	58	62,02 %	84,48 %	57	45,56 %	63,15 %
2	53	56,95 %	84,90 %	29	25,31 %	68,96 %
3	45	50,62 %	88,88 %	25	25,21 %	80,00 %
4	38	44,03 %	92,10 %	19	22,78 %	94,73 %
5	27	30,37 %	88,88 %	19	22,78 %	94,73 %

Tabulka B.7: Výsledky meraní určenia pozícií strihov v ukážke číslo 6 pri šírke okna 10 snímkov použitím Euklidovej metriky.

	Color layout			Color histogram		
	Strihy: 79	Priem.: 13,00	Odch.: 29,86	Strihy: 79	Priem.: 5,01	Odch.: 13,59
	Nájdené	Úplnosť	Presnosť	Nájdené	Úplnosť	Presnosť
0	56	55,69 %	78,57 %	67	58,22 %	68,65 %
1	37	43,03 %	91,89 %	48	48,10 %	79,16 %
2	35	40,50 %	91,42 %	27	30,37 %	88,88 %
3	30	37,97 %	100,00 %	23	27,84 %	95,65 %
4	29	36,70 %	100,00 %	17	21,51 %	100,00 %
5	23	29,11 %	100,00 %	17	21,51 %	100,00 %

Tabulka B.8: Výsledky meraní určenia pozícií strihov v ukážke číslo 6 pri šírke okna 15 snímkov použitím Euklidovej metriky.

	Color layout			Color histogram		
	Strihy: 79	Priem.: 4,48	Odch.: 20,94	Strihy: 79	Priem.: 8.03	Odch.: 24.63
	Nájdené	Úplnosť	Presnosť	Nájdené	Úplnosť	Presnosť
0	68	68,35 %	79,41 %	92	74,68 %	64,13 %
1	52	55,69 %	84,61 %	58	53,16 %	72,41 %
2	47	53,16 %	89,36 %	37	35,44 %	75,67 %
3	44	50,62%	90,90 %	23	24,05 %	82,60 %
4	34	39,23 %	91,17 %	21	24,05 %	90,47 %
5	28	32,66 %	92,15 %	20	22,78 %	90,00 %

Tabulka B.9: Výsledky meraní určenia pozícií strihov v ukážke číslo 6 pri šírke okna 10 snímkov použitím chi-square metriky.

	Color layout			Color histogram		
	Strihy: 79	Priem.: 4,48	Odch.: 20,94	Strihy: 79	Priem.: 8.03	Odch.: 24.63
	Nájdené	Úplnosť	Presnosť	Nájdené	Úplnosť	Presnosť
0	43	46,83 %	86,04 %	67	64,55 %	76,11 %
1	35	40,50 %	91,42 %	45	49,36 %	86,66 %
2	33	40,50 %	96,96 %	32	37,97 %	93,75 %
3	32	40,50 %	100,00 %	21	25,31 %	95,23 %
4	25	31,64 %	100,00 %	19	24,05 %	10,00 %
5	23	29,11%	100,00 %	18	22,78 %	100,00 %

Tabulka B.10: Výsledky meraní určenia pozícií strihov v ukážke číslo 6 pri šírke okna 15 snímkov použitím chi-square metriky.

	Color layout			Color histogram		
	Strihy: 79	Priem.:3,95	Odch.: 4,83	Strihy: 79	Priem.: 2,76	Odch.: 2,70
	Nájdené	Úplnosť	Presnosť	Nájdené	Úplnosť	Presnosť
0	100	68,35 %	54,00 %	74	64,55 %	68,91 %
1	67	67,08 %	79,10 %	62	62,02 %	79,03 %
2	54	58,22 %	85,18 %	39	45,56 %	92,30 %
3	43	49,03 %	90,09 %	21	25,31 %	95,23 %
4	15	18,98 %	100,00 %	8	10,12 %	100,00 %
5	3	3,79 %	100,00 %	1	1,26 %	100,00 %

Tabulka B.11: Výsledky meraní určenia pozícií strihov v ukážke číslo 6 pri šírke okna 10 snímkov použitím Manhattan metriky.

	Color layout			Color histogram		
	Strihy: 79	Priem.: 3,95	Odch.: 4,83	Strihy: 79	Priem.: 2,76	Odch.:2,70
	Nájdené	Úplnosť	Presnosť	Nájdené	Úplnosť	Presnosť
0	69	60,75 %	69,56 %	71	56,87 %	63,28 %
1	43	49,36 %	90,69 %	51	46,85 %	72,54 %
2	36	43,03 %	94,44 %	36	36,70 %	80,55 %
3	31	39,24 %	100,00 %	20	22,78 %	90,00 %
4	15	18,98 %	100,00 %	17	20,25 %	94,11 %
5	3	3,79 %	100,00 %	14	17,72 %	100,00 %

Tabulka B.12: Výsledky meraní určenia pozícií strihov v ukážke číslo 6 pri šírke okna 15 snímkov použitím Manhattan metriky.

	Color layout			Color histogram		
	Strihy: 137	Priem.: 9,55	Odch.: 23,03	Strihy: 137	Priem.: 3,30	Odch.: 8,40
	Nájdené	Úplnosť	Presnosť	Nájdené	Úplnosť	Presnosť
0	161	59,85 %	50,93 %	188	62,04 %	45,21 %
1	135	54,74 %	55,55 %	135	48,90 %	49,62 %
2	120	48,90 %	55,83 %	60	19,70 %	45,00 %
3	104	40,87 %	53,84 %	47	15,32 %	44,68 %
4	73	29,19 %	54,79 %	38	10,21 %	36,84 %
5	31	11,67 %	51,69 %	34	8,75 %	35,29 %

Tabulka B.13: Výsledky meraní určenia pozícií strihov v ukážke číslo 3 pri šírke okna 10 snímkov použitím Euklidovej metriky.

	Color layout			Color histogram		
	Strihy: 137	Priem.: 9,55	Odch.: 23,03	Strihy: 137	Priem.: 3,30	Odch.: 8,40
	Nájdené	Úplnosť	Presnosť	Nájdené	Úplnosť	Presnosť
0	123	66,42 %	73,98 %	147	72.26 %	67.34 %
1	108	64,23 %	81,48 %	115	61.31 %	73,21 %
2	99	58,39 %	80,80 %	53	28.46 %	73.04 %
3	87	51,09 %	80,45 %	43	24.08 %	76.74 %
4	61	37,22 %	83,60 %	37	19.70 %	72.97 %
5	29	15,32 %	72,41 %	33	16,78 %	69.69 %

Tabulka B.14: Výsledky meraní určenia pozícií strihov v ukážke číslo 3 pri šírke okna 15 snímkov použitím Euklidovej metriky.

