Rješenje za prikaz vremenske skale i snimljenog video sadržaja s greškama uočenim pri analizi video zapisa

Birtić, Kristijan

Master's thesis / Diplomski rad

2017

Degree Grantor / Ustanova koja je dodijelila akademski / stručni stupanj: **Josip Juraj Strossmayer University of Osijek, Faculty of Electrical Engineering, Computer Science and Information Technology Osijek / Sveučilište Josipa Jurja Strossmayera u Osijeku, Fakultet elektrotehnike, računarstva i informacijskih tehnologija Osijek**

Permanent link / Trajna poveznica: https://urn.nsk.hr/urn:nbn:hr:200:652069

Rights / Prava: In copyright / Zaštićeno autorskim pravom.

Download date / Datum preuzimanja: **2024-04-19**

Repository / Repozitorij:

Faculty of Electrical Engineering, Computer Science and Information Technology Osijek

SVEUČILIŠTE JOSIPA JURJA STROSSMAYERA U OSIJEKU
FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA I INFORMACIJSKIH TEHNOLOGIJA

Sveučilišni studij

Rješenje za prikaz vremenske skale i snimljenog video sadržaja s greškama uočenim pri analizi video zapisa

Diplomski rad

Kristijan Birtić

Osijek, 2017.

FERIT
FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA I INFORMACIJSKIH TEHNOLOGIJA OSIJEK

Obrazac D1: Obrazac za imenovanje Povjerenstva za obranu diplomskog rada

Osijek, 21.09.2017.

Odboru za završne i diplomske ispite

Imenovanje Povjerenstva za obranu diplomskog rada

Ime i prezime studenta:	Kristijan Birtić
Studij, smjer:	Diplomski sveučilišni studij Računarstvo
Mat. br. studenta, godina upisa:	D 790 R, 09.10.2015.
OIB studenta:	30833538920
Mentor:	Doc.dr.sc. Mario Vranješ
Sumentor:	Jelena Vlaović
Sumentor iz tvrtke:	Kristijan Vučković
Predsjednik Povjerenstva:	Izv. prof. dr. sc. Marijan Herceg
Član Povjerenstva:	Jelena Vlaović
Naslov diplomskog rada:	Rješenje za prikaz vremenske skale i snimljenog video sadržaja s greškama uočenim pri analizi video zapisa
Znanstvena grana rada:	**Obradba informacija (zn. polje računarstvo)**
Zadatak diplomskog rada:	Pri prikazu video sadržaja moguće je uočiti razne artefakte u videu, poput artefakta gubitka paketa, artefakta stvaranja blokova, smrzavanja slike, itd. Postoje razni algoritmi za detekciju takvih artefakata u videu, za koje postoje gotove DLL biblioteke. U radu je potrebno dizajnirati i implementirati rješenje koje će omogućiti prikaz snimljenog video sadržaja s artefaktima detektiranim tijekom analize pomoću postojećih algoritama za detekciju artefakata, uz postojanje vremenske skale sa seek/play/pause mogućnostima. Implementirano rješenje treba omogućiti odloženo pregledavanje snimljenih sadržaja s detektiranim artefaktima. (sumentor Kristijan Vučković, Institut RT-RK Osijek, Cara Hadrijana 10b) (sumentor Jelena Vlaović)
Prijedlog ocjene pismenog dijela ispita (diplomskog rada):	Izvrstan (5)
Kratko obrazloženje ocjene prema Kriterijima za ocjenjivanje završnih i diplomskih radova:	Primjena znanja stečenih na fakultetu: 3 bod/boda Postignuti rezultati u odnosu na složenost zadatka: 3 bod/boda Jasnoća pismenog izražavanja: 3 bod/boda Razina samostalnosti: 3 razina
Datum prijedloga ocjene mentora:	21.09.2017.

Potpis mentora za predaju konačne verzije rada u Studentsku službu pri završetku studija:	Potpis:
	Datum:

FERIT
FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA
I INFORMACIJSKIH TEHNOLOGIJA **OSIJEK**

IZJAVA O ORIGINALNOSTI RADA

Osijek, 27.09.2017.

Ime i prezime studenta:	Kristijan Birtić
Studij:	Diplomski sveučilišni studij Računarstvo
Mat. br. studenta, godina upisa:	D 790 R, 09.10.2015.
Ephorus podudaranje [%]:	1

Ovom izjavom izjavljujem da je rad pod nazivom**: Rješenje za prikaz vremenske skale i snimljenog video sadržaja s greškama uočenim pri analizi video zapisa**

izrađen pod vodstvom mentora Doc.dr.sc. Mario Vranješ

i sumentora Jelena Vlaović

moj vlastiti rad i prema mom najboljem znanju ne sadrži prethodno objavljene ili neobjavljene pisane materijale drugih osoba, osim onih koji su izričito priznati navođenjem literature i drugih izvora informacija.
Izjavljujem da je intelektualni sadržaj navedenog rada proizvod mog vlastitog rada, osim u onom dijelu za koji mi je bila potrebna pomoć mentora, sumentora i drugih osoba, a što je izričito navedeno u radu.

Potpis studenta:

SADRŽAJ

1. UVOD 1

2. DIGITALNI VIDEO SIGNAL 2

2.1. Općenito o video zapisu 2

2.2. Načini analiziranja video zapisa 3

2.3. Sažimanje video zapisa 5

2.4. Prostorna i vremenska izobličenja video zapisa 8

3. POSTOJEĆI ALGORITMI I ALATI ZA OTKRIVANJE ARTEFAKATA 16

3.1. Algoritmi za otkrivanje artefakata u videu 16

3.2. Razvojno okruženje Qt 16

4. PROGRAMSKO RJEŠENJE ZA PRIKAZ VREMENSKE SKALE I SNIMLJENOG VIDEO SADRŽAJA S GREŠKAMA UOČENIM PRI ANALIZI VIDEO ZAPISA 20

4.1. Zadatak koji je bilo potrebno riješiti 20

4.2. Detalji rada programskog rješenja 22

5. TESTIRANJE FUNKCIONALNOSTI PROGRAMSKOG RJEŠENJA 26

5.1. Provođenje testiranja 26

5.2. Nedostaci programskog rješenja 29

6. ZAKLJUČAK 30

LITERATURA 31

SAŽETAK 33

ABSTRACT 34

ŽIVOTOPIS 35

1. UVOD

Pri prikazu video sadržaja moguće je uočiti razne artefakte (engl. *artifact*), tj. vizualna izobličenja nastala zbog gubitka paketa pri prijenosu videa mrežom, zbog stvaranja blokova (engl. *blocking)* uzrokovanih postupkom sažimanja (engl. *compression*), zbog smrzavanja slike (engl. *freezing*) itd. Iako takve greške najčešće nastaju prilikom sažimanja video zapisa i prijenosa, one mogu nastati i zbog same opreme za snimanje niže kvalitete. Postoje razni gotovi algoritmi za otkrivanje grešaka u video zapisu za koje postoje gotove biblioteke. Nakon primjene nekog od gotovih algoritama i provedbe obrade video zapisa tim algoritmima, stvara se datoteka sa zapisom o greškama (lokacija greške, dimenzije oštećenog područja, ...). U ovom diplomskom radu riješen je problem prikazivanja vremenske skale i snimljenog video sadržaja s greškama uočenim pri primjeni algoritama za obradu video zapisa. Programsko rješenje na osnovu datoteke sa zapisom o greškama stvara vremensku skalu na kojoj su oznake za svaku grešku koja postoji u obrađenom video zapisu. Nadalje, u stvorenom rješenju omogućeno je pregledavanje snimljenih sadržaja u kojima se pojavljuju greške s mogućnostima pokretanja (engl. *play*), zaustavljanja (engl. *pause*) i pretraživanja (engl. *seek*).

U drugom poglavlju opisan je video zapis, načini obrade video zapisa, tehnike sažimanja i vrste artefakata koje se javljaju. U trećem poglavlju navedeni su algoritmi za otkrivanje artefakata i opisano je razvojno okruženje Qt. U četvrtom poglavlju opisano je programsko rješenje i zadatak koji rješava. U petom poglavlju opisano je testiranje programskog rješenja.

2. DIGITALNI VIDEO SIGNAL

2.1. Općenito o video zapisu

Video zapis (dalje u radu – video) je niz pokretnih slika prikazanih u vrlo brzom slijedu. Postoje analogni i digitalni video zapisi, no zbog veće kvalitete i manje cijene, danas se pretežito koriste digitalni zapisi. Filmovi se snimaju s 24 slike, odnosno okvira (engl. *frame*) po sekundi (engl. *frames per second).* To je najmanji broj okvira u sekundi kojim se dobiva dojam kontinuiranog pokreta. Standardi poput *Phase Alternating Line* (PAL) i *Séquentiel couleur à mémoire* (SECAM) određuju 25 okvira po sekundi, dok *National Television System Committee* (NTSC) određuje 29.97 okvira po sekundi. U početku se video koristio u filmskoj industriji, a kasnije i u televiziji. Pojavom digitalnih medija poput *digital versatile disc* (DVD), proširilo se korištenje digitalnog videa u domovima. Razvojem interneta, računala i pametnih telefona (engl. *smartphone*), video zapisi se koriste svakodnevno u sve većem obujmu. Poznati servisi poput Youtube-a, Twitch-a i mnogih drugih omogućavaju korisnicima da bilo kada gledaju video sadržaje preko računala, tableta ili pametnog telefona.

Video karakterizira više svojstava poput visine i širine okvira (veličine okvira), broja okvira u sekundi, formata u kojem je snimljen, dužine trajanja, omjera slike (engl. *aspect ratio*), dubine boje (engl. *color depth*) i dr. Uobičajene veličine okvira (engl. *resolution*), odnosno rezolucije video zapisa koje se danas koriste u televiziji i na računalima su 1280 elemenata slike (engl. *pixel*) širine i 720 visine te 1920 x 1080 elemenata slike. Rezolucija 1280 x 720 se još naziva i HD (engl. *high definition*), a 1920 x 1080 elemenata slike FHD (engl. *full high definition*). Također postoji UHD (engl. *ultra high definition*) ili 4K veličine 3840 x 2160 elemenata slike te još veća UHD 8K rezolucija veličine 7680 x 4320 elemenata slike. Iako se većim rezolucijama dobiva jasnija i oštrija slika, problem je što takvi video zapisi u izvornom obliku trebaju jako veliku količinu resursa pri prijenosu i pohrani. Ako se kao primjer uzme HD video signal veličine 1280 x 720 elemenata slike, trajanja 10 minuta, odnosno 600 sekundi, sniman s 30 okvira u sekundi i dubine boja od 24 bita, dobiva se sljedeća ukupna količina bita potrebna za njegovu pohranu.

$$broj\ elemenata\ slike\ po\ okviru = širina\ okvira \cdot visina\ okvira \tag{2-1}$$

Za primjer HD video signala broj elemenata slike po okviru je 921600. Dalje je potrebno izračunati broj bita po okviru.

$$broj\ bita\ po\ okviru = broj\ elemenata\ slike \cdot dubina\ boje \tag{2-2}$$

Prema formuli (2-2), za HD signal iz gornjeg primjera dobije se da je broj bita po okviru 22.12 Mbit. Nakon toga je potrebno izračunati broj bita po sekundi videa.

$$broj\ bita\ po\ sekundi = broj\ bita\ po\ okviru \cdot broj\ okvira\ po\ sekundi \quad (2\text{-}3)$$

Prema formuli (2-3), za HD signal iz gornjeg primjera dobije se da je broj bita po sekundi 663.55 Mbit/s. Naposljetku se može izračunati veličina video zapisa.

$$veličina\ video\ zapisa = broj\ bita\ po\ sekundi \cdot trajanje \quad (2\text{-}4)$$

10 minuta videa HD rezolucije zauzima 49.77 GB. Za gledanje takvog video sadržaja preko interneta, korisnik bi trebao imati brzinu preuzimanja (engl. *download speed*) od 663 Mbit/s, odnosno 83 MB/s. Takve brzine je nemoguće dobiti za kućnu uporabu. S druge strane, film trajanja od 120 minuta bi bio veličine od oko 600 GB. Najveći diskovi na koje se pohranjuju filmovi su *Blu-ray* diskovi veličine do 128 GB što je mnogo manje nego veličina filma izvorne kvalitete. Zato su se razvile razne tehnike sažimanja video zapisa s ciljem smanjenja njegove veličine uz zadržavanje kvalitete [1, 2].

