Sinkronizacija videosignala i audiosignala

Livaja, Toni

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:200:456558

Rights / Prava: In copyright / Zaštićeno autorskim pravom.

Download date / Datum preuzimanja: **2024-03-28**

Repository / Repozitorij:

Faculty of Electrical Engineering, Computer Science and Information Technology Osijek

zir.nsk.hr

Image not found or type unknown

SVEUČILIŠTE JOSIPA JURJA STROSSMAYERA U OSIJEKU

FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA I INFORMACIJSKIH TEHNOLOGIJA

Sveučilišni studij

Sinkronizacija videosignala i audiosignala

Diplomski rad

Toni Livaja

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:	Toni Livaja
Studij, smjer:	Diplomski sveučilišni studij Računarstvo
Mat. br. studenta, godina upisa:	D 793 R, 09.10.2015.
OIB studenta:	88793645303
Mentor:	Izv. prof. dr. sc. Marijan Herceg
Sumentor:	
Sumentor iz tvrtke:	Danijel Babić
Predsjednik Povjerenstva:	Doc.dr.sc. Ratko Grbić
Član Povjerenstva:	Doc.dr.sc. Mario Vranješ
Naslov diplomskog rada:	Sinkronizacija videosignala i audiosignala
Znanstvena grana rada:	**Obradba informacija (zn. polje računarstvo)**
Zadatak diplomskog rada:	Vrlo čest problem koji se javlja kod reprodukcije video sadržaja je nesinkroniziranost slike i tona pa se u području testiranja ovom problemu posvećuje posebna pažnja. HbbTV (engl. Hybrid Broadcast Broadband TV) specifikacija definira specijalizirane funkcije za provjeru sinkronizacije audio i video sadržaja, pri čemu se sama provjera oslanja na posebne hardverske komponente. Cilj ovog rada je razvoj softverskog rješenja koje bi funkcioniralo u skladu sa specifikacijom, a bilo bi u stanju provjeru sinkronizacije obaviti izvan realnog vremena na osnovu video snimke izvršenja testa na uređaju. Video snimak može poticati s kamere ili grabber uređaja. (sumentor: Danijel Babić, Institut RT-RK Osijek d.o.o., Cara Hadrijana 10b, Osijek)
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:
	Datum:

FERIT
FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA
I INFORMACIJSKIH TEHNOLOGIJA OSIJEK

IZJAVA O ORIGINALNOSTI RADA

Osijek, 28.09.2017.

Ime i prezime studenta:	Toni Livaja
Studij:	Diplomski sveučilišni studij Računarstvo
Mat. br. studenta, godina upisa:	D 793 R, 09.10.2015.
Ephorus podudaranje [%]:	0

Ovom izjavom izjavljujem da je rad pod nazivom**: Sinkronizacija videosignala i audiosignala**

izrađen pod vodstvom mentora Izv. prof. dr. sc. Marijan Herceg

i sumentora

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. Teorijske osnove 2
2.1. HbbTV 2
2.2. Opis problema 4
2.3. Testna specifikacija 5
2.4. QR kod 8
3. Koncept rješenja 10
3.1. Opis modula 10
3.2. Princip rada 11
4. Testiranje i rezultati 19
4.1. Kreiranje testnih sekvenci 19
4.2. Testiranje aplikacije 20
4.3. Rezultati testiranja 21
4.3.1. Sinkronizirani video zapisi 22
4.3.2. Nesinkronizirani video zapisi 23
4.3.3. Video zapisi snimani u lošijim uvjetima 25
5. Zaključak 28
Literatura 29
Sažetak 30
Abstract 31
Životopis 32

1. Uvod

Tehnološki napredak u području digitalne televizije je doveo do nastanka „pametnih“ televizora (takozvani SmartTV) i televizijskih usluga koji koriste širokopojasni pristup Internetu (engl. *Broadband*) te ga kombiniraju s klasičnim odašiljanjem (engl. *Broadcast*). Ovakav hibridni pristup omogućuje gledanje klasičnog televizijskog sadržaja obogaćenog mnogim funkcionalnostima koje pruža pristup Internetu, kao što su poboljšani teletext, usluge videa na zahtjev, elektronski programski vodič, personalizacija usluge, igre, pristup društvenim mrežama i mnoge multimedijske usluge. Trenutno na tržištu postoji velik broj rješenja za hibridnu televiziju, što stvara dodatne troškove pružateljima usluge zbog potrebe za prilagodbom što većem broju uređaja, kao i samim korisnicima koji često moraju kupovati dodatnu opremu.

HbbTV konzorcij je udruženje koje razvija HbbTV standard (engl. *Hybrid Broadcast Broadband TV*). Cilj HbbTV konzorcija je osigurati široku prihvaćenost njihovog standarda što bi dovelo do međusobne kompatibilnosti televizijskih uređaja svih proizvođača, a kao posljedica toga bi došlo do jednostavnije i brže izrade programskih rješenja te smanjenja troškova. HbbTV definira sklopovsko i programsko okruženje za testiranje, a u sklopu postupka certifikacije proizvođači opreme i sadržaja moraju proći sve testove definirane u testnoj specifikaciji.