	Color layout			Color histogram		
	Strihy: 137	Priem.: 2,98	Odch.: 16,33	Strihy: 137	Priem.:4,61	Odch.: 13,33
	Nájdené	Úplnosť	Presnosť	Nájdené	Úplnosť	Presnosť
0	139	58,39 %	57,55 %	185	64,96 %	48,10 %
1	115	48,17 %	57,39 %	133	50,36 %	51,87 %
2	93	39,41 %	58,06 %	81	29,19 %	49,38 %
3	67	27,00 %	55,22 %	57	18,24 %	43,85 %
4	43	18,97 %	60,46 %	41	11,67 %	39,02 %
5	23	10,21 %	60,86 %	35	10,21 %	40,00 %

Tabulka B.15: Výsledky meraní určenia pozícií strihov v ukážke číslo 3 pri šírke okna 10 snímkov použitím chi-square metriky.

	Color layout			Color histogram		
	Strihy: 137	Priem.: 2,98	Odch.: 16,33	Strihy: 137	Priem.:4,61	Odch.: 13,33
	Nájdené	Úplnosť	Presnosť	Nájdené	Úplnosť	Presnosť
0	112	65,69 %	80,35 %	146	73,72 %	69,17 %
1	95	56,20 %	81,05 %	110	59,85 %	74,54 %
2	77	45,98 %	81,81 %	70	40,14 %	78,57 %
3	58	35,03 %	82,75 %	48	26,27 %	75,00 %
4	40	23,35 %	80,00 %	39	20,43 %	71,79 %
5	21	10,94 %	71,42 %	34	17,51 %	70,58 %

Tabulka B.16: Výsledky meraní určenia pozícií strihov v ukážke číslo 3 pri šírke okna 15 snímkov použitím chi-square metriky.

	Color layout			Color histogram		
	Strihy: 137	Priem.: 3,57	Odch.: 3,99	Strihy: 137	Priem.: 2,57	Odch.: 2,08
	Nájdené	Úplnosť	Presnosť	Nájdené	Úplnosť	Presnosť
0	179	62,77 %	48,04 %	209	65,69 %	43,06 %
1	147	60,58 %	56,46 %	179	64,23 %	49,16 %
2	128	54,01 %	57,81 %	110	43,06 %	53,63 %
3	105	42,33 %	55,23 %	48	13,86 %	39,58 %
4	35	16,78 %	60,52 %	10	2,91 %	40,00 %
5	5	2,91 %	80,00 %	2	0,72 %	50,00 %

Tabulka B.17: Výsledky meraní určenia pozícií strihov v ukážke číslo 3 pri šírke okna 10 snímkov použitím Manhattan metriky.

	Color layout			Color histogram		
	Strihy: 137	Priem.: 3,57	Odch.: 3,99	Strihy: 137	Priem.: 2,57	Odch.: 2,08
	Nájdené	Úplnosť	Presnosť	Nájdené	Úplnosť	Presnosť
0	131	68,61 %	71,75 %	156	73,72 %	64,74 %
1	115	67,88 %	80,86 %	140	72,26 %	70,71 %
2	103	62,04 %	82,52 %	93	53,28 %	78,49 %
3	88	52,55 %	81,81 %	41	21,89 %	73,17 %
4	35	20,43 %	80,00 %	10	5,10 %	70,00 %
5	5	2,91 %	80,00 %	2	0,72 %	50,00 %

Tabulka B.18: Výsledky meraní určenia pozícií strihov v ukážke číslo 3 pri šírke okna 15 snímkov použitím Manhattan metriky.

	Color layout			Color histogram		
	Strihy: 137	Priem.: 7,60	Odch.: 19,91	Strihy: 137	Priem.: 3,27	Odch.: 9,42
	Nájdené	Úplnosť	Presnosť	Nájdené	Úplnosť	Presnosť
0	169	84,67 %	68,63 %	249	85,40 %	46,98 %
1	139	81,75 %	80,57 %	146	52,55 %	49,31 %
2	128	75,91 %	81,25 %	105	39,41 %	51,42 %
3	110	63,50 %	79,09 %	83	29,92 %	49,39 %
4	91	51,82 %	78,02 %	73	24,08 %	45,20 %
5	52	29,19 %	76,92 %	45	16,78 %	51,11 %

Tabulka B.19: Výsledky meraní určenia pozícií strihov v ukážke číslo 2 pri šírke okna 10 snímkov použitím Euklidovej metriky.

	Color layout			Color histogram		
	Strihy: 137	Priem.: 7,60	Odch.: 19,91	Strihy: 137	Priem.: 3,27	Odch.: 9,42
	Nájdené	Úplnosť	Presnosť	Nájdené	Úplnosť	Presnosť
0	138	84,67 %	84,05 %	196	83,94 %	58,67 %
1	122	81,02 %	90,98 %	129	54,74 %	58,13 %
2	113	75,18 %	91,15 %	90	40,87 %	62,22 %
3	100	65,69 %	90,00 %	75	34,30 %	62,66 %
4	83	54,74 %	90,36 %	65	27,00 %	56,92 %
5	49	31,38 %	87,75 %	42	18,97 %	61,90 %

Tabulka B.20: Výsledky meraní určenia pozícií strihov v ukážke číslo 2 pri šírke okna 15 snímkov použitím Euklidovej metriky.

	Color layout			Color histogram		
	Strihy: 137	Priem.: 2,23	Odch.: 14,17	Strihy: 137	Priem.: 5,04	Odch.: 15,04
	Nájdené	Úplnosť	Presnosť	Nájdené	Úplnosť	Presnosť
0	145	82,48 %	77,93 %	252	86,13 %	46,82 %
1	121	70,07 %	79,33 %	206	75,18 %	50,00 %
2	95	54,01 %	77,89 %	129	47,44 %	50,38 %
3	70	38,68 %	75,71 %	95	34,30 %	49,47 %
4	44	23,35 %	72,72 %	67	26,27 %	53,73 %
5	26	11,67 %	61,53 %	42	17,51 %	57,14 %

Tabulka B.21: Výsledky meraní určenia pozícií strihov v ukážke číslo 2 pri šírke okna 10 snímkov použitím chi-square metriky.

	Color layout			Color histogram		
	Strihy: 137	Priem.: 2,23	Odch.: 14,17	Strihy: 137	Priem.: 5,04	Odch.: 15,04
	Nájdené	Úplnosť	Presnosť	Nájdené	Úplnosť	Presnosť
0	126	81,75 %	88,88 %	197	82,48 %	57,36 %
1	107	70,07 %	89,71 %	171	75,18 %	60,23 %
2	87	56,93 %	89,65 %	118	52,55 %	61,01 %
3	66	41,60 %	86,36 %	85	37,95 %	61,17 %
4	42	26,27 %	85,71 %	64	28,46 %	60,93 %
5	24	13,13 %	75,00 %	39	18,97 %	66,66 %

Tabulka B.22: Výsledky meraní určenia pozícií strihov v ukážke číslo 2 pri šírke okna 15 snímkov použitím chi-square metriky.