2.2. Načini analiziranja video zapisa

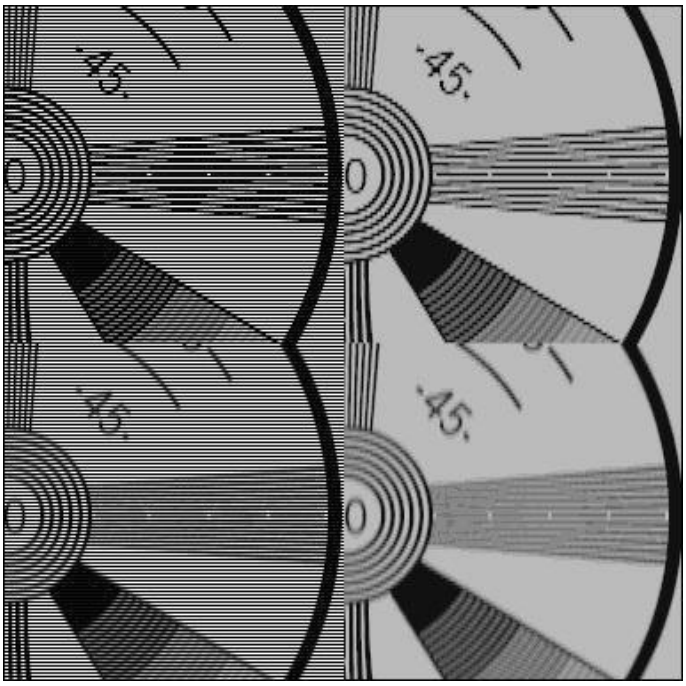
Video može biti progresivan (engl. *progressive*) ili isprepleten (engl. *interlaced*). Uglavnom se uz rezoluciju video zapisa stavlja oznaka „p“ ako je progresivan, odnosno „i“ ako je isprepleten. Na primjer, 720p označava progresivan video dimenzija 1280 x 720 elemenata slike. Kod progresivnog videa, svaki okvir se prikazuje u potpunosti od vrha prema dnu, dok se kod isprepletenog videa svaki okvir dijeli na redove od kojih se stvaraju dva polja (engl. *field*). Jedno polje čine svi parni redovi jednog okvira, a drugo polje svi neparni redovi tog istog okvira. Kada se kod progresivnog videa prikazuje jedan puni okvir, kod isprepletenog se prikazuje jedno polje, odnosno pola punog okvira. Zato će se video zapisi s primjerice 30 okvira po sekundi prikazivati kao 30 okvira po sekundi kod progresivnog, odnosno 60 okvira po sekundi kod isprepletenog video zapisa. Time se postiže bolji prikaz objekata u pokretu u slučaju isprepletenog videa, a nije veće veličine od progresivnog video zapisa. Ovakav pristup se koristio kod CRT (engl. *cathod ray tube*) zaslona i danas je koristi kod SD (engl. *standard definition*) televizije i 1080i HDTV (engl. *high-definition television*) standarda.

Iako je isprepleteni video zapis dobro rješenje za udvostručenje okvira po sekundi bez da se poveća brzina prijenosa video zapisa (engl. *bitrate*), postoje određeni problemi. Jedan od problema je češljanje (engl. *combing*), a može se vidjeti na slici 2.1. Do ove pojave dolazi kada se objekt kreće brzo pa je dio njega snimljen u jednom polju, dok je drugi dio snimljen u drugom.

Ovu je pojavu moguće uočiti prilikom reprodukcije videa koji se prikazuje s manjim brojem okvira nego što je sniman.

Sl. 2.1. Pojava češljanja [3]

Drugi problem koji se javlja kod isprepletenih video zapisa je titranje (engl. *interline twitter*). Do njega dolazi kada se pojavi objekt s mnogo paralelnih crta, primjerice čovjek koji nosi prugastu majicu. Na videu se pruge ne vide jasno, već su spojene u oblike i ostavljaju dojam kao da titraju kao što se vidi na slici 2.2 [4].



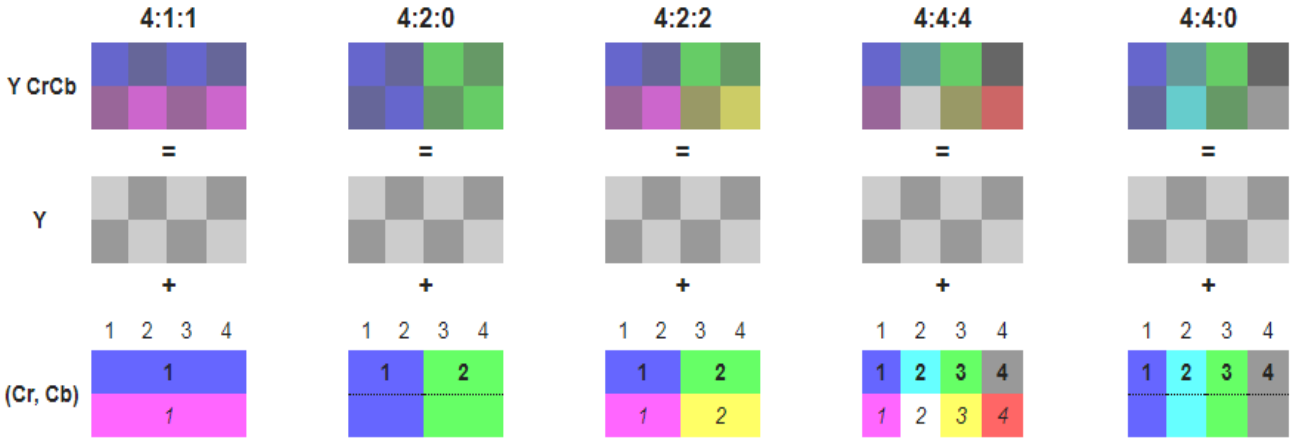
Sl. 2.2. Pojava titranja [3]

2.3. Sažimanje video zapisa

Budući da su izvorni video zapisi jako veliki i nepraktični za svakodnevnu uporabu, potrebno ih je sažeti. Cilj sažimanja je smanjiti veličinu video zapisa što je više moguće bez primjetnog smanjenja kvalitete. Postoje sažimanja bez gubitaka (engl. *lossless*) i s gubicima (engl. *lossy*).

Jedna od uobičajenih metoda smanjenja veličine video zapisa je poduzorkovanje komponenata boje (engl. *chroma subsampling*). Video signali su rastavljeni u tri komponente, jednu svjetlosnu i dvije za boje. Ova metoda oslanja se na činjenicu da je ljudsko oko manje osjetljivo na razlike u bojama nego one u osvjetljenju. Zato se kod spremanja podataka za svaki okvir sprema manje podataka za komponente boje nego za komponentu osvjetljenja. Primjer takvog rastavljanja može se vidjeti na slici 2.3.

Sl. 2.3. Rastavljanje video signala na svjetlosnu i komponentu boje [5]



Sl. 2.4. Primjeri različitih metoda poduzorkovanja boje [6]

Shema poduzorkovanja je zamišljeno područje širine 4 elementa slike i visine 2 elementa slike koje se izražava kao omjer Y:Cr:Cb. Y predstavlja svjetlosnu komponentu, odnosno broj stupaca u zamišljenom području, Cr predstavlja broj uzoraka boja, a Cb predstavlja broj promjena boja između prvog i drugog reda zamišljenog područja. Na slici 2.4. je teorijski prikazano kako radi poduzorkovanje.

Kod omjera 4:4:4 nema poduzorkovanja jer se koristi ista količina podataka za osvjetljenje kao i za boje. Moguće je izračunati relativno smanjenje količine podataka (faktor veličine dan izrazom (2-5)) bilo kojeg omjera poduzorkovanja s omjerom 4:4:4, tako da se zbroj sve tri komponente podjeli s 12.

$$faktor\ veličine = \frac{Y+Cr+Cb}{12} \tag{2-5}$$

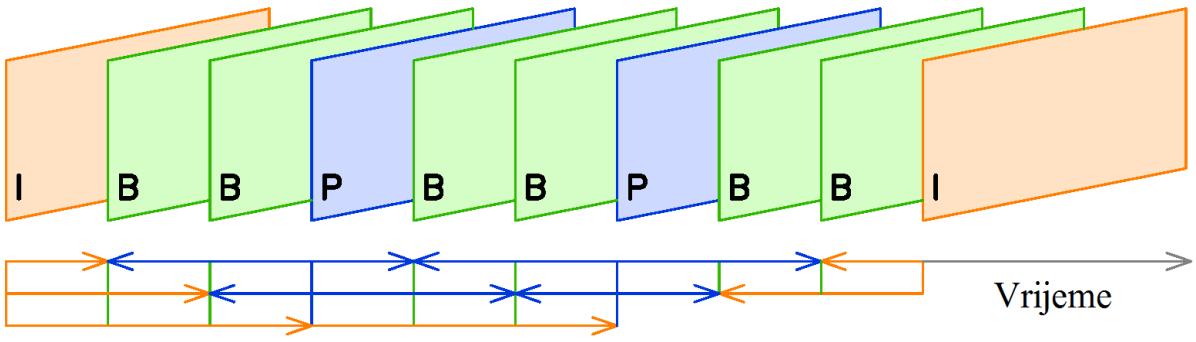
Na primjer, video zapis s omjerom 4:2:2 bi bio manji za 33% od video zapisa bez poduzorkovanja jer faktor veličine ispada (4+2+2)/12, tj. 0.67.

Omjer 4:2:2 koristi se kod sustava za profesionalno snimanje filmskog sadržaja, primjerice kod Panasonic-ovih DVCPRO 50 i DVCPRO HD formata. Omjer 4:2:0 sadrži dvostruko manje podataka od 4:4:4 te ima vrlo raširenu primjenu. Koristi se u standardima H.261, H.262, H.264, H.265 i dr. Filmovi na DVD-ovima i *Blu-ray* diskovima su spremljeni u 4:2:0 omjeru. Na slikama 2.5. i 2.6. može se vidjeti razlika između izvorne slike i slike spremljene omjerom poduzorkovanja boja 4:2:0. Iako je razlika vidljiva, slika smanjene kvalitete je dovoljno dobra za neprofesionalno i kućno korištenje [5, 7].

Sl. 2.5. Primjer slike bez poduzorkovanja [7]

Sl. 2.6. Primjer slike s poduzorkovanjem 4:2:0 [7]

Skupina slika (engl. *group of pictures* - GOP) je način spremanja okvira u video zapisu. Ideja je da se samo nekoliko okvira spremi u potpunosti, dok se kod ostalih okvira sprema samo dio informacija od kojih je potrebno ponovo izraditi potpuni okvir da bi se mogao prikazati. GOP može sadržavati I (engl. *intra*), P (engl. *predicted*) i B (engl. *bidirectional*) okvire te se sastoji od proizvoljnog broja okvira (obično oko 15 okvira). Na slici 2.7. dan je primjer uobičajene strukture GOP-a.