Jedan od testova je test sinkroniziranosti videosignala i audiosignala, a definiran je u testnoj specifikaciji [1] u sklopu *Media Synchronization* API-ja (engl. *Application Program Interface*). U sklopu ovog rada je napravljeno vlastito programsko rješenje koje na zadanim video zapisima provodi analizu sinkroniziranosti zvuka i slike te pronalazi odstupanja između njih, izražena u milisekundama.

2. Teorijske osnove

U ovom poglavlju je pojašnjen koncept HbbTV-a, tehnologije korištene u radu i testna specifikacija koju je bilo potrebno zadovoljiti.

2.1. HbbTV

HbbTV je naziv standarda koji kombinira klasični način odašiljanja i pristup internetu za isporuku sadržaja digitalne televizije, ali i udruženja koje razvija istoimeni standard. HbbTV konzorcij je nastao udruživanjem francuskog H4TV i njemačkog HTML profil projekta 2009. godine. 2014. godine se HbbTV spojio s Open IPTV (engl. *Internet Protocol Television*) Forumom, a 2016. je najavljeno spajanje sa Smart TV savezom. Standard je trenutno prihvaćen i u upotrebi u 32 države, najavljen je u 6 država (među njima je i Republika Hrvatska), a službeno je u postupku razmatranja u 30 država.



Slika 2.1.1. Službeni logo HbbTV konzorcija [2]

Ideja iza nastanka ovog standarda je omogućavanje proizvođačima programske podrške da isti kod mogu koristiti na različitim uređajima, a proizvođačima opreme da svoje uređaje ne moraju prilagođavati tržištu svake države ili usluzi svakog telekom operatera. Da bi se postigla što veća kompatibilnost, u HbbTV konzorciju sudjeluje više od 70 tvrtki i udruženja, a među njima se ne nalaze samo proizvođači opreme i programske podrške, nego i isporučitelji televizijskog sadržaja.

HbbTV kombinira pristup linearnom televizijskom programu i nelinearnom sadržaju kojem se pristupa putem Interneta. Osim gledanja videa, neke od najčešće korištenih funkcionalnosti HbbTV-a su elektronski programski vodič, povezivanje TV uređaja i pametnog telefona ili tableta, napredni teletekst, razni interaktivni sadržaji (npr. igre) i pretraživanje Interneta.



Slika 2.1.2. Izgled elektronskog programskog vodiča na HbbTV uređaju [3]

Iako je HbbTV globalna inicijativa, najveću podršku i prihvaćenost ima u Europi, stoga se certificira po ETSI (Europski institut za telekomunikacijske standarde) standardima. Trenutna službena verzija je HbbTV 2.0, pod ETSI standardom TS 102 796 v1.3.1. Iduća verzija će uključiti neke od funkcionalnosti koje se nalaze u MHEG-5 (engl. *Multimedia and Hypermedia Experts Group*) standardu. MHEG-5 je bio glavni konkurent HbbTV-u u Europi, ali je danas u raširenoj upotrebi samo u Ujedinjenom Kraljevstvu i Republici Irskoj, koji su već najavili budući prelazak na HbbTV. Razlozi njegove slabije prihvaćenosti su to što MHEG-5 koristi vlastiti programski jezik istog imena za izradu aplikacija koje se na njemu pokreću i zatvorenost standarda.

2.2. Opis problema

U sklopu certifikacije za HbbTV, potrebno je zadovoljiti sve testove koje propisuje testna specifikacija. Jedan od testova je test sinkroniziranosti audiosignala i videosignala. U sklopu testa se provjerava sinkroniziranost zvučnih signala i para bljeskova na određenom području ekrana. Svaki zvučni signal treba imati svoj par bljeskova i trajati jednako dugo, a moguće je malo odstupanje koje je prisutno zbog nesavršenosti mjernih uređaja. Svako veće odstupanje u vremenu detekcije zvučnog signala i para bljeskova upućuje na nesinkroniziranost, a ona može biti posljedica problema sa sklopovljem ili programskom podrškom uređaja na kojem se vrši reprodukcija sadržaja.