	Color layout			Color histogram		
	Strihy: 137	Priem.: 3,07	Odch.: 3,59	Strihy: 137	Priem.: 2,43	Odch.: 2,17
	Nájdené	Úplnosť	Presnosť	Nájdené	Úplnosť	Presnosť
0	205	86,13 %	57,56 %	269	87,59 %	44,60 %
1	151	84,67 %	76,82 %	245	87,59 %	48,97 %
2	135	79,56 %	80,74 %	174	69,34 %	54,59 %
3	119	70,07 %	80,67 %	73	29,92 %	56,16 %
4	60	34,30 %	78,33 %	16	5,83 %	50,00 %
5	14	6,56 %	64,28 %	3	1,45 %	66,66 %

Tabulka B.23: Výsledky meraní určenia pozícií strihov v ukážke číslo 2 pri šírke okna 10 snímkov použitím Manhattan metriky.

	Color layout			Color histogram		
	Strihy: 137	Priem.: 3,07	Odch.: 3,59	Strihy: 137	Priem.: 2,43	Odch.: 2,17
	Nájdené	Úplnosť	Presnosť	Nájdené	Úplnosť	Presnosť
0	157	85,40 %	74,52 %	210	85,40 %	55,71 %
1	131	83,21 %	87,02 %	194	85,40 %	60,30 %
2	117	77,37 %	90,59 %	148	70,80 %	65,34 %
3	104	68,61 %	90,38 %	69	32,11 %	63,76 %
4	56	36,49 %	89,28 %	16	8,02 %	68,75 %
5	14	8,82 %	78,57 %	3	2,18 %	100,00 %

Tabulka B.24: Výsledky meraní určenia pozícií strihov v ukážke číslo 2 pri šírke okna 15 snímkov použitím Manhattan metriky.

	Color layout			Color histogram		
	Strihy: 137	Priem.: 8,25	Odch.: 21,47	Strihy: 137	Priem.: 3,89	Odch.: 10,95
	Nájdené	Úplnosť	Presnosť	Nájdené	Úplnosť	Presnosť
0	171	86,13 %	69,00 %	223	88,32 %	54,26 %
1	138	81,75 %	81,15 %	158	63,50 %	55,06 %
2	127	75,91 %	81,88 %	134	54,01 %	55,22 %
3	113	66,42 %	80,53 %	73	31,38 %	58,90 %
4	89	51,09 %	78,65 %	48	23,35 %	66,66 %
5	51	28,46 %	76,47 %	33	17,51 %	72,72 %

Tabulka B.25: Výsledky meraní určenia pozícií strihov v ukážke číslo 4 pri šírke okna 10 snímkov použitím Euklidovej metriky.

	Color layout			Color histogram		
	Strihy: 137	Priem.: 8,25	Odch.: 21,47	Strihy: 137	Priem.: 3,89	Odch.: 10,95
	Nájdené	Úplnosť	Presnosť	Nájdené	Úplnosť	Presnosť
0	141	85,40 %	82,97 %	157	73,72 %	64,33 %
1	122	81,02 %	90,98 %	123	58,39 %	65,04 %
2	113	75,18 %	91,15 %	107	51,82 %	66,35 %
3	104	68,61 %	90,38 %	66	33,57 %	69,69 %
4	81	53,28 %	90,12 %	43	25,54 %	81,39 %
5	48	30,65 %	87,50 %	30	18,97 %	86,66 %

Tabulka B.26: Výsledky meraní určenia pozícií strihov v ukážke číslo 4 pri šírke okna 15 snímkov použitím Euklidovej metriky.

	Color layout			Color histogram		
	Strihy: 137	Priem.: 2,57	Odch.: 16,13	Strihy: 137	Priem.: 5,41	Odch.: 16,49
	Nájdené	Úplnosť	Presnosť	Nájdené	Úplnosť	Presnosť
0	143	82,48 %	79,02 %	227	92,70 %	55,94 %
1	122	71,53 %	80,32 %	176	73,72 %	57,38 %
2	96	55,47 %	79,16 %	120	52,55 %	60,00 %
3	70	38,68 %	75,71 %	74	34,30 %	63,51 %
4	45	24,08 %	75,33 %	46	21,89 %	65,21 %
5	26	11,67 %	61,53 %	34	16,78 %	67,64 %

Tabulka B.27: Výsledky meraní určenia pozícií strihov v ukážke číslo 4 pri šírke okna 10 snímkov použitím chi-square metriky.

	Color layout			Color histogram		
	Strihy: 137	Priem.: 2,57	Odch.: 16,13	Strihy: 137	Priem.: 5,41	Odch.: 16,49
	Nájdené	Úplnosť	Presnosť	Nájdené	Úplnosť	Presnosť
0	125	81,75 %	89,60 %	162	75,18 %	63,58 %
1	109	71,53 %	89,90 %	135	64,23 %	65,18 %
2	88	57,66 %	89,77 %	103	48,90 %	65,04 %
3	66	41,60 %	86,36 %	65	34,30 %	72,30 %
4	43	27,00 %	86,04 %	40	22,62 %	77,50 %
5	24	13,13 %	75,00 %	32	18,97 %	81,25 %

Tabulka B.28: Výsledky meraní určenia pozícií strihov v ukážke číslo 4 pri šírke okna 15 snímkov použitím chi-square metriky.

	Color layout			Color histogram		
	Strihy: 137	Priem.: 3,24	Odch.: 3,76	Strihy: 137	Priem.: 2,63	Odch.: 2,26
	Nájdené	Úplnosť	Presnosť	Nájdené	Úplnosť	Presnosť
0	199	89,05 %	61,30 %	258	91,24 %	48,44 %
1	149	85,40 %	78,52 %	218	90,51 %	56,88 %
2	135	80,29 %	81,48 %	153	65,69 %	58,82 %
3	121	72,26 %	81,81 %	61	27,00 %	60,65 %
4	60	34,30 %	78,33 %	17	8,02 %	64,70 %
5	11	5,10 %	63,63 %	3	2,18 %	100,00 %

Tabulka B.29: Výsledky meraní určenia pozícií strihov v ukážke číslo 4 pri šírke okna 10 snímkov použitím Manhattan metriky.

	Color layout			Color histogram		
	Strihy: 137	Priem.: 3,24	Odch.: 3,76	Strihy: 137	Priem.: 2,63	Odch.: 2,26
	Nájdené	Úplnosť	Presnosť	Nájdené	Úplnosť	Presnosť
0	153	85,40 %	76,47 %	182	77,37 %	58,24 %
1	130	83,21 %	87,69 %	159	76,64 %	66,03 %
2	118	78,10 %	90,67 %	124	60,58 %	66,93 %
3	108	71,53 %	90,74 %	52	27,00 %	71,15 %
4	56	36,49 %	89,28 %	16	9,48 %	81,25 %
5	11	5,83 %	72,72 %	3	2,18 %	100,00 %

Tabulka B.30: Výsledky meraní určenia pozícií strihov v ukážke číslo 4 pri šírke okna 15 snímkov použitím Manhattan metriky.