Sl. 2.7. Struktura GOP-a [8]

GOP-ova može biti mnogo u video zapisu. I okviri su spremljeni u potpunosti. Još se nazivaju ključni okviri i pomoću njih se ostali okviri mogu ispravno dekodirati. Oni sadrže najviše podataka pa su zato i memorijski najzahtjevniji. P okviri sadrže informacije o promjenama

naspram najbližeg prijašnjeg I ili P okvira. Budući da ovi okviri nisu spremljeni u potpunosti, njihov sadržaj se predviđa na osnovu prijašnjih I ili P okvira i informacija pohranjenih u okviru. B okviri sadrže još manje informacija od P okvira te su oni najmanje veličine (u smislu memorije). Njihov se sadržaj na osnovu prijašnjeg i idućeg P ili I okvira predviđa i izgrađuje.

Uobičajeno GOP počinje I okvirom, slijede nekoliko B okvira pa P okvir pa opet isti broj B okvira itd. Dugi GOP nizovi prikladni su za dijelove videa gdje nema mnogo kretanja i promjena, dok se kod brzih scena koriste kraći nizovi. GOP-om se može višestruko puta smanjiti veličina video zapisa ili video toka, no gubi se na kvaliteti slike kod brzih scena [9].

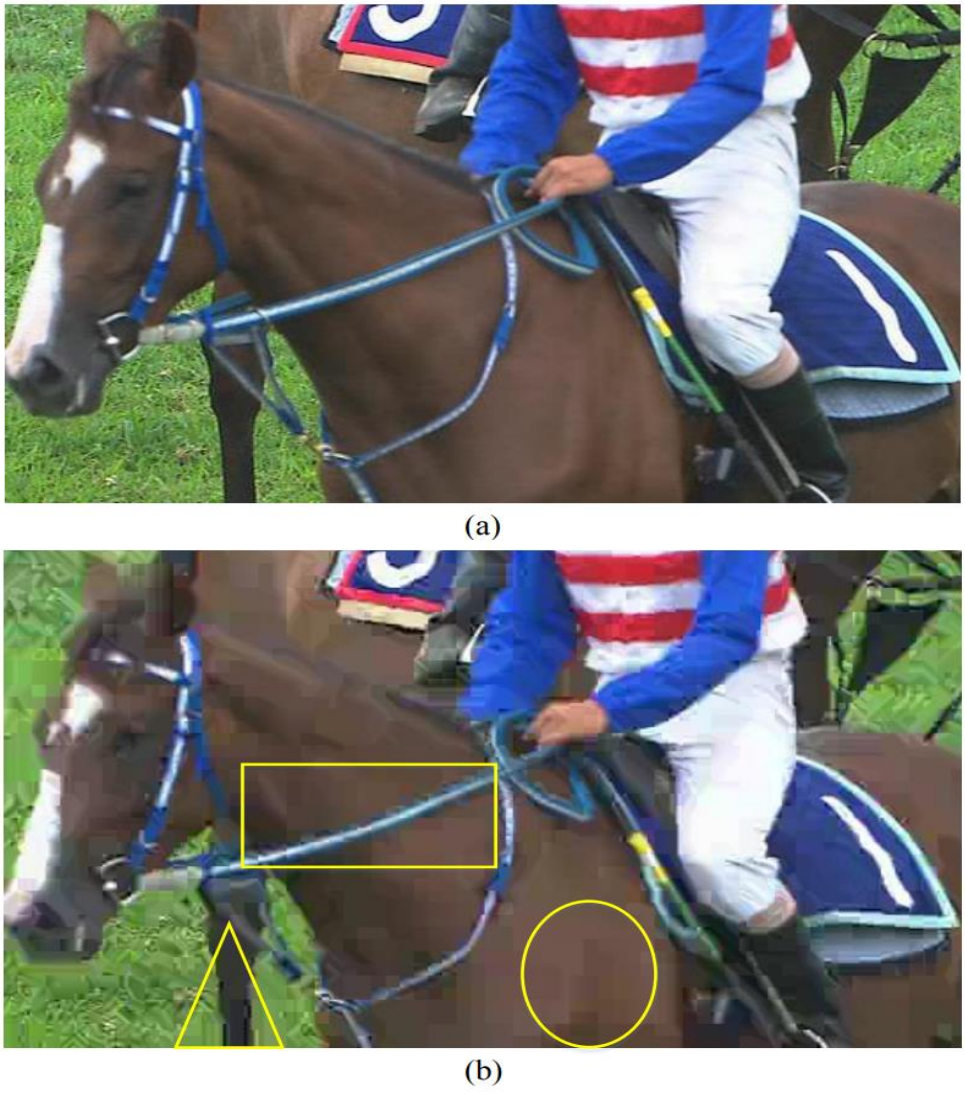
Moving Picture Experts Group (MPEG) je udruga stvorena do strane *International Organization for Standardization* (ISO) i *International Electrotechnical Commission* (IEC) organizacija za standardizaciju prijenosa, formata i sažimanja audio i video zapisa. Svaki od standarda podijeljen je u nekoliko dijelova koji obuhvaćaju sve, od prijenosnih tokova, video i audio zapisa, tehnika sažimanja, sklopovlja (engl. *hardware*) i dr. 1993. godine uveden je MPEG-1 standard za sažimanje video i audio zapisa. Propusnost (engl. *bandwidth*) je bila ograničena na 1.5 Mbit/s i zbog toga je kvaliteta bila niska. MPEG-1 se uglavnom koristio za sažimanje videa kako bi stao na kompaktni disk (engl. *compact disc* - CD). 1995. je postavljen MPEG-2 2. dio, poznatiji kao H.262. On je poboljšao sažimanje i uveo podršku za isprepletene i video zapise visoke definicije (engl. *high definition*). Koriste se GOP te 4:2:2 i 4:2:0 omjeri poduzorkovanja. Primjene je našao u video zapisima na DVD, HD DVD i *Blu-ray* medijima. MPEG-4 10. dio, poznatiji pod nazivom H.264, je uveden 2003. godine i napravljeno je više inačica s unaprjeđenjima kroz godine. To je danas najkorišteniji standard za sažimanje video zapisa. Za istu kvalitetu slike potreban je 50 % manji protok podataka naspram H.262 standarda. Uvela se podrška za rezolucije do 4096 x 2304 elemenata slike, bolja primjena GOP metode, korištenje od 8 do 14 bita po boji, omjeri poduzorkovanja 4:2:0, 4:2:2 i 4:4:4 i mnoge druge novine i poboljšanja. 2012. godine je uveden HEVC ili H.265 standard koji proširuje i poboljšava mogućnosti H.264 [10 – 13].

2.4. Prostorna i vremenska izobličenja video zapisa

Greške u video zapisima najčešće se stvaraju prilikom visoke razine sažimanja ili lošeg algoritma za sažimanje. No, do njih može doći i prilikom grešaka u prijenosnom toku (engl. *transport stream*). Većina njih nastaje kod uređaja za dekodiranje (engl. *decoder*) koji iz sažetog video zapisa pomoću dostupnih informacija matematički izračunava i izgrađuje sve okvire kako bi se mogli prikazati. Ovaj se proces još naziva i dekompresija (engl. *decompression*). Artefakti se

dijele na dvije skupine; prostorne (engl. *spatial*) i vremenske (engl. *temporal*). Prostorni se artefakti mogu uočiti i kada je video zapis zaustavljen i oni se pojavljuju unutar okvira, dok se vremenski artefakti uočavaju tijekom reprodukcije sadržaja jer do njih dolazi zbog promjene položaja objekata između uzastopnih okvira.

Pojava blokova na okviru u video zapisu (engl. *blocking*) je prostorni artefakt koji nastaje prilikom sažimanja video toka. Budući da zbog sažimanja nema dovoljno podataka za točnu i detaljnu izgradnju slike, uzimaju se prosječne vrijednosti za blokove elemenata slike. Time se stvaraju četvrtasta područja slične ili približne boje kao i okolini prostor. Stvaranjem tih blokova gubi se na detaljima i oštrini slike. Gledateljima je lako uočiti pojavu blokova i često im jako smeta prilikom reprodukcije videa. Kod velikih zaslona manjih rezolucija ova pojava je uočljivija nego kod zaslona veće rezolucije. Na slici 2.8. dan je primjer referentne slike i slike s pojavom blokova.



Sl. 2.8. a) referentna slika, b) slika s pojavom blokova [14]

Stvaranje uzorka baze (engl. *basis pattern*) pojavljuje se u područjima s teksturama koje sadrže mnogo detalja poput trave, drveća, morskih valova i sl. U tim područjima na slici dolazi do vidljivog gubitka kvalitete. Ova vrsta greške slična je pojavi blokova jer se zbog sažimanja gube informacije o tom dijelu slike pa uređaj za dekodiranje prilikom izgradnje slike pretpostavlja sadržaj velike količine sličnih blokova elemenata slike i uzima njihove prosječne vrijednosti. Zato takva područja izgledaju mutno i zamrljano. Na slici 2.9. dan je primjer pojave stvaranja uzorka baze. Ova vrsta greške također je prostorni artefakt.

Sl. 2.9. Pojava stvaranja uzorka baze [14]

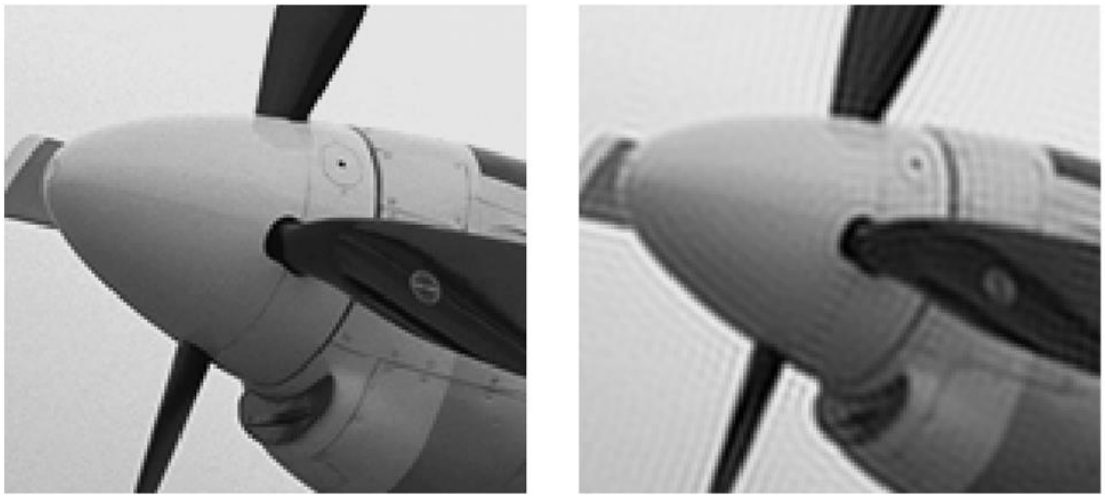
Izlijevanje boja (engl. *color bleeding*) je prostorni artefakt kod kojega je karakteristično da se kod naglih promjena boja jedna boja prelijeva preko druge. Do ove vrste greške dolazi zbog visokog omjera poduzorkovanja boje. Primjer je dan na slici 2.10.