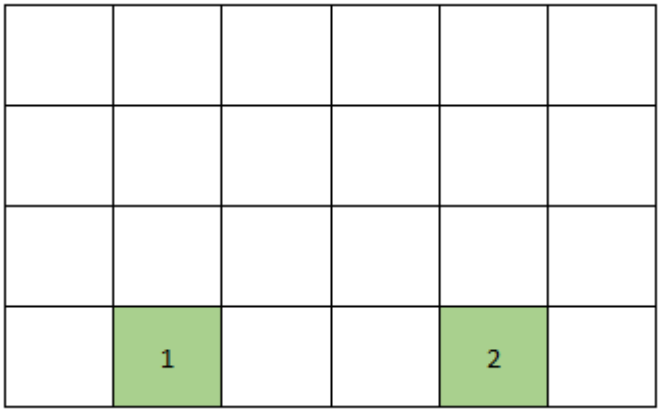
Postoji nekoliko metoda kojima se može testirati sinkroniziranost audiosignala i videosignala. Jedno od postojećih rješenja podrazumijeva lijepljenje mikrofona na zvučnike i lijepljenje fotosenzitivnih senzora na ekran uređaja na kojem se vrši reprodukcija video zapisa. Time se postiže visoka preciznost detekcije zvučnih signala i bljeskova, ali velik nedostatak je potreba za korištenjem mikrofona i senzora te njihovo lijepljenje za svaki uređaj koji se testira.

U sklopu ovog rada je razvijeno rješenje koje koristi samo kameru visoke razlučivosti za snimanje sadržaja koji se reproducira na uređaju koji se testira, a potom se snimljeni video zapis analizira na računalu. Zbog različite brzine kojom zvuk i svjetlost dolaze do kamere, kod ove metode su uvijek prisutna određena odstupanja između zvučnog signala i odgovarajućeg para bljeskova. Odstupanja su nekoliko milisekundi pa se mora zadati određena tolerancija na njih.

2.3. Testna specifikacija

Testna specifikacija definira video zapise na kojima se provodi testiranje, parametre funkcija i uvjete za prolazak testa. Sinkronizacija se vrši usporedbom vremena pojave audiosignala i bljeskova na slici koji se trebaju pojaviti u isto vrijeme i jednako trajati.

Specifikacijom je definirano da se na ekranu nalaze 2 kvadrata unutar kojih se pojavljuju bljeskovi. Prvi kvadrat se nalazi na prostoru između 1/6 i 1/3 ekrana po x-osi, a drugi od 2/3 do 5/6 ekrana po x-osi. Položaj po y-osi je jednak za oba i definiran je između 3/4 i 1, gdje 1 predstavlja dno ekrana.



Slika 2.3.1. Položaj bljeskova na ekranu

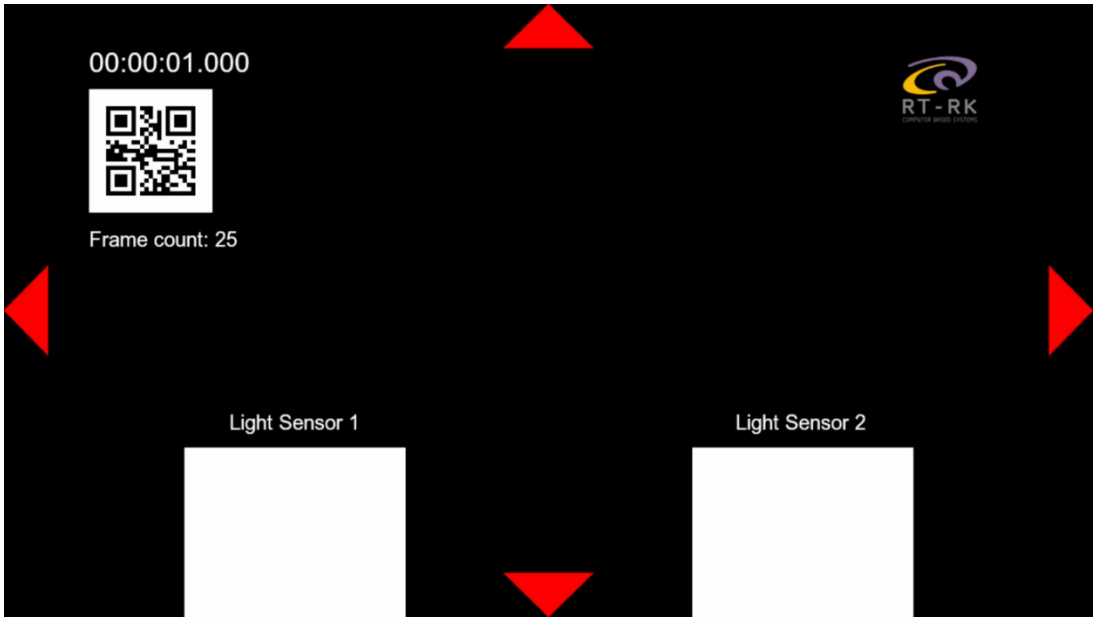
Kvadrati moraju biti ispunjeni crnom bojom, u trenutku bljeska se moraju ispuniti bijelom bojom, a po završetku ponovno zacrniti. Trajanje svakog bljeska mora biti točno 120 milisekundi.

Zvučni signali moraju biti na frekvenciji između 1 i 5 kHz, a traju 120 milisekundi. Pojavljuju se periodički, a prvi zvučni signal označuje početak perioda. Potom slijedi određena pauza, nakon čega slijedi novi zvučni signal. Raspored pojavljivanja se može vidjeti na slici 2.3.2. Nakon završetka posljednjeg zvučnog signala i isteka pauze nakon njega, započinje novi period koji slijedi isti uzorak.