Příloha C

Výsledky testov citlivosti porovnávania

	Color layout			Color histogram		
Názov	Euklidová	Chi-square	Manhattan	Euklidová	Chi-square	Manhattan
posun1	9,92	0,41	5,36	1,26	1,22	3,22
posun2	5,41	0,18	4,16	2,55	4,28	4,46
posun3	10,87	0,50	5,77	1,97	2,19	3,82
posun4	8,84	0,311	5,39	2,53	1,29	3,16
posun5	4,07	0,08	3,45	1,84	3,73	4,04

Tabulka C.1: Výsledky meraní citlivosti porovnávania na sade posunutých snímkoch. Hodnoty predstavujú nameranú vzdialenosť.

	Color layout			Color histogram		
Názov	Euklidová	Chi-square	Manhattan	Euklidová	Chi-square	Manhattan
väčší1	6,26	0,16	4,07	8,11	5,63	4,67
väčší2	5,91	0,16	3,86	20,80	34,31	7,72
väčší3	9,87	0,46	5,67	14,43	31,35	7,96
väčší4	21,45	2,30	7,15	9,81	11,71	6,29
väčší5	16,36	1,21	6,13	55,54	90,08	10,87

Tabulka C.2: Výsledky meraní citlivosti porovnávania na sade snímkov s rozdielnými rozmermi. Hodnoty predstavujú nameranú vzdialenosť.

	Color layout			Color histogram		
Názov	Euklidová	Chi-square	Manhattan	Euklidová	Chi-square	Manhattan
kompresia1	23,69	3,57	7,76	38,35	49,81	8,24
kompresia2	18,16	1,54	7,01	45,94	76,30	10,18
kompresia3	20,96	2,77	6,94	41,64	56,45	8,50
kompresia4	23,24	3,01	8,14	7,50	7,60	5,48
kompresia5	18,66	1,59	7,13	6,30	2,38	4,27

Tabulka C.3: Výsledky meraní citlivosti porovnávania na sade snímkov s inou kompresiou. Hodnoty predstavujú nameranú vzdialenosť.

	Color layout			Color histogram		
Názov	Euklidová	Chi-square	Manhattan	Euklidová	Chi-square	Manhattan
rôzne1 / rôzne2	84,93	27,28	14,96	6,36	26,52	7,46
rôzne1 / rôzne3	75,64	23,54	15,75	3,85	10,57	5,75
rôzne1 / rôzne4	101,92	51,59	18,34	17,20	57,56	9,08
rôzne1 / rôzne5	230,54	222,49	28,57	4,33	12,63	5,98
rôzne2 / rôzne3	134,62	63,69	18,82	5,98	24,56	7,24
rôzne2 / rôzne4	109,52	64,00	18,49	19,52	92,47	10,57
rôzne2 / rôzne5	211,74	181,90	26,91	6,42	23,21	7,22
rôzne3 / rôzne4	141,97	93,46	21,82	17,88	66,94	9,61
rôzne3 / rôzne5	227,53	206,78	26,54	5,05	15,36	6,51
rôzne4 / rôzne5	259,25	313,20	30,21	16,41	58,12	9,26

Tabulka C.4: Výsledky meraní citlivosti porovnávania na sade rôznych snímkov. Hodnoty predstavujú nameranú vzdialenosť.

Příloha D

Výsledky testov funkčnosti aplikácie

D.1 Výsledky testov vzoriek s rôznym počtom snímkov za sekundu

Ukázka D.1: Výstup testu pri použití rysov Color layout a Euklidovej metriky.

```
00:44.59,00:46.47d00:45.45
01:08.39,01:08.58c01:08.72,01:08.82
01:08.67a01:08.92,01:09.65
01:22.03a01:22.49,01:22.59
01:22.35a01:22.69,01:23.02
01:27.83a01:27.39,01:27.82
01:29.19a01:28.39,01:29.22
01:29.35a01:29.39,01:30.15
01:37.23a01:36.72,01:37.52
01:38.19a01:37.65,01:38.82
01:42.15a01:41.72,01:42.15
01:50.47a01:50.55,01:50.89
01:51.47a01:51.09,01:51.42
01:51.71a01:51.59,01:51.75
01:56.23,01:56.39d01:57.05
01:57.03a01:57.35,01:57.45
02:04.27a02:03.39,02:04.49
02:05.47a02:05.02,02:05.59
02:11.59a02:11.22,02:11.32
02:18.90a02:17.79,02:18.72
02:19.15a02:18.85,02:19.39
```

Ukázka D.2: Výstup testu pri použití rysov Color layout a chi-square metriky.

```
01:22.35a01:21.72,01:22.59
01:45.91a01:41.72,01:42.55
02:15.07a02:09.52,02:11.59
02:17.90a02:13.35,02:18.72
```

Ukázka D.3: Výstup testu pri použití rysov Color layout a Manhattan metriky.

```
02:32.67a02:12.12,02:32.59
```

Ukázka D.4: Výstup testu pri použití rysov Color histogram a Euklidovej metriky.

```
00:09.47,00:10.10c00:09.22,00:10.35
00:10.43a00:10.59,00:10.82
00:12.95a00:13.32,00:13.59
00:16.83a00:15.65,00:16.85
00:34.71,00:36.35c00:34.45,00:36.09
00:41.39,00:41.63c00:40.95,00:41.55
00:45.35a00:45.32,00:45.82
00:47.03,00:47.23c00:46.79,00:47.09
00:48.63a00:47.22,00:47.35
00:51.11,00:52.03c00:48.85,00:52.35
00:57.23,01:00.15c00:55.92,00:59.92
01:05.75,01:06.43c01:04.85,01:05.39
01:14.39a01:08.35,01:09.52
01:18.03a01:15.32,01:17.75
01:21.87a01:18.75,01:19.89
01:22.91a01:20.82,01:21.05
01:22.99a01:21.29,01:21.75
01:24.47,01:25.71c01:24.29,01:25.85
01:26.79a01:26.45,01:27.75
01:28.59,01:30.43c01:30.15,01:30.72
01:32.07,01:32.99d01:32.05
01:34.39a01:32.89,01:34.39
01:37.67,01:39.39c01:36.72,01:37.59
01:40.35a01:38.39,01:39.89
01:41.59,01:43.23c01:41.59,01:43.29
01:45.15,01:45.35c01:44.75,01:45.15
01:45.55,01:47.79d01:45.25
01:48.07a01:45.35,01:45.45
01:50.03a01:46.49,01:49.72
01:51.27,01:51.51c01:50.15,01:51.59
01:51.99,01:53.99c01:51.99,01:53.95
01:54.71,01:56.07c01:54.29,01:56.09
01:59.63,01:59.75c01:58.89,02:00.52
01:59.87,02:00.87c02:00.69,02:00.85
02:01.95,02:02.03c02:01.42,02:03.62
02:03.11,02:03.95d02:05.05
02:04.55a02:06.22,02:06.55
02:05.03a02:06.72,02:07.15
02:11.46,02:14.75c02:12.68,02:13.99
02:15.62a02:14.29,02:14.72
02:15.87,02:22.51c02:14.92,02:21.85
02:23.79a02:23.12,02:24.12
02:26.99,02:32.18c02:25.12,02:32.52
```