Sl. 2.10. Izlijevanje boja [14]

Rubni šum (engl. *edge busyness*) je vremenski artefakt koji je moguće uočiti kod rubova objekata. Primjer je dan na slici 2.11. Još se naziva i šum komarca (engl. *mosquito noise*) jer se pojavljuje prilikom promjene okvira u video zapisu i ostavlja dojam komarca koji leti oko glave čovjeka. Vremenski je artefakt jer se uočava tijekom reprodukcije.

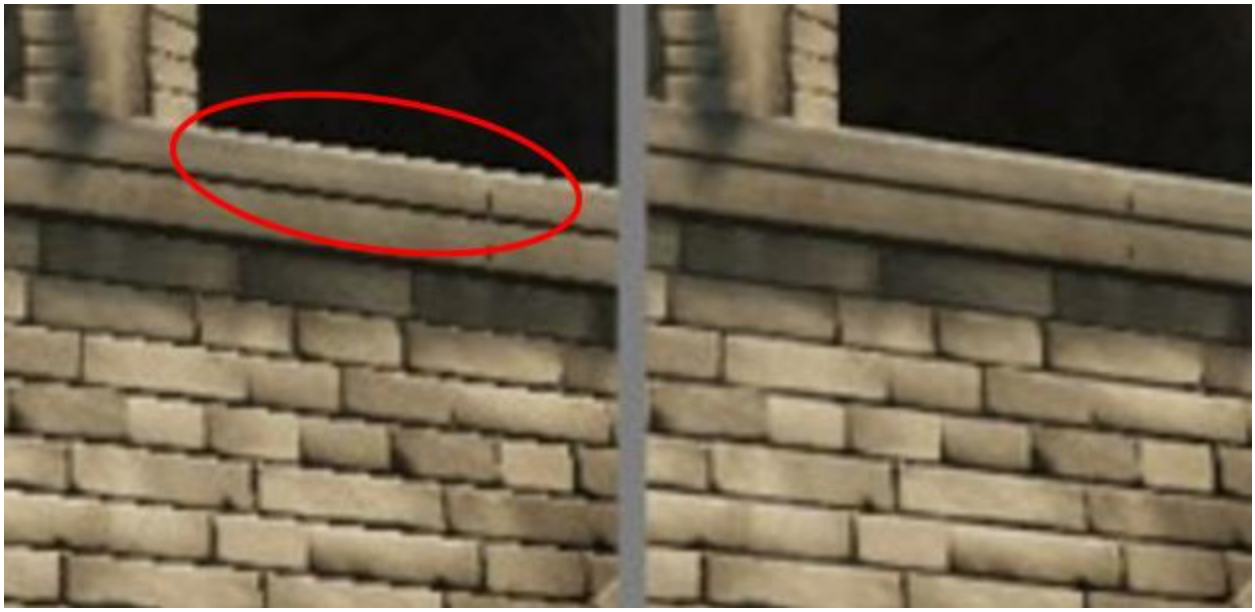
Sl. 2.11. Rubni šum ili šum komarca [14]

Zvonjava (engl. *ringing*) je prostorni artefakt koji nastaje zbog predviđanja sadržaja slike matematičkim izračunima. Zato dolazi do pojave stvaranja obruba oko objekata. Primjer je dan na slici 2.12.



Sl. 2.12. Artefakt zvan „zvonjava“ [14]

Pojava stepenica (engl. *staircase noise*) je jedan oblik pojave blokova kod dijagonalnih rubova. Dijagonalni rubovi su u tom slučaju nazubljeni i izgledom podsjećaju na stepenice. Primjer je dan na slici 2.13 [14 - 16].



Sl. 2.13. Pojava stepenica [14]

Artefakt gubitka paketa (engl. *packet loss*) je oblik artefakta koji se događa zbog smetnji u prijenosnom toku. Svaki okvir sastoji se od nekoliko paketa i zbog gubitka nekih paketa nastaje nepotpuni okvir. Takav okvir izgleda jako slično kao kod pojave blokova, no razlika je u tome što ti blokovi nisu nastali računanjem prosječnih vrijednosti susjednih elemenata slike, već podaci za njih nedostaju. Na slici 2.14. dan je primjer gubitka paketa.

Sl. 2.14. Gubitak paketa [17]

Na slici 2.15. dan je još jedan primjer gubitka paketa. Na ovom primjeru izgubljen je velik broj paketa pa je izgrađena slika jako niske kvalitete i korisniku negledljiva.

Sl. 2.15. Gubitak velike količine paketa

Izbjeljivanje boja (engl. *color fading*) je tip artefakta kod kojega dolazi do promjene osvjetljenja video signala. Boje na slici izgledaju izblijedjelo. Na slici 2.16 dan je primjer artefakta izbjeljivanja boja.

Sl. 2.16. Izbjeljivanje boja [17]

Kidanje slike (eng. *image tearing*) je artefakt koji se javlja prilikom reprodukcije video zapisa. Nastaje zbog razlike brzine osvježavanja (engl. *refresh rate*) zaslona uređaja i broja okvira po sekundi tog videa. Primjer je dan na slici 2.17.

Sl. 2.17. Kidanje slike [17]

Šum (engl. *noise*) je tip artefakta koji je sličan ranije opisanom rubnom šumu. Za razliku od rubnog šuma, šum se odnosi na cijelu sliku, a ne samo na rubove objekata. Primjer je dan na slici 2.18.

Sl. 2.18. Šum [17]

Do šuma dolazi kada se zbog pogreške tijekom prijenosa video signala promijene vrijednosti za osvjetljenje i boje elemenata slike.

Smrzavanje slike (engl. *freezing*) je oblik artefakta do kojega dolazi zbog smetnji u prijenosnom toku. Za razliku od gubitka paketa, ovdje se gube cijeli okviri i korisnik određeno vrijeme vidi isti okvir ili potpuno crni okvir [17].

Većina artefakata može se ispraviti u takvoj mjeri da krajnji korisnici dobivaju video zadovoljavajuće kvalitete. Boljim i pametnijim algoritmima za sažimanje videa postiže se manja količina vidljivih artefakata. Jači uređaji za dekodiranje koji mogu raditi s kompleksnijim algoritmima za dekompresiju također osiguravaju bolju kvalitetu i manje vidljivih artefakata u video zapisu.

Obrada videa se radi zbog poboljšanja kvalitete videa. Potrebno je biti u mogućnosti vidjeti gdje i kada se artefakt dogodio da bi se moglo otkriti zašto je došlo do njega i konačno spriječiti da se ponovi. Programsko rješenje koje je izrađeno u sklopu ovog diplomskog rada obradit će rezultate dugotrajne obrade videa i omogućiti korisniku da ima pregled pojave otkrivenih artefakata na vremenskoj skali (engl. *timeline*) i pregled video isječka u trenucima kada je došlo do grešaka.

3. POSTOJEĆI ALGORITMI I ALATI ZA OTKRIVANJE ARTEFAKATA

3.1. Algoritmi za otkrivanje artefakata u videu

Postoje razni algoritmi za otkrivanje pojedinih artefakata u video signalu, koji kao rezultat obrade videa daju razne podatke o pojedinom artefaktu (lokacija, dimenzije, vidljivost artefakta, trajanje artefakta itd.). Rješenje izrađeno u sklopu ovog diplomskog rada napravljeno je kao proširenje rješenja PQM (engl. *picture quality measurement*) *Embedded* korištenog od strane tvrtke partnera. PQM služi za otkrivanje artefakata u video signalu u stvarnom vremenu i koristi algoritme za otkrivanje sljedećih artefakata: pojava blokova, smrzavanje slike, gubitak paketa, šum, izbjeljivanje boja, titranje slike (engl. *image jitter*), kidanje slike, zamućenje slike (engl. *blurring*) i pojavu zvonjave. Više informacija o PQM-u i uređaju na kojemu se pokreće može se pronaći u [18], dok se više o pojedinim artefaktima implementiranim u PQM rješenje može pronaći u [17].

3.2. Razvojno okruženje Qt

Qt je programsko okruženje za razvoj programske podrške (engl. *software*) za stolna računala (engl. *desktop*), ugradbene (engl. *embedded*) i prijenosne sustave. Pri tome nema potrebe za značajnim prilagođavanjem koda te brzina i mogućnosti ostaju iste na svim platformama. Qt trenutno razvija The Qt Company u sklopu Qt Project koji je projekt otvorenog koda i uključuje mnoge programere i tvrtke koje doprinose razvitku i unaprjeđenju Qt-a.

Razvoj Qt-a započela su 1990. godine dvojica norveških programera, Eirik Chambe-Eng i Haarvard Nord. Oni su osnovali tvrtku Trolltech koju su kroz godine više puta kupile različite tvrtke. Danas se ta tvrtka zove The Qt Company i u vlasništvu je Digia Plc. 1995. je izašla prva verzija Qt-a i još uvijek je u razvoju.

Qt ima svoje integrirano razvojno okruženje (engl. *integrated development environment* - IDE) koje se zove *Qt Creator*. Služi za pisanje koda te omogućuje pametno nadopunjavanje koda, označavanje sintaksnih pogrešaka, pronalazak pogrešaka (engl. *debugging*), sustav pomoći i dr. No, moguće je koristiti bilo koje drugo razvojno okruženje. Qt koristi C++ programski jezik proširen s mnogim Qt modulima poput Qt GUI (engl. *graphical user interface*) za rad s korisničkim sučeljem, Qt *Multimedia* za rad s video zapisima, radio uređajem i kamerom, Qt SQL (engl. s*tructured query language*) za rad s bazama podataka i dr. Postoji mehanizam koji se zove signali i utori (engl. *signals and slots*). On omogućava da se povežu događaji s objektima. Svaki

objekt ima svoje signale i utore. Signali predstavljaju signale koji se odašilju prema drugom objektu kada se dogodio događaj. Utor predstavlja funkciju koja se izvršava kada se pojavi određeni signal. Signal jednog objekta se povezuje s utorom drugog. Nema ograničenja koliko signala može biti povezano na jedan utor i obrnuto. U tablicama 3.1., 3.2. i 3.3. dani su dijelovi koda u kojemu je pokazan rad sa signalima i utorima [19].

Klasa *Counter* ima atribut *value* koji sprema jednu cjelobrojnu vrijednosti te utor *setValue* i signal *valueChanged*. Cilj je omogućiti kada se promjeni vrijednost atributa *value* koristeći utor, odnosno funkciju *setValue*, da se odašilje signal *valueChanged* koji označava da je došlo do promjene atributa *value*. U tablici 3.1. dan je kod klase *Counter*.

Tab. 3.1. Klasa *Counter* [20]

```
#include <QObject>

class Counter : public QObject
{
   Q_OBJECT

public:
   Counter() { m_value = 0; }
   int value() const { return m_value; }

public slots:
   void setValue(int value);

signals:
   void valueChanged(int newValue);

private:
   int m_value;
};
```

U tablici 3.2. dan je kod funkcije *setValue*. Ključnom riječju *emit* se označava koji signal će se odašiljati nakon izvršavanja te funkcije.

Tab. 3.2. Funkcija *setValue* [20]

```
void Counter::setValue(int value)
{
    if (value != m_value) {
        m_value = value;
        emit valueChanged(value);
    }
}
```

U tablici 3.3. dan je isječak iz koda gdje se vrši povezivanje signala *valueChanged* s utorom *setValue*. Koristi se ključna riječ *connect* i povezuje se signal objekta *one* s utorom objekta *two*.