Početak signala	Vrijeme između signala
T + 0 sec	
	1 sec
T + 1 sec	
	1 sec
T + 2 sec	
	2 sec
T + 4 sec	
	2 sec
T + 6 sec	
	3 sec
T + 9 sec	
	2 sec

Slika 2.3.2. Vrijeme pojave zvučnih signala

Testiranje sinkronizacije se vrši na video zapisu snimljenom kamerom visoke razlučivosti. Svaki okvir u videu mora imati sredstvo kojim se jednoznačno određuje, a u ovom slučaju se koristi kod brzog odgovora (engl. *Quick Response code* – QR kod) u kojem je sadržano točno vrijeme u kojem se okvir pojavljuje. Format zapisa je vidljiv na slici 2.3.3., a sadrži vrijeme zapisano u obliku sati:minute:sekunde.milisekunde.



Slika 2.3.3. Izgled jednog okvira u videu

Za potrebe sinkronizacije radi se usporedba srednjeg vremena zvučnih signala i bljeskova. Formula za izračun glasi:

$$T_s = \frac{T_P + T_K}{2}$$

gdje je:

- T_s – srednje vrijeme detekcije
- T_P – početak zvučnog signala ili bljeska
- T_K – kraj zvučnog signala ili bljeska

Potom se radi usporedba srednjeg vremena lijevog bljeska i desnog bljeska, lijevog bljeska i zvučnog signala te desnog bljeska i zvučnog signala. Uvjeti koji određuju uspješnost prolaska testa su sljedeći:

- ako srednja vremena nisu unutar određene tolerancije, test nije zadovoljen
- ako bilo koji bljesak ili zvučni signal nije popraćen odgovarajućim bljeskom ili zvučnim signalom, test nije zadovoljen
- ako se ne detektira niti bljesak niti zvučni signal duže od 3.5 sekundi, test nije zadovoljen
- ako se unutar vremenskog raspona od 400 ms pojavi više od jednog bljeska ili zvučnog signala, test nije zadovoljen

Ako niti jedno od navedenih pravila nije prekršeno, test je zadovoljen. Kao rezultat se vraća uspjeh ili neuspjeh prolaska testa, a u posebnoj datoteci se sprema izvještaj koji sadrži vrijeme svakog bljeska i zvučnog signala.

2.4. QR kod

QR kod je vrsta matričnog barkoda. Razvila ga je tvrtka Denso Wave 1994. godine za potrebe označavanja i praćenja vozila u proizvodnim pogonima za Toyotu. Ima raširenu primjenu zato što omogućuje veliku brzinu čitanja i spremanje veće količine podataka od klasičnih barkodova.

Sastoji se od 3 veća kvadrata koji se nalaze u donjem lijevom, gornjem lijevom i gornjem desnom uglu, a optički čitači ih koriste za potvrdu da se radi o QR kodu i za ispravno pozicioniranje. Ostatak koda čine manji crni kvadrati na bijeloj pozadini i u sebi sadrže kodiranu informaciju. Crni kvadrati se pretvaraju u binarne brojeve i procesuiraju korištenjem Reed-Solomonovog algoritma za ispravak grešaka dok se ne dobije ispravna informacija.



Slika 2.4.1. Izgled QR koda

Postoje 4 standardizirana načina kodiranja – numerički, alfanumerički, binarni i kanji (japanski tradicionalni simboli). Numerički način omogućuje korištenje samo arapskih brojeva, a njime je moguće spremiti 7089 znakova. Alfanumerički zapis omogućuje korištenje arapskih brojeva, velikih slova engleske abecede i određenih znakova. Binarni način može spremiti 2953 znaka u 8-bitnom zapisu, a kanji/kana 1817 japanskih znakova.

U sklopu ovog rada, na svakom okviru u videu se nalazi QR kod u kojem je zapisano vrijeme trenutnog okvira. Aplikacija mora biti u stanju pročitati vrijeme zapisano u QR kodu i iz pročitane informacije izvući koristan podatak. Format u kojem se zapisuje vrijeme je vidljiv na slici 2.3.3. Obzirom da osim znamenki sadržava i znakove, čitač QR koda će ga pročitati kao niz znakova (engl. *string*). Dobiveni niz znakova je potrebno pretvoriti u cjelobrojnu vrijednost koja će se moći kasnije koristiti i uspoređivati.

Za čitanje QR koda se koristi C++ biblioteka otvorenog koda *zxing* (skraćeno od *Zebra Crossing*), koja je izabrana zbog permisivne licence Apache 2.0, robusnosti i brzine izvođenja. Testirano je više sličnih biblioteka na nekoliko stotina slika na kojima se nalazi QR kod, slikanih pod različitim kutovima, razinama osvjetljenja i udaljenostima od ekrana, a *zxing* nudi najbolji omjer brzine i točnosti u radu. Biblioteka funkcionira na način da se pozove funkcija za čitanje QR koda koja kao parametar prima sliku na kojoj se kod nalazi, a vraća znakovni niz koji sadrži podatak zapisan u kodu.