Ukázka D.5: Výstup testu pri použití rysov Color histogram a chi-square metriky.

```
00:09.79a00:09.39,00:09.79
00:10.10a00:10.22,00:10.62
00:12.95a00:12.99,00:13.85
00:16.27a00:15.32,00:16.88
00:23.39a00:22.95,00:23.55
00:29.63a00:28.32,00:29.85
00:51.19,00:51.27d00:49.69
00:51.43a00:50.02,00:51.52
00:57.83,00:58.15c00:57.15,00:58.09
00:58.95,00:59.55c00:58.92,00:59.45
00:59.67,01:00.75c00:59.52,01:00.92
01:18.22,01:19.30c01:16.62,01:17.92
01:21.95a01:19.29,01:19.52
01:22.67a01:20.29,01:20.42
01:23.35a01:21.42,01:21.75
01:23.47,01:23.59c01:21.82,01:21.95
01:23.71,01:26.23d01:22.45
01:27.43,01:27.51c01:24.32,01:27.75
01:38.39,01:39.31c01:37.05,01:37.59
01:40.51,01:41.87c01:38.55,01:39.29
01:42.67,01:43.47d01:40.25
01:47.23a01:42.52,01:44.62
01:51.75a01:46.49,01:50.49
01:54.71a01:52.42,01:54.12
01:56.03a01:54.82,01:56.09
01:59.07,01:59.39c01:59.15,01:59.59
01:59.75a02:00.19,02:00.35
02:01.95a02:02.32,02:03.05
02:02.03a02:03.22,02:03.39
02:05.91,02:07.23c02:06.39,02:07.19
02:10.87,02:12.55d02:10.42
02:13.39a02:12.25,02:13.32
02:15.27a02:14.42,02:14.49
02:17.95,02:19.03d02:17.35
02:19.75,02:22.43c02:18.35,02:21.85
02:23.79a02:23.12,02:24.12
02:25.18a02:25.05,02:25.32
02:26.87a02:25.82,02:28.25
02:30.59a02:29.18,02:30.59
02:30.79a02:30.68,02:30.79
02:31.18a02:30.99,02:31.65
```

Ukázka D.6: Výstup testu pri použití rysov Color histogram a Manhattan metriky.

```
00:14.47a00:13.72,00:14.72
00:15.79a00:15.59,00:16.29
00:24.75a00:23.42,00:23.82
00:50.75a00:50.02,00:50.79
00:52.63a00:52.29,00:52.65
00:57.83,00:58.15c00:56.99,00:58.09
00:58.95a00:58.92,00:58.99
00:59.03,00:59.43c00:59.05,00:59.39
00:59.91a00:59.85,01:00.15
01:00.43a01:00.25,01:00.89
01:16.58a01:14.29,01:14.89
01:17.75a01:16.25,01:17.75
01:18.71,01:19.03c01:19.29,01:22.49
01:21.39a01:24.32,01:26.22
01:21.79,01:21.95d01:26.39
01:24.31,01:26.23d01:28.32
01:29.15,01:29.23d01:30.45
01:31.59a01:31.65,01:31.72
01:33.59a01:33.05,01:33.69
01:38.11,01:38.35c01:35.79,01:36.52
01:38.59,01:39.31c01:36.72,01:37.52
01:40.03a01:37.72,01:38.22
01:40.19a01:38.39,01:40.25
01:41.07a01:41.02,01:41.15
01:42.31a01:42.45,01:42.52
01:44.75,01:44.95d01:44.22
01:45.15,01:45.55d01:44.62
01:46.63,01:47.51d01:45.15
01:48.35a01:45.45,01:45.55
01:49.55a01:46.49,01:49.55
01:53.87,01:54.51d01:54.12
01:55.35,01:55.63c01:54.82,01:55.62
01:56.23,01:56.55d01:56.25
01:57.19,01:57.87d01:57.19
01:59.07,01:59.39c01:59.02,01:59.59
02:00.23a02:00.69,02:01.22
02:00.47a02:01.72,02:02.25
02:04.55,02:06.71c02:04.89,02:04.95
02:07.47a02:05.82,02:07.39
02:07.91,02:08.03d02:07.59
02:08.27,02:08.75d02:08.05
02:12.55,02:13.27c02:09.82,02:10.75
02:13.39a02:11.32,02:13.32
02:15.27a02:14.42,02:14.49
02:16.46,02:16.95c02:15.49,02:16.99
02:18.27a02:18.35,02:19.29
02:18.35,02:22.71c02:19.39,02:21.85
02:23.79a02:23.12,02:23.62
02:24.23a02:23.79,02:24.42
02:25.99a02:25.82,02:26.75
02:31.31a02:30.99,02:32.12
```

D.2 Výsledky testov vzoriek kódovaných rôznym video komprimačným algoritmom

Ukázka D.7: Výstup testu pri použití rysov Color histogram a chi-square metriky.

```
00:09.82a00:09.39,00:10.62
00:22.75a00:21.75,00:22.35
00:46.25,00:46.92c00:41.69,00:42.22
00:58.32,00:59.35d00:48.32
01:00.75a00:49.69,00:51.19
01:03.72,01:03.85c00:52.19,00:53.62
01:05.52a00:56.65,01:00.92
01:06.85a01:03.72,01:05.05
01:19.15a01:16.19,01:17.75
01:21.32,01:22.79c01:19.52,01:21.95
01:26.72,01:27.19c01:26.55,01:27.75
01:29.89a01:30.45,01:30.95
01:31.15a01:31.45,01:32.55
01:31.62,01:32.09d01:32.72
01:37.52,01:37.59c01:37.45,01:39.29
01:39.02,01:39.55d01:40.95
01:41.15,01:41.52d01:42.09
01:41.62,01:41.92c01:42.45,01:43.69
01:42.55,01:43.62d01:44.75
01:47.39,01:48.39c01:47.12,01:49.55
01:52.69,01:52.89d01:53.79
01:54.72,01:55.09c01:54.82,01:55.62
01:59.69,02:00.19d01:59.79
02:00.32,02:01.22c02:00.19,02:01.39
02:01.49,02:01.75d02:01.55
02:06.59,02:07.19c02:03.62,02:04.39
02:07.92a02:06.22,02:07.19
02:14.25a02:12.68,02:13.49
02:15.85,02:17.35c02:14.22,02:15.32
02:18.32,02:19.32c02:16.32,02:19.32
02:21.09a02:20.95,02:21.09
02:22.25a02:21.59,02:23.62
02:26.85,02:30.99c02:25.12,02:32.12
```