Tab. 3.3. Povezivanje signala s utorom [20]

```
Counter one, two;
QObject::connect(&one, SIGNAL(valueChanged(int)), &two, SLOT(setValue(int)));
```

QtQuick modul je jednostavan alat za dizajniranje korisničkog sučelja. On pruža sve potrebne klase za razvoj korisničkog sučelja. Kod za izgled sučelja se piše u *Qt Modeling Language* (QML), a to je opisni jezik koji koristi *Javascript* programski jezik. Ključnom riječju *function* pišu se pojedine funkcije koje se kasnije mogu pozvati određenim događajem, npr. pritiskom na određeni gumb. Kompletan QML kod se u pozadini prevodi u C++. Programi rađeni u Qt-u se ne moraju nužno pisati ili u C++ ili u QML-u već se mogu kombinirati. Moguće je stvoriti C++ objekte sa svojstvima i metodama i onda ih pozivati i koristiti unutar QML koda. Qt je dizajniran tako da je moguće proširiti jedan dio s drugim [20].

Mnoge tvrtke koriste Qt za razvoj svojih proizvoda. LG je jedna od vodećih svjetskih tvrtki koja proizvodi mnoge uređaje, od televizora, mobilnih uređaja, klima uređaja, hladnjaka sve do automobilskih informativno-zabavnih sustava (engl. *infotainment system*). LG se odlučio koristiti Qt za razvoj vlastitog WebOS operativnog sustava koji se koristi u pametnim TV uređajima i pametnim satovima. Također koriste Qt kao podlogu za upravljačke programe na mnogim drugim uređajima.

Panasonic Avionics Corporation je korporacija koja dizajnira, prodaje i ugrađuje sustave za zabavu i komunikaciju u avionima. Te sustave su izradili pomoću Qt-a koristeći C++ i QML i danas preko 5 milijuna ljudi dnevno koristi te sustave tijekom leta.

Qt Automotive Suite je komplet alata za izradu programske podrške za automobilske sustave. Rimac automobili su razvili vlastiti program pomoću Qt-a za prikazivanje velike količine informacija dobivenih sa senzora, upravljanje svim dijelovima automobila i jedinstveni sustav vektoriranja zakretnog momenta svih kotača (engl. *all wheel torque vectoring system*). Poput Rimac automobila, Mercedes-Benz i Tesla također koriste Qt za razvoj vlastitih informativno-zabavnih sustava u automobilima.

AMD, jedna od najvećih svjetskih tvrtki za proizvodnju grafičkih kartica i procesora, je razvio koristeći Qt svoj upravljački program za grafičke kartice zvan *Radeon Software Crimson Edition*. Odabrali su Qt upravo zbog mogućnosti razvoja za više platformi i brzog rada [21].

Za izradu rješenja u sklopu ovog diplomskog rada također se koristio Qt, a razlozi njegova odabira najbolje su vidljivi u tekstu opisanom ranije.

4. PROGRAMSKO RJEŠENJE ZA PRIKAZ VREMENSKE SKALE I SNIMLJENOG VIDEO SADRŽAJA S GREŠKAMA UOČENIM PRI ANALIZI VIDEO ZAPISA

4.1. Zadatak koji je bilo potrebno riješiti

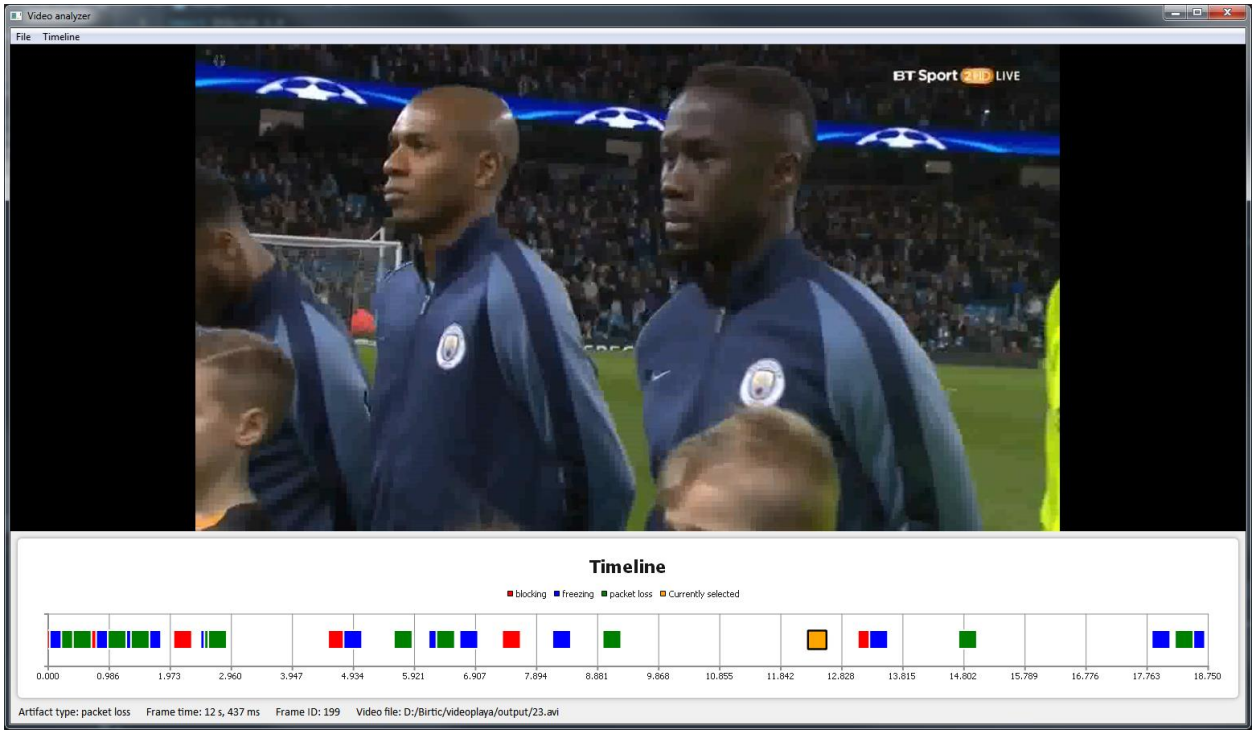
Zadatak je bio napraviti rješenje koje će korisniku omogućiti da na osnovu prethodne obrade video zapisa prikaže vremensku skalu s oznakama artefakata koji su se pojavili i omogući reprodukciju isječka video zapisa u kojem je došlo do pojave artefakta. Prethodnu obradu video zapisa vrši PQM koji otkriva koji artefakti su se pojavili, sprema podatke o njima u zapisnu datoteku i stvara isječke iz obrađenog video zapisa za svaki pojedini artefakt. Zapisna datoteka (engl. *log file*) spremljena je u JSON formatu. U nju su spremljeni podaci o početku i kraju obrade, ukupnom broju okvira, broju okvira po sekundi, svim vrstama artefakata koji se pojavljuju i pojedinačne informacije o svakom artefaktu. To su tip artefakta, redni broj okvira u kojem se on pojavljuje i naziv video isječka u kojem se artefakt pojavljuje. Primjer zapisne datoteke je dan u tablici 4.1.

Tab. 4.1. Primjer sadržaja zapisne datoteke

{
"start_time":"2017-07-11T11:24:41",
"end_time":"2017-07-11T11:25:41",
"number_of_frames":"40“,
„fps“:“16“,
„artifact_types“:[„blocking“,“freezing“,“packet loss“],
„artifacts“:[
{„type“:“blocking“,“frame_id“:“0“,“video_file“:“0.avi“},
{„type“:“freezing“,“frame_id“:“3“,“video_file“:“1.avi“},
{„type“:“packet loss“,“frame_id“:“6“,“video_file“:“2.avi“},
{„type“:“packet loss“,“frame_id“:“9“,“video_file“:“3.avi“},
{„type“:“blocking“,“frame_id“:“12“,“video_file“:“4.avi“},
{„type“:“freezing“,“frame_id“:“15“,“video_file“:“5.avi“},
{„type“:“packet loss“,“frame_id“:“18“,“video_file“:“6.avi“},
{„type“:“freezing“,“frame_id“:“21“,“video_file“:“7.avi“},
{„type“:“packet loss“,“frame_id“:“24“,“video_file“:“8.avi“},

{„type“:“freezing“,“frame_id“:“27“,“video_file“:“9.avi“},
{„type“:“blocking“,“frame_id“:“35“,“video_file“:“10.avi“}
]
}

Najprije korisnik odabire zapisnu datoteku koju želi učitati. Nakon odabira program čita odabranu datoteku i popunjava vremensku skalu s oznakama za svaki pojedini artefakt. Korisniku je omogućeno da odabere bilo koji artefakt i onda se pokreće reprodukcija video isječka na kojemu se dogodio taj artefakt. Svaki video isječak traje u ovisnosti o trajanju artefakta. Video reprodukciju je moguće zaustaviti i pretraživati. Moguće je također uvećati (engl. *zoom in*) i umanjiti (engl. *zoom out*) vremensku skalu. Kada je vremenska skala uvećana, korisnik može pomicati skalu ulijevo ili udesno. Na slici 4.1. može se vidjeti kako programsko sučelje izgleda prilikom reprodukcije video isječka.



Sl. 4.1. Sučelje programskog rješenja tijekom rada s odabranim video zapisom

4.2. Detalji rada programskog rješenja

Bitno je napomenuti da je prije pokretanja rješenja iz ovog diplomskog rada potrebno izvršiti obradu oštećenog video zapisa pomoću PQM-a kako bi se stvorila zapisna datoteka. Nakon pokretanja rješenja, jedino je moguće odabrati učitavanje zapisne datoteke. Odabir datoteke riješen je u QML dijelu programa. Otvara se skočni prozor koji omogućava odabir datoteke. Kada se odabere odgovarajuća datoteka, program čita podatke iz nje, sprema ih te ih prikazuje na vremenskoj skali. Učitavanje datoteke JSON formata razvijeno je koristeći C++ programski jezik. Qt pruža jednostavnu funkciju za čitanje JSON datoteke. Primjer čitanja dan je u tablici 4.2.

Tab. 4.2. Primjer koda za čitanje JSON datoteke

```
 QFile file(path);

QFile filejson(path);
filejson.open(QIODevice::ReadOnly);

QByteArray rawData = filejson.readAll();
JsonDocument doc(QJsonDocument::fromJson(rawData));

QJsonObject json = doc.object();
frames = json["frames"].toString();

QJsonValue types = json.value("types");
QJsonArray typeArray = types.toArray();
```

Čitav sadržaj datoteke se sprema u polje iz kojeg se potom mogu čitati podaci koji su potrebni. Funkcija za čitanje JSON datoteke poziva se iz QML dijela programa nakon što je odabrana datoteka. Kada funkcija za čitanje JSON datoteke završi, poziva se funkcija za postavljanje pročitanih vrijednosti na vremensku skalu. Vrijednosti se postavljaju kao oznake u obliku raznobojnih kvadratića na vremensku skalu. Oni predstavljaju svaki pojedini artefakt i kronološki su poredani. Kada su svi podaci uspješno postavljeni, program čeka na daljnju korisnikovu radnju.

Korisnik pritiskom lijevog gumba miša može odabrati bilo koji kvadratić. Time se pokreće reprodukcija video isječka tog artefakta i mijenja se obrub i boja kvadratića da bi bio istaknut. Prelaskom pokazivača miša preko videa, gumbovi za upravljanje reprodukcijom videa postaju vidljivi. Video je moguće pokrenuti, zaustaviti i premotavati.