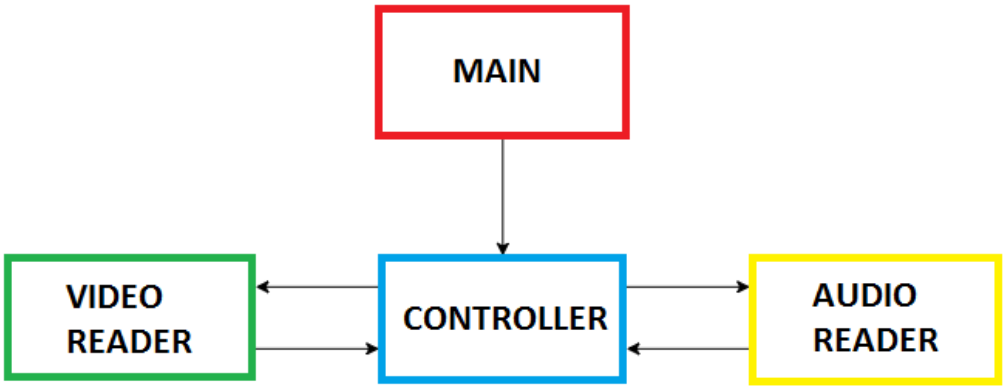
Za konverziju znakovnih nizova u cjelobrojne vrijednosti je osmišljeno vlastito rješenje koje parsira znakovni niz znak po znak. Znamenke koje predstavljaju sate su dvotočkama odvojene od znamenki koje predstavljaju minute, minute su dvotočkama odvojene od sekundi, a sekunde su točkom odvojene od milisekundi. Aplikacija prolazi kroz znakovni niz i koristi dvotočke i točke kao separatore, a vrijednosti između njih sprema u odgovarajuće varijable. Potom se sve pretvara u cjelobrojnu vrijednost koja predstavlja broj milisekundi i koristi se u daljnjem radu.

3. Koncept rješenja

Aplikacija se pokreće iz konzole i prima naziv video datoteke kao parametar. Podijeljena je na 4 modula od kojih je svaki zadužen za obavljanje neke funkcionalnosti, a svi su detaljno opisani u idućem potpoglavlju. Na konzoli se prilikom ulaska u svaki modul ispisuje trenutno stanje u kojem se aplikacija nalazi i što se trenutno obrađuje. Po završetku obrade se kao rezultat vraća poruka o sinkroniziranosti sadržaja te se kreiraju 2 izvještaja, jedan u obliku tekstualne datoteke a drugi u CSV (engl. *Comma Separated Values*) formatu.

3.1. Opis modula

Aplikacija se sastoji od 4 modula – *main*, *controller*, *videoReader* i *audioReader*. Prilikom pokretanja aplikacije se pokreće *main* modul, u kojem se provjerava koliko je parametara korisnik unio, a potom se poziva *controller* modul koji poziva sve ostale. Međusobna komunikacija je vidljiva na slici 3.1.1.



Slika 3.1.1. Prikaz komunikacije između modula

U *controller* modulu se deklariraju vektori u koje će se spremati vremena detekcije i pozivaju se moduli za analizu videa i zvuka. Prvo se radi demultipleksiranje video zapisa da bi se izvukao zvuk koji se potom sprema u posebnu datoteku. Zvuk se uvijek sprema u istom formatu (*wave* format), uz istu frekvenciju uzorkovanja koja iznosi 44.1 kHz. Zatim se poziva modul za analizu videa, koji detektira bljeskove i bilježi vremena njihovih početaka i završetaka. Potom se poziva modul za analizu zvuka, koji detektira zvučne signale i bilježi