Ukázka D.8: Výstup testu pri použití rysov Color histogram a Euklidovej metriky.

```
00:14.69a00:13.85,00:14.22
00:15.15,00:15.55c00:14.39,00:15.65
00:20.38,00:20.82c00:21.99,00:22.92
00:21.75,00:23.69c00:23.39,00:23.85
00:29.75a00:30.09,00:31.65
00:30.05a00:31.89,00:33.32
00:40.39a00:40.82,00:40.95
00:49.92,00:53.72c00:45.99,00:47.09
00:57.79,00:57.92c00:49.69,00:58.15
00:58.85,00:59.85c00:58.85,00:59.79
01:07.02a01:03.89,01:06.62
01:13.25a01:08.02,01:09.19
01:22.72a01:10.85,01:20.02
01:24.32a01:20.15,01:24.29
01:24.52,01:25.75c01:24.45,01:25.85
01:27.95a01:26.95,01:27.75
01:30.42a01:30.15,01:30.95
01:31.52,01:32.05c01:31.55,01:32.55
01:37.65,01:38.32d01:36.72
01:38.52,01:38.92c01:37.05,01:38.72
01:39.42,01:39.69c01:39.22,01:39.55
01:40.49,01:41.05d01:40.55
01:41.55a01:41.59,01:43.69
01:42.45,01:43.25c01:44.35,01:44.62
01:47.39,01:48.99c01:47.12,01:49.59
01:50.55a01:50.69,01:51.39
01:50.82a01:51.59,01:52.35
01:50.95,01:53.99c01:52.49,01:53.95
01:54.72,01:56.09c01:54.29,01:56.09
01:59.39,01:59.49d01:58.62
01:59.59a01:58.75,02:01.15
01:59.79,02:00.59d02:01.29
02:03.05,02:08.52c02:03.09,02:04.39
02:12.39,02:16.15c02:06.22,02:07.15
02:17.85,02:18.02d02:08.79
02:19.79a02:12.68,02:19.68
02:20.12,02:20.45c02:20.02,02:20.59
02:22.25a02:21.59,02:23.62
02:26.85,02:32.02c02:25.12,02:32.52
```

Ukázka D.9: Výstup testu pri použití rysov Color layout a Manhattan metriky.

```
02:25.49,02:32.59d02:32.59
```

Ukázka D.10: Výstup testu pri použití rysov Color histogram a Manhattan metriky.

```
00:05.79,00:07.02d00:07.92
00:08.02,00:09.22d00:09.25
00:09.89a00:10.22,00:10.42
00:21.55a00:19.52,00:20.82
00:52.45,00:52.92c00:49.69,00:53.22
00:57.82a00:56.99,00:57.69
00:59.35,00:59.49c00:59.32,00:59.95
01:15.45,01:15.85d01:14.75
01:17.72,01:20.02c01:16.25,01:17.75
01:21.32,01:21.92c01:19.52,01:21.92
01:24.72,01:26.45c01:24.72,01:25.72
01:27.45a01:26.95,01:27.52
01:30.42a01:30.45,01:30.95
01:31.02a01:31.45,01:32.55
01:31.35,01:32.35d01:32.89
01:38.52a01:36.92,01:37.12
01:38.72,01:39.39c01:37.45,01:39.29
01:41.72a01:41.95,01:43.69
01:42.55,01:44.22d01:44.89
01:46.92,01:48.55c01:47.19,01:49.55
01:50.55,01:51.75d01:51.79
01:55.45,01:56.09c01:54.82,01:56.09
01:56.25,01:56.42d01:56.25
01:58.29,01:59.02d01:59.02
01:59.49,02:01.35c01:59.49,02:01.55
02:05.99a02:06.22,02:06.39
02:06.09,02:09.65c02:06.55,02:07.19
02:11.85a02:09.15,02:09.82
02:12.39,02:12.52d02:10.52
02:12.65,02:12.79d02:10.75
02:13.35,02:13.45d02:11.79
02:14.02a02:12.68,02:13.32
02:14.99,02:15.89c02:14.22,02:15.32
02:16.49,02:16.62c02:16.32,02:19.29
02:16.75,02:19.25d02:19.39
02:22.25a02:21.59,02:23.62
02:22.72a02:23.79,02:23.95
02:25.65a02:25.32,02:25.55
02:26.79,02:31.02c02:25.68,02:32.12
```

D.3 Výsledky testov vzoriek s rozdielnými rozmermi

Ukázka D.11: Výstup testu pri použití rysov Color layout a Euklidovej metriky.

```
01:53.99a01:52.99,01:53.99
```

Ukázka D.12: Výstup testu pri použití rysov Color layout a chi-square metriky.

```
01:24.32,01:25.65d01:25.65
02:02.22,02:03.05d02:02.22
```

Ukázka D.13: Výstup testu pri použití rysov Color layout a Manhattan metriky.

```
00:14.25a00:13.39,00:14.25
00:15.19a00:15.19,00:16.42
```

Ukázka D.14: Výstup testu pri použití rysov Color histogram a Euklidovej metriky.

```
00:08.52,00:09.22d00:08.52
00:09.35,00:10.12c00:09.42,00:10.72
00:13.79a00:13.45,00:14.15
00:14.69,00:15.55c00:14.35,00:15.79
00:23.82,00:27.85c00:23.82,00:27.92
00:28.32a00:28.39,00:28.85
00:29.45a00:29.35,00:31.85
00:36.22a00:33.09,00:36.25
00:39.95a00:37.09,00:39.75
00:40.39a00:40.42,00:41.42
00:41.62a00:42.05,00:42.22
00:42.29a00:42.55,00:43.09
00:44.75a00:43.69,00:45.59
00:46.85,00:53.95c00:46.19,00:47.35
00:57.79a00:48.92,00:52.09
00:58.65,01:05.65c00:52.59,00:56.22
01:12.19a00:57.45,01:04.19
01:20.25,01:26.12d01:05.02
01:30.29a01:07.49,01:12.89
01:31.35a01:13.39,01:19.22
01:32.35a01:22.05,01:25.65
01:33.65,01:33.85c01:26.45,01:33.95
01:34.05,01:34.72c01:34.45,01:34.72
01:35.79a01:35.25,01:36.89
01:36.85,01:37.45c01:37.22,01:40.22
01:37.59,01:40.22d01:40.95
01:41.55,01:41.72c01:41.52,01:41.62
01:42.45,01:45.15c01:41.72,01:42.35
01:46.09a01:43.55,01:44.95
01:46.49,01:48.39d01:45.55
01:50.15a01:46.82,01:49.62
01:50.82,01:51.02d01:50.85
01:56.59a01:56.19,01:56.62
01:59.02,02:00.32d01:58.89
02:00.45,02:01.62d01:59.02
02:07.19,02:07.49d02:07.85
02:08.52,02:11.45d02:10.72
02:14.02,02:14.89c02:13.35,02:14.29
02:16.09,02:24.25c02:17.09,02:33.05
02:25.92,02:33.22c02:33.62,02:33.68
```