Vremenska skala je napravljena u QML-u kao raspršeni graf (engl. *scatter chart*). Na x-osi se prikazuje vrijeme, dok je y-os skrivena. Podaci se dodaju u tzv. raspršene serije (engl. *scatter series*). U ovom slučaju, svaka serija predstavlja jednu vrstu artefakta i obojana je jednom bojom. Ako program pročita da u zapisnoj datoteci postoje samo primjerice dvije vrste artefakata, onda će podatke dodati u samo dvije raspršene serije. Podaci se dodaju u serije kao točke. Svaka točka je prikazana kao kvadratić i ima svoju vremensku vrijednost. Budući da je pozadina QML-a *Javascript*, vrijeme se računa kao broj milisekundi koji je prošao od 1.1.1970. Da bi se dobile tražene x vrijednosti svake točke, program računa vrijeme od početka mjerenja i od svake x vrijednosti oduzima to vrijeme. Y vrijednosti svake točke su identične zato da bi bile u istom redu na vremenskoj skali.

Svaka serija ima funkciju koja prati događaje s miša. Za pritisak lijevog gumba miša vrši se isticanje tog kvadratića drugom bojom i pokretanje reprodukcije video isječka. Budući da sama točka ne sadrži nikakve druge vrijednosti osim x i y koordinata, program izračunava redni broj artefakta tako što od x vrijednosti te točke oduzme vrijeme početka mjerenja i taj zbroj pomnoži s brojem okvira po sekundi. Video isječci spremljeni su pod rednim brojevima tako da im se lako može pristupiti. Prelaskom pokazivača miša preko kvadratića ažuriraju se podaci o tom kvadratiću i prikazani su na dnu prozora.

Dvostrukim pritiskom lijevog gumba miša na kvadratić korisnik može uvećati vremensku skalu i postaviti taj kvadratić prema sredini skale. Pomicanjem kotačića miša vremenska skala se može uvećavati i umanjivati. Držanjem lijevog gumba pritisnutim i povlačenjem pokazivača miša ulijevo ili udesno korisnik može pomicati skalu prema lijevo ili desno. Vremenska skala se ne može pomaknuti izvan vremenskog okvira pročitanog iz zapisne datoteke i zato je moguće pomicati skalu samo kada je uvećana. Uvećavanje i pomicanje vremenske skale je napravljeno pomoću nekoliko posebnih funkcija. Svaka radnja ima svoju funkciju, no one rade vrlo slično.

Tab. 4.3. Primjer funkcije za pomicanje ulijevo

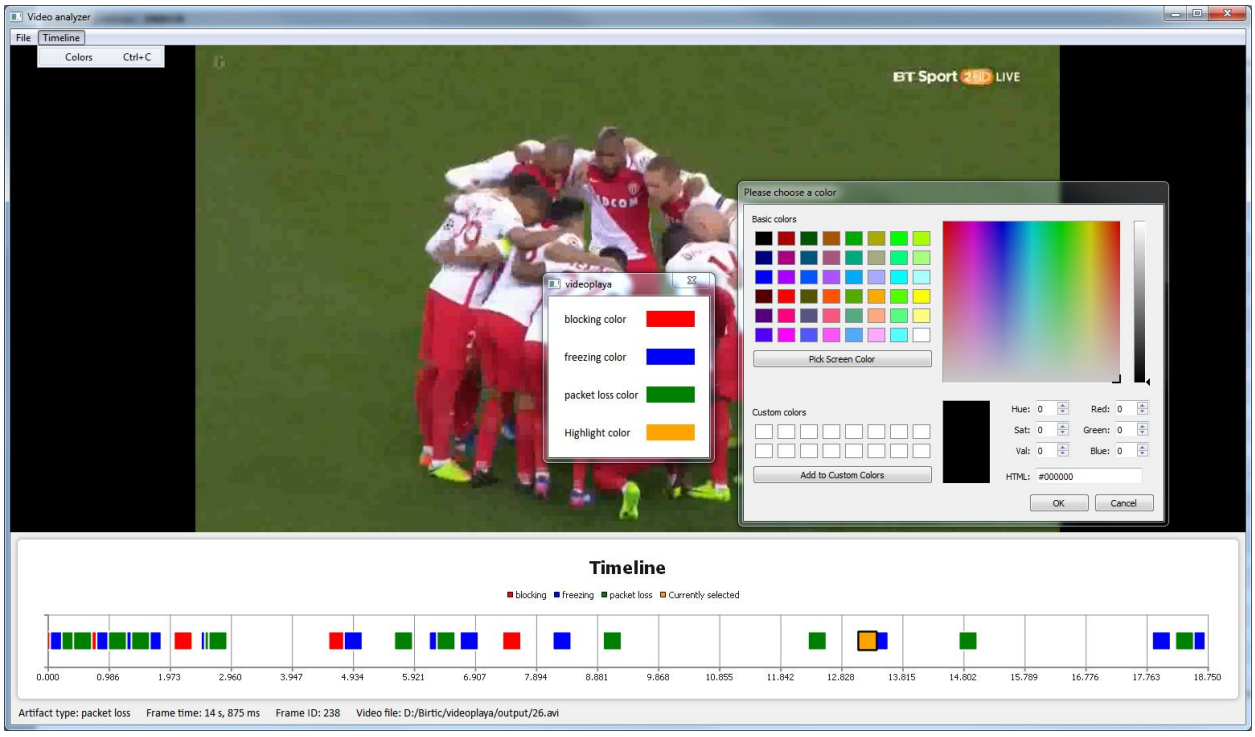
```
function scrollLeftFor(distance) {
        if(xAxis.min.getTime() !== defaultTime)
            scatterChart.scrollLeft(distance)

        if(xAxis.min.getTime() < defaultTime)
            xAxis.min = xAxis.axisMin
    }
```

Tab. 4.4. Primjer funkcije za uvećavanje

```
function zoomFor(value) {
        if(value > 0)
            scatterChart.zoom(value)
    }
```

U tablici 4.3. dan je primjer funkcije za pomicanje vremenske skale ulijevo. Funkcija prima broj koji predstavlja udaljenost u elementima slike za koju se pomiče vremenska skala. Prvo se provjerava je li minimum vremenske skale na nuli. Ako je, ništa se ne događa, a ako nije onda se vrši pomicanje. Druga provjera služi da bi se minimum vratio na početnu vrijednost u slučaju da se pomakne izvan granica koje su pročitane iz zapisne datoteke. Za samo pomicanje ulijevo po skali koristi se funkcija *scrollLeft* koja je ugrađena u objekt raspršenog grafa. Za pomicanje udesno je ista logika, samo se koristi funkcija *scrollRight*. U tablici 4.4. dan je primjer funkcije za uvećavanje. Funkcija prima broj koji predstavlja faktor uvećanja, odnosno umanjenja. Taj faktor mora biti broj veći od nula. Ako je manji od jedan onda se vremenska skala umanjuje, a ako je veći od jedan, onda se uvećava. Za samo uvećavanje i umanjivanje koristi se funkcija *zoom* koja je ugrađena u objekt raspršenog grafa. Pritiskom desnog gumba miša vremenska skala se vraća u izvorni oblik, odnosno minimum i maksimum se vraćaju na početne vrijednosti.



Sl. 4.2. Izbornik za mijenjanje boja

Unutar izbornika *Timeline* postoji mogućnost za promjenu boja kvadratića na vremenskoj skali. Prvo se otvara prozorčić u kojemu se s lijeve strane nalaze nazivi artefakata, a s desne obojani pravokutnici. Pritiskom lijevog gumba miša na bilo koji od tih pravokutnika, otvara se novi prozor u kojemu je moguće odabrati bilo koju boju. Promjena boja se vrši tako da se za pojedinu raspršenu seriju mijenja svojstvo boje. Do promjene boje dolazi nakon pritiska gumba *OK* unutar izbornika za boje. Na slici 4.2. dan je snimak iz programa prilikom mijenjanja boja na vremenskoj skali.

Tijekom rada programa moguće je odabrati nove zapisne datoteke. Program će učitati novu datoteku, obrisati sve točke u postojećim raspršenim serijama i dodati u njih nove podatke.

Dio za reprodukciju video isječaka je napravljen u QML-u koristeći *QMultimedia* modul. Iz tog modula korišten je *Video* objekt kojemu su postavljena svojstva poput širine, visine, pozicije i dr. Svojstvu *source* se treba predati putanja na računalu do video isječka i reprodukcija se može pokrenuti. Ovo se svojstvo svaki put mijenja kada korisnik pritisne neku od oznaka na vremenskoj skali. U tablici 3.5. dan je primjer koda za stvaranje video objekta.

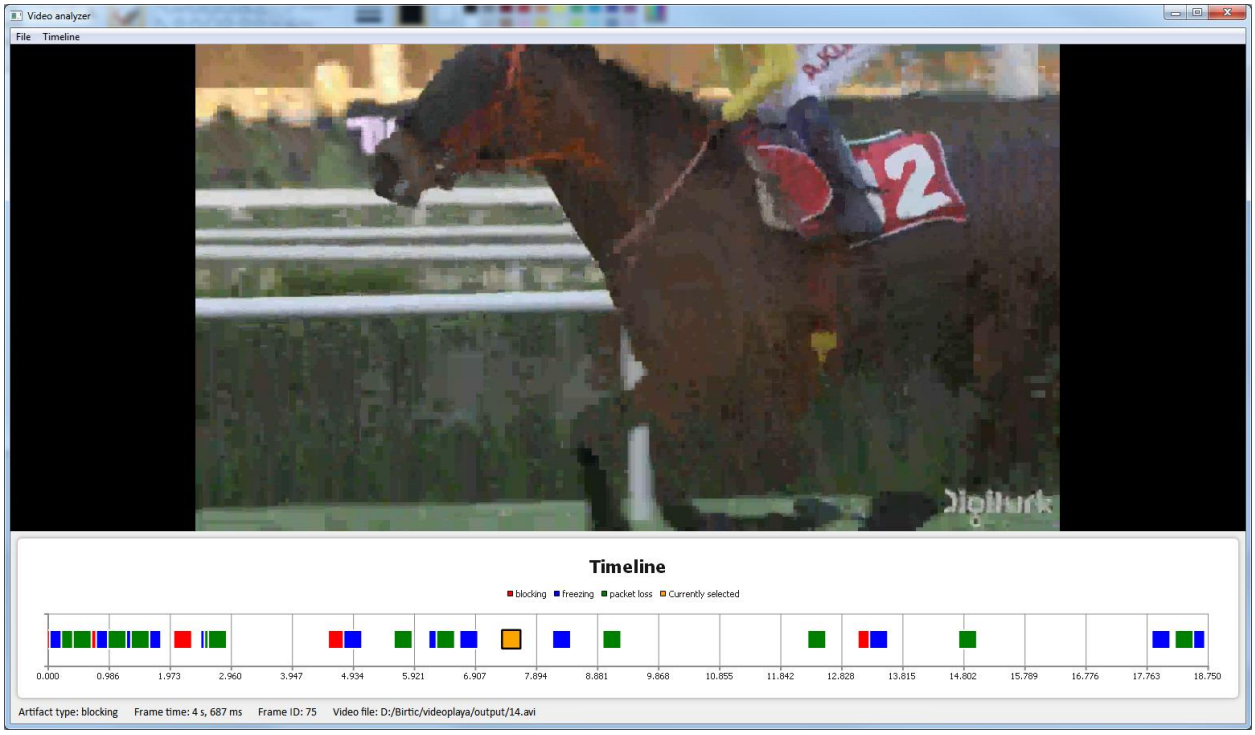
Tab. 4.5. Primjer koda za video objekt

```
Video {
    id: video
    source: path
    width: 1280
    height: 720
    fillMode: VideoOutput.Stretch
  }
```