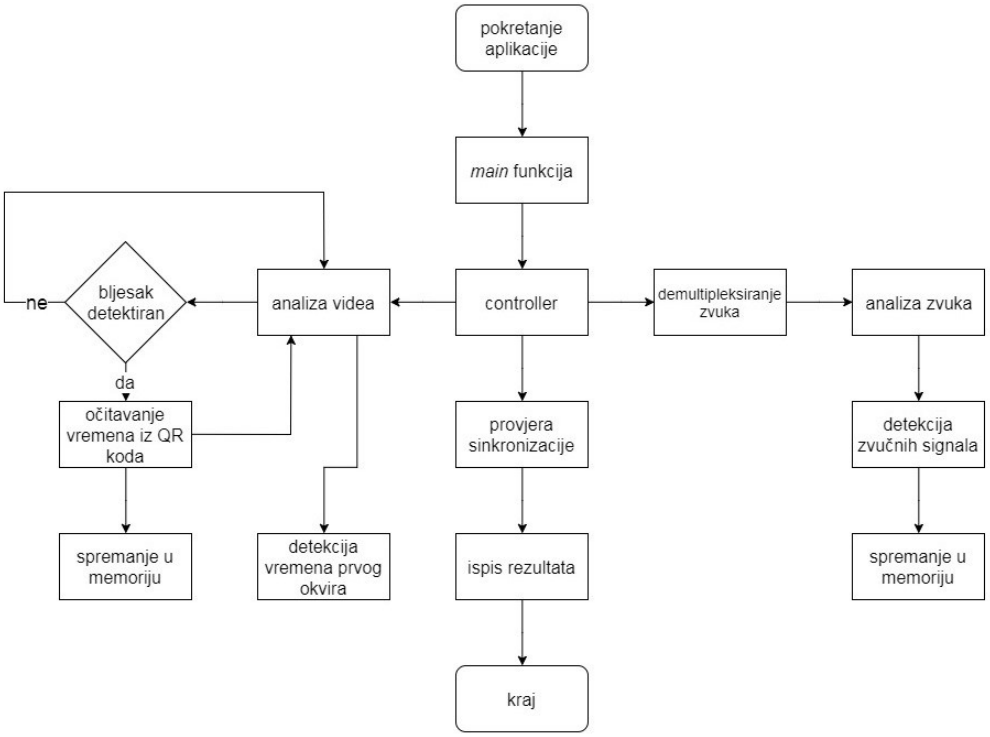
vremena njihovih početaka i završetaka. Vremena detekcije bljeskova i zvučnih signala se spremaju u vektore koji se u module prosljeđuju po referenci, zbog čega su vidljivi *controller* modulu. Na kraju se poziva funkcija koja uspoređuje srednja vremena pojave bljeskova i zvučnih signala i sprema ih u izvještaj u CSV formatu, a korisniku kao rezultat vraća odgovor o sinkroniziranosti videa i zvuka.

Modul *videoReader* se sastoji od funkcija koje vrše obradu cijelog videa okvir po okvir. Koristi se C++ biblioteka otvorenog koda *OpenCV* koja je razvijena za brz, jednostavan i optimiziran rad na slikama i videu. Pomoću nje se video rastavlja na okvire i na svakom se okviru radi analiza. Ako se detektira početak ili kraj bljeska, iz QR koda se očitava vrijeme i sprema se u memoriju. Vremena početka i završetka se spremaju na odvojena mjesta u memoriji, a zatim se iz njih dobije srednje vrijeme pojave koje specifikacija zahtijeva.

Modul *audioReader* se sastoji od funkcija koje vrše obradu cijelog zvučnog zapisa uzorak po uzorak. Prilikom izdvajanja zvučnog zapisa iz videa, pretvoren je u jednokanalni 16-bitni PCM zapis u *wave* formatu sa 44100 uzoraka po sekundi. Svaki uzorak predstavlja jednu točku u vremenu i nad svakim uzorkom se provodi analiza. Detektiraju se vremena pojave i završetka svakog zvučnog signala te se spremaju u memoriju.

3.2. Princip rada

Aplikacija za testiranje sinkroniziranosti se pokreće putem konzole, a prilikom pokretanja se od korisnika očekuje da kao ulazne parametre preda putanju do video zapisa na kojem se vrši analiza i frekvenciju na kojoj su zvučni signali. Iz *main* funkcije se poziva *controller* modul, koji poziva ostale module, upravlja njihovim radom i obavlja testiranje sinkroniziranosti.