Ukázka D.15: Výstup testu pri použití rysov Color histogram a chi-square metriky.

```
00:09.49a00:09.29,00:09.55
00:09.82a00:09.68,00:09.82
00:12.65a00:11.35,00:12.49
00:13.59,00:13.69d00:13.59
00:14.85,00:15.65c00:14.62,00:16.92
00:23.69a00:23.82,00:24.52
00:27.82a00:26.39,00:34.05
00:28.09a00:34.45,00:35.25
00:34.19a00:38.25,00:40.92
00:46.42a00:44.65,00:47.85
00:46.75a00:48.15,00:52.09
00:47.92a00:52.65,00:54.15
00:59.35a00:59.49,00:59.62
01:00.02a01:00.32,01:00.39
01:00.75a01:00.45,01:03.89
01:04.52a01:05.52,01:05.82
01:04.92a01:06.19,01:08.52
01:05.05a01:08.58,01:08.65
01:05.52a01:08.89,01:09.15
01:07.15a01:10.05,01:11.39
01:08.72a01:11.99,01:12.89
01:10.72a01:13.39,01:15.58
01:17.92a01:19.02,01:19.19
01:27.32a01:26.45,01:27.19
01:33.65,01:34.72d01:30.79
01:38.19a01:32.42,01:32.69
01:38.45,01:39.29c01:33.22,01:34.45
01:41.72a01:35.95,01:37.05
01:42.45a01:38.22,01:39.72
01:47.72,01:48.05d01:42.15
01:54.72a01:46.49,01:49.72
02:01.75a01:55.22,01:57.05
02:06.59,02:07.19c01:59.02,01:59.69
02:07.92a02:00.45,02:00.55
02:08.05a02:00.85,02:02.02
02:16.62a02:06.59,02:08.85
02:16.89a02:09.15,02:09.29
02:23.65,02:24.25c02:15.68,02:30.75
02:25.92,02:31.65c02:31.65,02:32.29
02:33.02a02:32.68,02:33.62
```

Ukázka D.16: Výstup testu pri použití rysov Color histogram a Manhattan metriky.

```
00:11.62a00:10.82,00:11.49
00:15.55a00:15.69,00:16.59
00:16.15a00:16.72,00:17.55
00:22.89a00:22.99,00:23.42
00:27.82a00:25.52,00:26.32
00:28.35a00:26.45,00:27.92
00:30.65,00:31.29c00:29.35,00:31.35
00:48.32a00:43.69,00:45.59
00:49.72a00:46.19,00:47.19
00:49.89,00:50.05d00:47.35
00:52.45,00:52.69c00:48.69,00:52.09
00:53.89a00:52.59,00:54.15
00:58.79,00:58.99c00:58.19,00:58.95
00:59.05,00:59.35c00:59.05,00:59.35
01:03.72,01:04.12c01:01.42,01:03.79
01:08.92,01:09.25c01:07.39,01:08.39
01:10.89,01:13.39c01:09.65,01:11.39
01:14.49a01:11.99,01:12.89
01:15.02a01:13.39,01:14.55
01:23.65,01:23.75d01:24.22
01:24.72a01:24.89,01:25.15
01:27.39a01:26.45,01:27.19
01:33.25a01:32.15,01:32.99
01:33.65a01:33.09,01:33.39
01:34.05a01:34.09,01:34.52
01:37.65a01:35.95,01:37.52
01:38.12a01:37.65,01:37.72
01:43.92,01:45.35c01:43.02,01:43.22
01:45.55a01:43.45,01:43.52
01:47.05a01:44.69,01:44.75
01:48.55a01:45.35,01:49.09
01:54.72a01:52.69,01:52.89
01:55.09a01:53.09,01:55.62
01:57.05a01:56.55,01:57.05
02:16.49,02:24.25c02:16.68,02:30.75
02:25.15,02:31.35c02:31.65,02:32.15
02:33.02a02:32.68,02:33.62
```

D.4 Výsledky testov vzoriek s vynechanými zábermi

Ukázka D.17: Výstup testu pri použití rysov Color layout a Euklidovej metriky.

```
00:37.39,01:17.62d00:37.22
```

Ukázka D.18: Výstup testu pri použití rysov Color layout a chi-square metriky.

```
00:37.75,01:04.52d00:37.22
01:04.65,01:14.82d00:37.28
01:21.45,01:21.82d00:42.12
01:24.32,01:27.25d00:46.25
02:01.49,02:02.22d01:21.65
02:02.32,02:03.05d01:21.75
```

Ukázka D.19: Výstup testu pri použití rysov Color layout a Manhattan metriky.

```
02:00.12,02:32.59d01:52.12
```

Ukázka D.20: Výstup testu pri použití rysov Color histogram a Euklidovej metriky.

```
00:38.62,00:41.55d00:39.35
00:41.69,00:47.35d00:41.02
00:48.19,00:48.85d00:42.75
00:49.19,00:56.99d00:42.82
00:57.62,01:05.25d00:43.32
01:06.15,01:10.39c00:43.72,00:45.39
01:14.58,01:19.89d00:49.85
01:20.15,01:20.29d00:50.39
01:20.82,01:27.02d00:50.45
01:27.75,01:31.72c00:50.95,00:51.09
01:35.19,01:37.65c00:56.39,00:58.99
01:38.39,01:39.89d00:59.79
01:41.82,01:44.62c01:01.35,01:01.62
01:44.75a01:02.25,01:03.25
01:46.49,01:49.22c01:06.02,01:08.69
01:52.35a01:11.42,01:11.49
01:52.62,01:53.19c01:12.22,01:12.62
01:53.45,01:53.79d01:13.32
02:00.19,02:00.35c01:19.65,01:20.15
02:07.22,02:07.35d01:26.95
02:12.95,02:13.65d01:33.02
02:19.52,02:19.68c01:39.22,01:39.29
02:19.85,02:20.39c01:39.35,01:39.75
02:20.45,02:20.52c01:39.85,01:39.95
02:27.15,02:27.29d01:46.42
02:30.02,02:32.39c01:49.52,01:51.62
```

Ukázka D.21: Výstup testu pri použití rysov Color histogram a chi-square metriky.

```
00:40.15,00:41.42d00:38.05
00:44.42,00:44.75d00:40.99
00:45.99,00:46.82d00:41.95
00:47.92a00:42.95,00:45.19
00:49.69,00:51.19d00:46.79
00:52.19,00:53.32d00:47.82
00:58.09,01:04.92d00:51.55
01:05.05,01:09.19d00:51.62
01:09.35,01:13.45d00:51.99
01:15.89,01:17.75c00:53.19,00:54.25
01:20.29,01:21.75c00:57.99,00:58.92
01:21.82,01:27.09d00:58.99
01:30.45,01:31.02d01:01.09
01:31.22,01:32.55d01:01.49
01:37.52,01:39.29c01:06.19,01:06.79
01:40.42,01:41.22d01:07.35
01:41.55,01:42.45d01:07.58
01:43.15,01:43.69c01:08.25,01:08.49
01:43.95,01:44.62c01:08.72,01:09.19
01:46.65,01:48.19d01:11.55
01:50.49,01:52.55d01:13.82
01:53.32,01:55.22d01:14.95
01:59.59a01:19.22,01:19.79
02:19.32,02:19.45c01:38.72,01:39.29
02:19.59,02:19.72d01:39.35
02:26.68,02:27.29d01:46.39
02:29.68,02:30.12d01:49.59
02:30.99,02:31.35c01:50.69,01:50.85
02:31.45a01:51.09,01:51.15
```