5. TESTIRANJE FUNKCIONALNOSTI PROGRAMSKOG RJEŠENJA

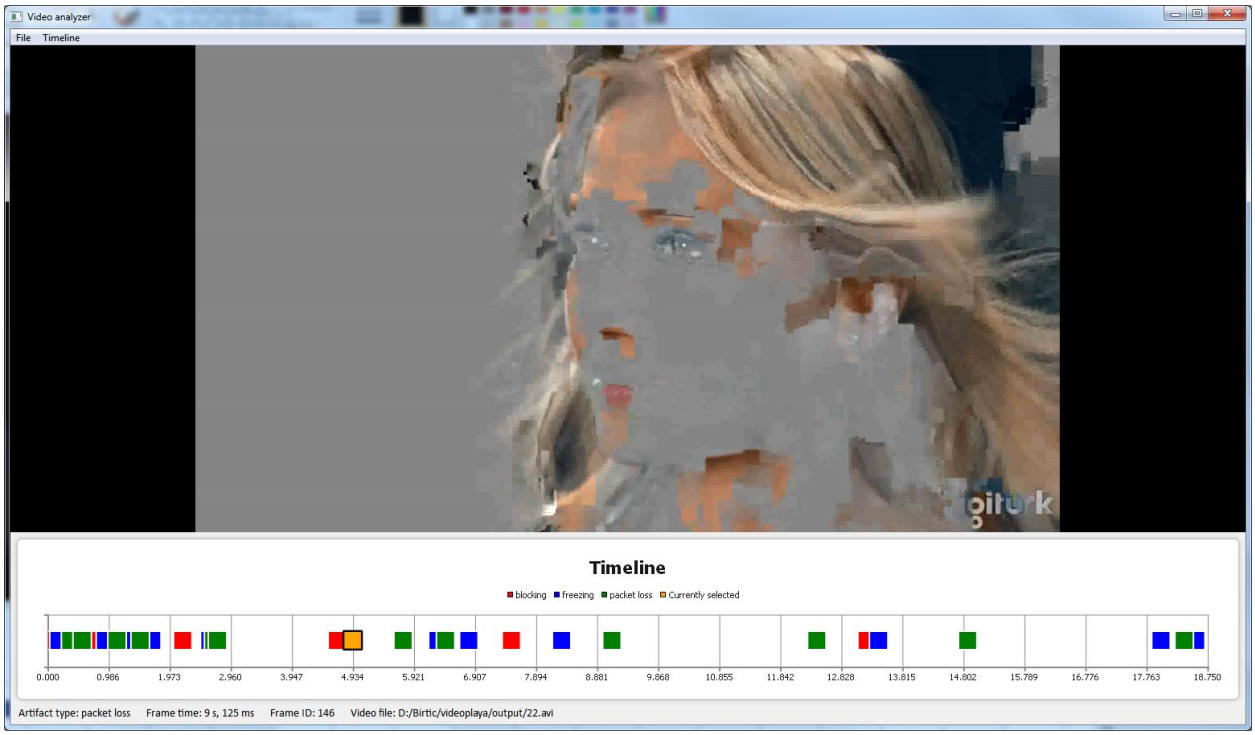
5.1. Provođenje testiranja

Programsko rješenje je testirano koristeći tri različite zapisne datoteke naziva „a“, „b“ i „c“. Prva datoteka, odnosno datoteka „a“ sadrži zapis o 10 artefakata podijeljenih u tri vrste s vremenskim rasponom od 20 sekundi, druga datoteka, odnosno datoteka „b“ sadrži 30 artefakata podijeljenih također u tri vrste s vremenskim rasponom od šest minuta, a treća datoteka, odnosno datoteka „c“ sadrži zapis o 10 artefakata podijeljenih u dvije vrste s vremenskim rasponom od 24 sata. Testirana je točnost čitanja zapisnih datoteka i prikazivanja ispravnih video isječaka te radnje uvećavanja, umanjivanja i pomicanja po vremenskoj skali. Na slici 5.1. dan je snimak programa u radu nakon učitavanja zapisne datoteke „a“ pri reprodukciji video isječka s artefaktom pojave blokova.



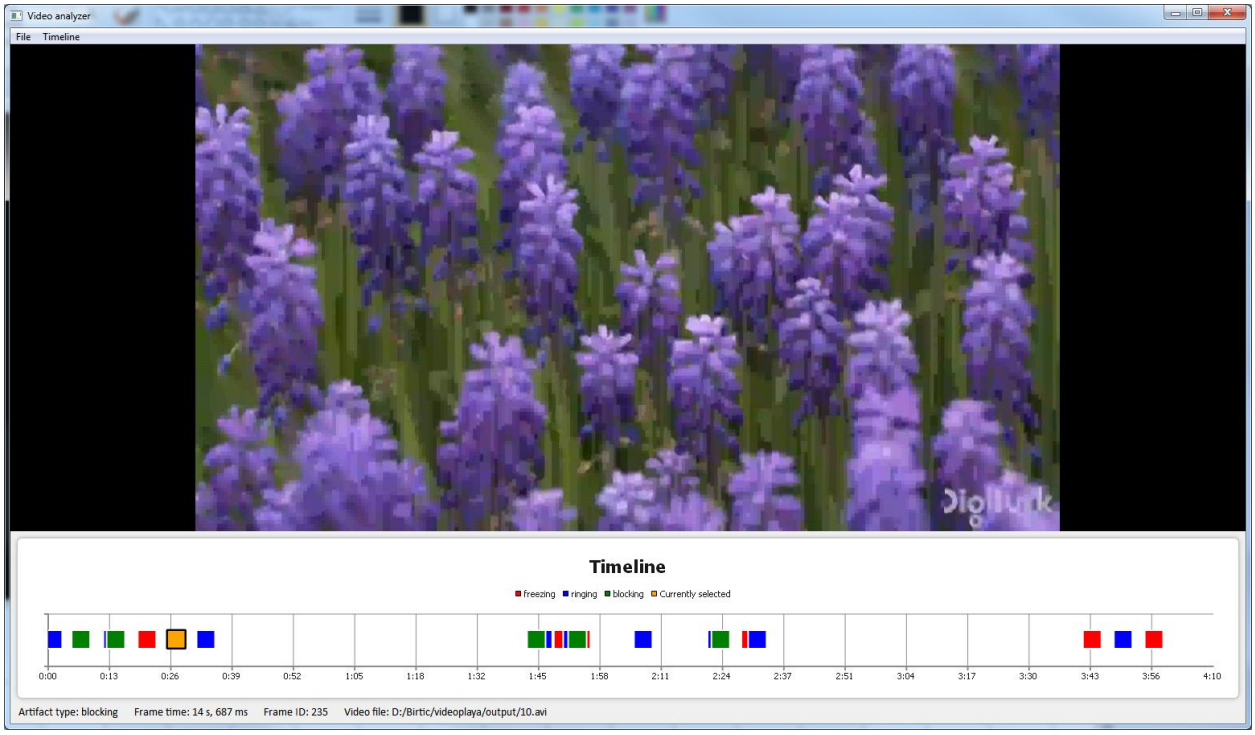
Sl. 5.1. Testiranje programskog rješenja s zapisnom datotekom „a“

Na slici 5.2. dan je primjer rada rješenja prilikom reprodukcije video isječka s artefaktom gubitka paketa.



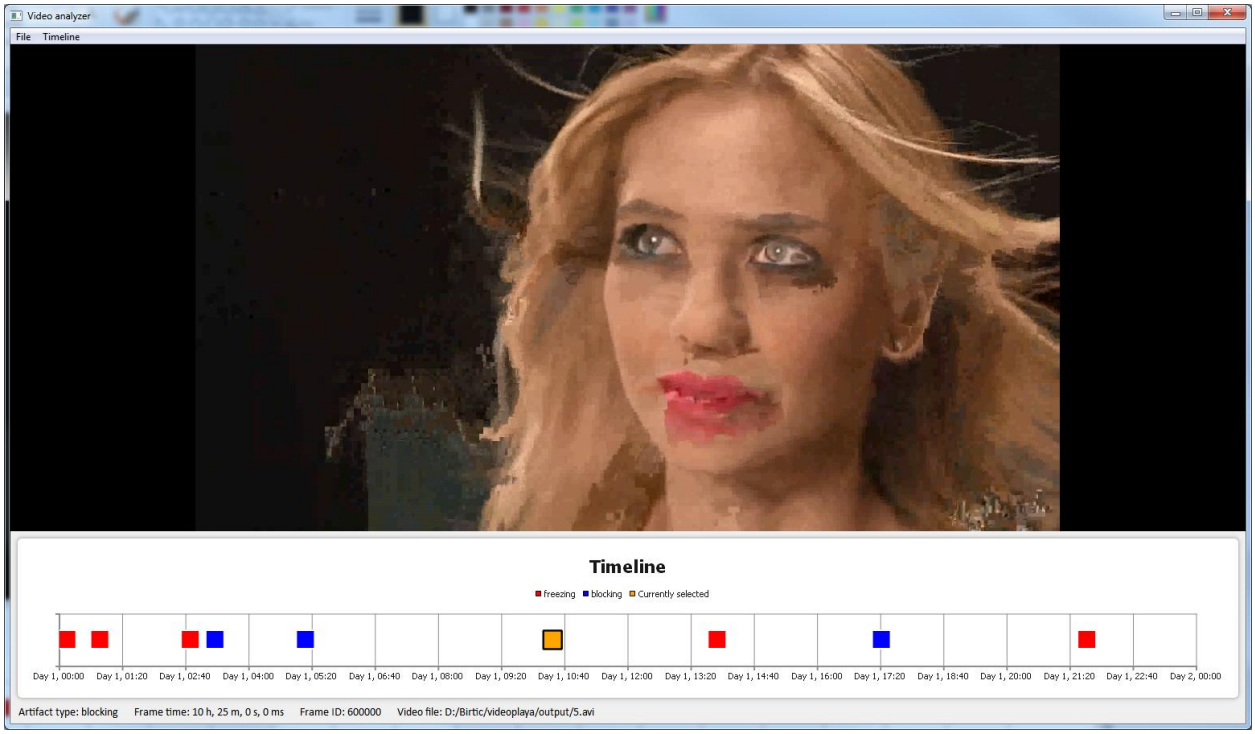
Sl. 5.2. Testiranje prikazivanja video isječka s artefaktom gubitka paketa

Na slici 5.3. dan je snimak rješenja tijekom rada nakon učitavanja zapisne datoteke „b“.