Slika 3.2.1. Dijagram toka aplikacije

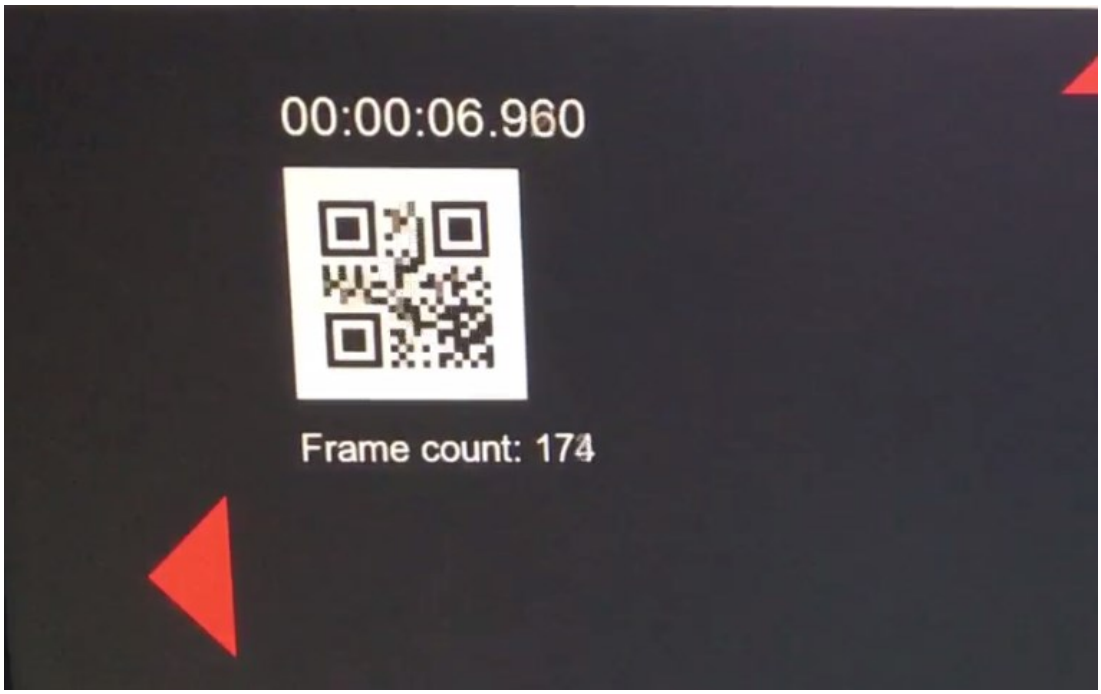
U *controller* modulu se deklariraju vektori u koje se sprema vrijeme početka, vrijeme završetka i srednje vrijeme pojave bljeskova i isti vektori za vremena zvučnih signala, odnosno ukupno 6 vektora. Vektori se funkcijama prosljeđuju po referenci, tako da su sve izmjene vidljive i u funkciji gdje su vektori deklarirani. Deklarira se i varijabla u koju se sprema vrijeme početka video zapisa, a potrebna je zato što se video ne mora krenuti snimati od nule. Video se može krenuti snimati u bilo kojem trenutku, a vrijeme bljeska će se pročitati iz QR koda. Vrijeme pojave zvučnog signala se ne može detektirati iz QR koda nego se uzima vrijeme od početka zvučnog zapisa, pa će u tom slučaju u savršeno sinkroniziranom videu biti razlika između bljeskova i zvučnih signala. Iz tog razloga se iz prvog okvira pročita QR kod da bi se dobilo početno vrijeme, a zatim se to vrijeme dodaje na zvuk kao vremenski odmak.

Da bi se provela analiza zvuka, potrebno je napraviti demultipleksiranje video zapisa i izlučiti zvučni zapis, a za tu svrhu se koristi besplatni skup programa *FFmpeg*. Njegovim korištenjem se može zadati željeni format audio datoteke. Format je *wave* zato što je u njemu sačuvan originalni zapis bez kompresije i jednostavan je za parsiranje. Većina zvučnih zapisa ima 2 kanala, ali za potrebe analize oni nisu potrebni. Stoga se zvučni zapis pretvara u

jednokanalni, čime se dvostruko smanjuje veličina zvučne datoteke i dvostruko se ubrzava vrijeme analize zvuka. Frekvencija uzorkovanja je 44.1 kHz.

Nakon demultipleksiranja i izlučivanja zvučnog zapisa, počinje analiza videa. Korištenjem *OpenCV*-a se dohvaćaju osnovni podaci o videu, kao što su trajanje videa, ukupan broj okvira, broj okvira po sekundi te širina i visina okvira u videu. Svi navedeni podaci se javljaju korisniku, a zatim se spremaju za potrebe daljnje obrade. Potom se pokreće petlja u kojoj se pri svakoj iteraciji obradi po jedan okvir. Obzirom da je u specifikaciji unaprijed određeno u kojem dijelu ekrana se pojavljuju bljeskovi, a poznate su širina i visina slike, aplikacija gleda točno određeno područje slike. Slika se prvo pretvara u sivu sliku (engl. *grayscale*), a zatim se na predefiniranom mjestu uzima 900 piksela (30x30), zbraja se njihov intenzitet svjetline i dobije se prosječna vrijednost svjetline tog područja. Ako se utvrdi da je svjetlina dovoljno velika (vrijednost intenziteta svjetline veća od 230), zaključuje se da je bljesak prisutan i pali se zastavica koja označava da je bljesak aktivan. Potom se poziva funkcija koja traži QR kod i čita vrijeme iz njega, a zatim se u novoj funkciji vrijeme pretvara u milisekunde. U tom obliku se sprema u vektor koji označava početak bljeska. Kad se utvrdi da bljesak nije prisutan, a zastavica je aktivna, ponovno se očitava QR kod i vrijeme se sprema u vektor koji označava kraj bljeska, a zastavica ponovno postaje neaktivna.