Ukázka D.22: Výstup testu pri použití rysov Color histogram a Manhattan metriky.

```
00:39.09,00:41.42d00:37.75
00:44.42,00:44.92d00:39.69
00:46.49,00:47.59d00:40.72
00:50.02,01:07.05c00:43.32,00:44.99
01:14.19,01:15.02d00:49.55
01:16.25,01:16.85d00:50.35
01:17.35,01:20.15c00:50.92,00:51.05
01:20.22a00:51.12,00:51.35
01:20.92,01:22.39d00:51.85
01:23.55a00:53.19,00:53.85
01:23.95,01:25.72d00:54.12
01:27.22,01:27.92d00:56.65
01:28.52a00:57.85,00:59.62
01:30.45,01:30.95d01:01.35
01:31.45,01:32.55d01:02.12
01:33.69,01:34.72c01:03.22,01:03.49
01:35.02,01:35.15d01:04.29
01:36.92,01:39.29d01:06.69
01:40.25,01:42.45d01:07.58
01:43.15,01:43.69c01:08.19,01:08.52
01:43.95,01:44.69c01:08.69,01:09.08
01:46.15,01:49.69d01:10.85
01:59.69a01:19.32,01:19.79
02:19.02a01:38.25,01:38.39
02:19.29,02:19.68c01:38.72,01:39.49
02:20.09,02:20.49d01:39.65
02:26.68,02:27.22d01:46.35
02:29.62,02:30.12d01:49.59
02:30.89,02:31.55d01:50.52
```

D.5 Výsledky testov navzájom rôznych vzoriek

Ukázka D.23: Výstup testu pri použití rysov Color layout a Euklidovej metriky.

```
00:00.49,00:04.02c00:10.61,00:14.15
00:06.82,00:09.02c00:21.32,00:22.36
00:13.49,00:20.38d00:27.15
00:20.82,00:21.95c00:27.99,00:33.36
00:24.45,00:49.02d00:37.53
00:52.15,01:07.89c00:44.82,00:44.94
01:07.99,01:14.39c00:45.07,00:45.32
01:15.08,01:16.25d00:57.57
01:17.25,01:24.32d01:03.24
01:25.65,01:48.32c01:06.49,01:13.57
01:49.02,01:56.25c01:13.78,01:16.78
01:57.15,01:58.89d01:19.53
01:59.45,02:14.02c01:21.03,01:22.11
02:15.25,02:20.29d01:26.61
02:21.79,02:24.25c01:28.74,01:33.94
02:26.49,02:32.50c01:34.78,01:55.41
```

Ukázka D.24: Výstup testu pri použití rysov Color layout a chi-square metriky.

```
00:08.79,00:47.92d00:14.15
00:49.35,00:53.62c00:21.32,00:21.65
00:53.75,00:53.89d00:22.61
00:56.29,00:57.15c00:27.99,00:33.44
00:59.09,01:02.09d00:39.24
01:03.85,01:05.72d00:41.94
01:08.82,01:08.92d01:04.40
01:13.99,01:30.42c01:10.49,01:11.82
01:34.35,01:41.02c01:14.61,01:15.65
01:42.15,01:43.02d01:20.53
01:49.42,01:54.72d01:29.82
01:57.45,01:57.89d01:32.11
01:58.29,02:00.72c01:32.82,01:33.49
02:01.09,02:01.22d01:33.94
02:01.35,02:01.62d01:34.07
02:02.12,02:03.22d01:34.32
02:05.39,02:07.59c01:38.28,01:41.40
02:08.32,02:30.99d01:52.49
```

Ukázka D.25: Výstup testu pri použití rysov Color layout a Manhattan metriky.

```
00:13.59,01:54.02d00:32.44
```

Ukázka D.26: Výstup testu pri použití rysov Color histogram a Euklidovej metriky.

```
00:01.62,02:24.25c00:01.82,00:39.24
02:25.92,02:33.02c00:40.53,01:53.65
```

Ukázka D.27: Výstup testu pri použití rysov Color histogram a chi-square metriky.

```
00:01.62,00:28.09c00:01.82,00:07.03
00:29.75,00:37.62c00:13.11,00:25.53
00:39.49,00:41.42c00:28.03,00:29.36
00:41.49,00:44.99c00:30.03,00:31.94
00:46.59a00:34.36,01:03.74
00:47.39,01:14.49c01:05.49,01:06.53
01:16.08,01:17.62d01:07.86
01:18.22,01:22.59c01:08.69,01:09.49
01:23.49,01:25.92d01:11.44
01:27.45,01:28.62c01:13.78,01:16.44
01:28.95,01:29.89c01:16.94,01:17.94
01:30.15,01:30.42c01:19.86,01:23.03
01:31.15,01:33.85d01:24.24
01:36.85a01:26.03,01:28.03
01:37.72,01:40.09d01:29.49
01:41.52,01:42.45c01:31.82,01:34.99
01:43.02,01:43.62d01:35.90
01:45.15,01:58.89d01:38.11
01:59.02a01:38.32,01:38.53
01:59.39,02:01.75d01:39.40
02:06.59,02:33.02c01:50.82,01:53.65
```

Ukázka D.28: Výstup testu pri použití rysov Color histogram a Manhattan metriky.

```
00:05.59,00:11.35c00:10.61,00:24.36
00:12.32,00:15.55d00:26.11
00:19.09,00:27.82c00:27.82,00:39.24
00:29.45,00:41.42c00:40.40,01:06.53
00:42.29,00:48.55c01:08.36,01:09.99
00:49.89,00:56.72d01:11.28
00:57.15,01:14.89d01:11.44
01:14.95,01:15.02d01:12.03
01:15.65,01:22.95c01:13.15,01:15.44
01:23.09a01:16.07,01:16.19
01:23.55a01:16.57,01:20.15
01:24.32,01:30.15d01:21.19
01:30.82,01:33.65c01:21.69,01:21.90
01:33.95,01:36.32c01:22.15,01:24.74
01:37.45,01:39.52c01:26.03,01:28.03
01:40.09,01:42.45c01:28.78,01:32.99
01:42.55,01:43.32c01:33.49,01:33.90
01:43.92,01:59.02d01:34.32
01:59.15,02:02.12c01:34.69,01:34.86
02:02.89,02:03.75d01:36.19
02:04.95a01:38.40,01:39.82
02:05.39a01:40.19,01:40.78
02:05.45a01:41.03,01:41.78
02:05.52,02:33.02c01:43.82,01:53.65
```