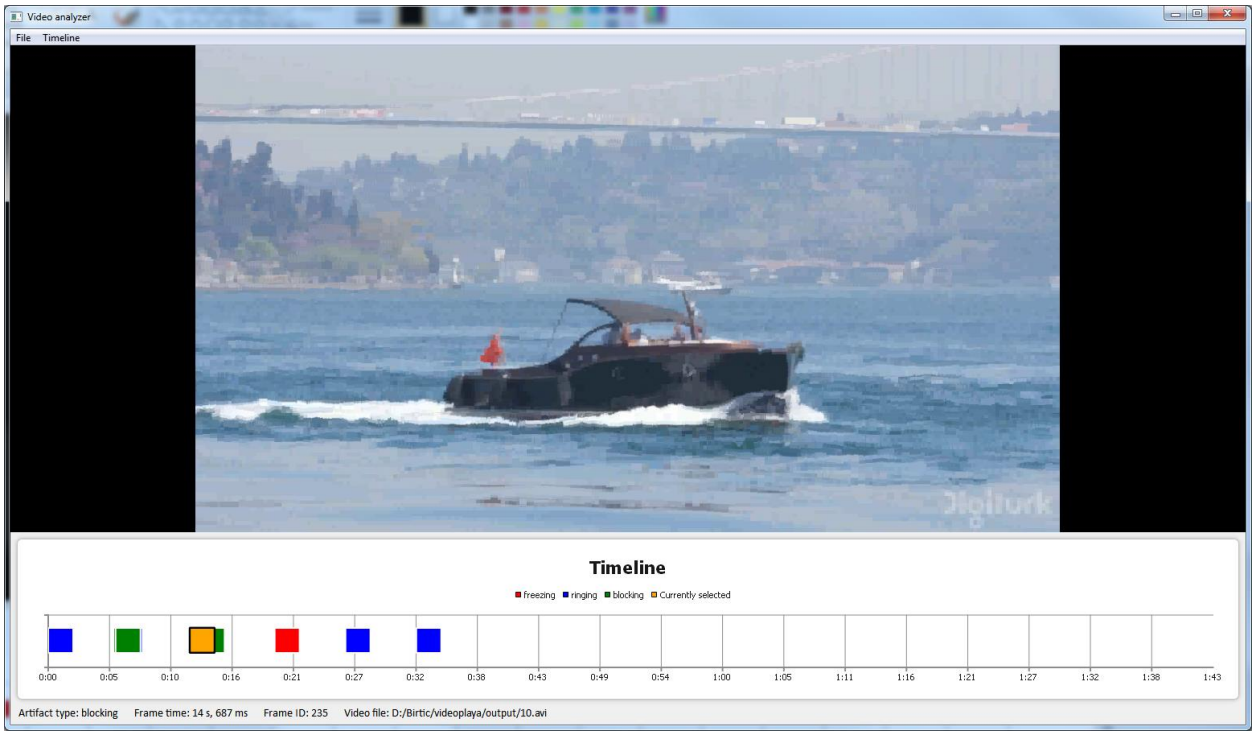


Sl. 5.3. Testiranje rada nakon učitavanja zapisne datoteke „b“

Na slici 5.4. dan je snimak rada rješenja nakon učitavanja zapisne datoteke „c“. Na video isječku je prikazan artefakt gubitka paketa, a vremenska skala prikazuje vremenski raspon od 24 sata. Može se uočiti da vremenski zapis na vremenskoj skali prikazuje dane, sate i minute.



Sl. 5.4. Testiranje rada nakon učitavanja zapisne datoteke „c“



Sl. 5.5. Testiranje uvećavanja i pomicanja

Slika 5.5. prikazuje snimak rješenja tijekom rada gdje su testiranje mogućnosti uvećavanja i pomicanja po vremenskoj skali. Vremenska skala je uvećana tri puta i pomaknuta sasvim ulijevo.

Učitavanje zapisne datoteke i postavljanje vrijednosti na vremensku skalu odvija se trenutno u sva tri slučaja. Ovisno o vremenskom rasponu, vremenska skala prilagođava granice da sve oznake stanu unutar nje. Nije bilo problema prilikom reprodukcije video isječaka niti u jednom slučaju. Također, radnje uvećavanja, umanjivanja, pomicanja ulijevo ili udesno i mijenjanja boje oznaka su uredno radile u sva tri slučaja.

Programsko rješenje je razvijeno na *Windows* platformi i testirano na računalu s *Windows* 7 operativnim sustavom, procesorom Intel *Core* i7 4790 i osam GB radne memorije (engl. RAM – *random access memory*). Tijekom rada program zauzima oko 130 MB radne memorije i opterećuje procesor od pet do 20 %.

Testiranjem je utvrđeno da program radi bez greške i zadovoljavajućom brzinom u svakom testnom slučaju. Nije dolazilo do nestabilnosti ili rušenja programa.

5.2. Nedostaci programskog rješenja

Programsko rješenje obavlja sve tražene funkcije. Nema velikih nedostataka ili problema. Jedno od poboljšanja koje bi se moglo napraviti je prilagođavanje animacija na vremenskoj skali da izgledaju glađe prilikom uvećavanja, umanjivanja i pomicanja.

6. ZAKLJUČAK

Video zapisi su podložni mnogim greškama. Većina grešaka se stvara zbog sažimanja, no postoje i one do kojih dolazi tijekom prijenosa. Neke greške je moguće ispraviti. Zato se vrši obrada video zapisa da bi se utvrdilo gdje i kada je došlo do greške. Tek onda se može raditi na njihovom ispravljanju ili sprječavanju njihove pojave.

Cilj ovog diplomskog rada bio je izraditi programsko rješenje koje će pomoći pri obradi video zapisa tako što će omogućiti korisniku da ima pregled trenutaka u kojima je došlo do greške i mogućnost pregledavanja istih. Programsko rješenje prikazuje vremensku skalu na kojoj se nalaze oznake koje predstavljaju trenutke u kojima je došlo do greške i ima mogućnosti pregledavanja video isječaka s greškama, uvećavanja, umanjivanja i pomicanja po vremenskoj skali. Programsko rješenje je razvijeno u programskom okruženju Qt koristeći programske jezike C++ i QML. Testiranje je izvršeno s tri različite zapisne datoteke i program radi bez greške i zadovoljavajućom brzinom.

Zadatak je u potpunosti izvršen i programsko rješenje ima sve tražene mogućnosti. Ono će poslužiti korisniku kao alat za pregledavanje rezultata prethodno odrađene analize postojanja izobličenja u video zapisu.

LITERATURA

[1] HiDEF, http://hidefnj.com/video.html, pristupio: 28.8.2017.

[2] C. Jewitt, MODE node, Institute of Education, London, ožujak 2012.,

http://eprints.ncrm.ac.uk/2259/4/NCRM_workingpaper_0312.pdf, pristupio: 29.8.2017.

[3] *Interlaced video*, https://en.wikipedia.org/wiki/Interlaced_video, pristupio: 28.8.2017.

[4] W. F. Schreiber, Prof. *Emeritus of Electrical Engineering*, MIT, http://www.vxm.com/Progvsinter.html, pristupio: 28.8.2017.

[5] Red, http://www.red.com/learn/red-101/video-chroma-subsampling, pristupio: 29.8.2017.

[6] *Chroma subsampling*, https://en.wikipedia.org/wiki/Chroma_subsampling, pristupio: 29.8.2017.

[7] C. Ponyton, *Digital video and HDTV algorithms and interfaces*, 2003., http://poynton.ca/PDFs/Chroma_subsampling_notation.pdf, pristupio: 29.8.2017.

[8] *Estructura GOP*, https://ca.wikipedia.org/wiki/Estructura_GOP, pristupio: 30.8.2017.

[9] Apple, 2010., https://documentation.apple.com/en/finalcutpro/usermanual/index.html#chapter=C%26section=12%26tasks=true, pristupio: 30.8.2017.

[10] MPEG-1, http://mpeg.chiariglione.org/standards/mpeg-1/video, pristupio: 31.8.2017.

[11] MPEG-2, http://mpeg.chiariglione.org/standards/mpeg-2/video, pristupio: 31.8.2017.

[12] MPEG-4, http://mpeg.chiariglione.org/standards/mpeg-4/video, pristupio: 31.8.2017.

[13] HEVC, http://mpeg.chiariglione.org/standards/mpeg-h/high-efficiency-video-coding, pristupio: 31.8.2017.

[14] J. Urban, 16.2.2017., http://blog.biamp.com/understanding-video-compression-artifacts/, pristupio: 1.9.2017.

[15] P.-T. Le Dinh i J. Patry, Algolith, 24.2.2006., http://www.embedded.com/print/4013028, pristupio: 1.9.2017.

[16] K. Zeng, T. Zhao, A. Rehman i Z. Wang, *Dept. of Electrical & Computer Engineering*, *University of Waterloo*, Waterloo, ON, Kanada, 6.2.2014., https://ece.uwaterloo.ca/~z70wang/publications/HVEI14.pdf, pristupio: 1.9.2017.

[17] RT RK, http://bbt.rt-rk.com/audiovideo-analysis/, pristupio 18.9.2017.

[18] RT RK, http://bbt.rt-rk.com/hardware/rt-av_station/, pristupio 19.9.2017.

[19] The Qt Company, http://wiki.qt.io, pristupio: 4.9.2017.

[20] The Qt Company, http://doc.qt.io/, pristupio: 5.9.2017.

[21] The Qt Company, https://www.qt.io/, pristupio: 5.9.2017.

[22] Sony, https://pro.sony.com/bbsccms/assets/files/micro/xdcam/solutions/Sony_HD_Formats_Guide.pdf, pristupio: 28.8.2017.

SAŽETAK

U ovom diplomskom radu riješen je problem prikazivanja vremenske skale i snimljenog video sadržaja s greškama uočenim pri obradi video zapisa. Video zapis je niz pokretnih slika prikazanih u vrlo brzom slijedu. Problem je u tome što su izvorni video zapisi jako veliki pa ih je potrebno sažeti. Najčešće korištene tehnike sažimanja su poduzorkovanje boja i metoda skupine slika. Zbog sažimanja video zapisa javljaju se mnoge greške poput stvaranja blokova, izlijevanja boja, zvonjave i dr.

Programsko rješenje je napravljeno u razvojnom okruženju Qt. Qt omogućava razvoj programskih rješenja za više različitih platformi koristeći programske jezike C++ i QML. Dio koda koji služi za čitanje zapisne datoteke i spremanje pročitanih podataka je napisan u programskom jeziku C++, a preostali dio koda je napisan u QML-u. Program sadrži vremensku skalu s oznakama za svaki artefakt i omogućuje korisniku reprodukciju video isječaka s odgovarajućim artefaktima. Također ima mogućnosti uvećavanja, umanjivanja i pomicanja po vremenskoj skali. Testiranje je pokazalo da program ispravno radi u svakom od testnih slučajeva i da je brzina izvođenja zadovoljavajuća.

Ključne riječi: video zapis, sažimanje, greške, Qt, vremenska skala

SOLUTION FOR DISPLAYING THE TIMELINE AND RECORDED VIDEO CONTENT WITH ARTIFACTS DETECTED DURING VIDEO ANALYSIS

ABSTRACT

This master's thesis solves the problem of displaying the timeline and saved video content with artifacts detected during video analysis. Video is a sequence of moving pictures that are displayed very fast. The problem is that uncompressed videos are very big so they need to be compressed. The most common compression techniques are chroma subsampling and group of pictures. Because of video compression many artifacts appear like blocking, color bleeding, ringing and others.

The program was developed using the Qt development environment. Qt enables development of applications for multiple different platforms using C++ and QML programming languages. Part of the code that reads the log file and saves the data was written in the C++ programming language while the rest of the code was written in QML. The program contains a timeline with markers for each artifact and it allows the user to play video cuts with corresponding artifacts. Also, it provides the functionalities of zooming in, zooming out and scrolling the timeline. The testing has shown that the program is working correctly in each of the test cases and its performance is satisfactory.

Key words: video, compression, artifacts, Qt, timeline

ŽIVOTOPIS

Kristijan Birtić je rođen u Osijeku, Republika Hrvatska, 1. listopada 1993. godine. Pohađao je osnovnu školu „Retfala“ u Osijeku i svih osam razreda prošao je s odličnim uspjehom. Od četvrtog razreda sudjelovao je svake godine na županijskim natjecanjima iz matematike, a u osmom razredu je sudjelovao i na županijskim natjecanjima iz engleskog jezika i kemije.

Nakon osnovne škole, 2008. godine upisuje prirodoslovnu-matematičku gimnaziju u Osijeku i završava prvi i treći razred s vrlo dobrim te drugi i četvrti s odličnim uspjehom. 2012. godine je maturirao s vrlo dobrim uspjehom.

2012. godine upisuje Elektrotehnički fakultet u Osijeku, smjer računarstvo. Preddiplomski studij završava 2015. godine i upisuje diplomski studij na istom fakultetu.

_____

Kristijan Birtić