Čitanje QR koda značajno usporava rad aplikacije i povećava vrijeme izvođenja. Da bi se izbjeglo nepotrebno dugačko vrijeme obrade, QR kod se očitava samo iz okvira u kojima postoji prijelaz iz aktivnog u neaktivno stanje zastavice, odnosno samo kada se bljesak pojavi ili kada se ugasi. Referentni video ima 25 okvira po sekundi pa je potrebno snimati ga kamerom koja ima minimalno dvostruko više okvira po sekundi da bi se moglo garantirati uspješno čitanje QR koda. Uz manji broj okvira po sekundi postoji mogućnost da se neki okvir pojavi samo jednom i to upravo u trenutku prijelaza na drugi okvir, a u tom slučaju se ne može garantirati da će se QR kod moći uspješno pročitati. Ako je broj okvira u sekundi 50 ili više, svaki se okvir mora pojaviti barem 2 puta pa je nemoguće da bude uhvaćen u trenutku prijelaza, odnosno kada je nečitljiv. Na slici 3.2.2. se vidi okvir koji je uhvatio prijelaz pa je QR kod nečitljiv.



Slika 3.2.2. Primjer prijelaza iz jednog u drugi okvir gdje je QR kod nečitljiv

Nakon što se pročitaju svi okviri, izlazi se iz petlje. Funkcija zatim vraća cjelobrojnu vrijednost koja predstavlja vrijeme prvog okvira izraženo u milisekundama, a ta se vrijednost zatim sprema u *controller* modulu.

Nakon obavljene analize videa, započinje analiza zvuka. Funkciji koja radi analizu zvuka se prosljeđuje vrijeme početka videa i to se vrijeme koristi kao vrijeme odmaka. Prvotno rješenje je podrazumijevalo korištenje brze Fourierove transformacije (engl. *Fast Fourier transform* – FFT) za detekciju zvučnih signala. FFT analizira određeni skup uzoraka i pretvara ga u frekvencijsku domenu, iz čega se može vidjeti koje su frekvencije prisutne u skupu. Najveći problem kod ovakvog pristupa proizlazi iz vremenske složenosti ovakvog pristupa. FFT se provodi nad svakim uzorkom u cijelom zvučnom zapisu, a za analizu svakog uzorka je potrebno kreirati niz u koji se sprema određen broj uzoraka koji slijede iza njega i zajedno tvore jedan skup uzoraka. Na tom skupu se odrađuju proračuni koji vraćaju dominantnu frekvenciju samo za prvi uzorak iz skupa. Isto se ponavlja za svaki uzorak u cijelom zapisu, što znači da se nad svakim uzorkom analiza ponavlja velik broj puta, uz dodatan zahtjev novog grupiranja i ponovnog provođenja računskih operacija. Sve navedeno dovodi do dugačkog vremena obrade, čak i za male zvučne zapise.

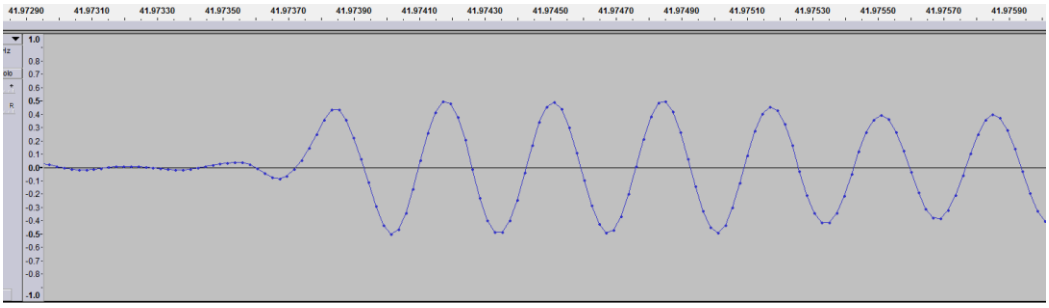
Obzirom na navedena ograničenja, odlučeno je koristiti jednostavnije rješenje. Metoda koja je odabrana podrazumijeva direktnu analizu svakog uzorka. Da bi se zvuk mogao analizirati, potrebno ga je pretvoriti u oblik pogodan za obradu. *Wave* format definira zapis na način da prvih 40 bajta čine zaglavlje koje sadrži informacije o zvučnom zapisu, iduća 4 bajta predstavljaju poseban podatkovni komad, a od 45. bajta nadalje su sadržani podaci o zvuku. Obzirom da je zvučni zapis pretvoren u 16-bitni PCM, svaka 2 susjedna bajta predstavljaju jedan uzorak u vremenskoj domeni. Korištenjem C++ programskog jezika je moguće učitati cijelu datoteku u memoriju, a zatim ju čitati bajt po bajt. Format zapisa je uvijek isti pa se čitanjem prvih 40 bajta dobiju svi podaci o zvučnom zapisu iz zaglavlja. Na taj način se može provjeriti da je zvučni zapis ispravno demultipleksiran iz video zapisa. Iduća 4 bajta koja čine podatkovni komad se pročitaju, ali nemaju konkretnu primjenu u ovom slučaju. Od 45. bajta se čitaju podaci koji predstavljaju uzorke i spremaju se u odvojenu tekstualnu datoteku u obliku cjelobrojnih vrijednosti. Izgled zvučnog zapisa u *wave* formatu je vidljiv na slici 3.2.3.

Struktura *wave* audio formata

Početni bajt	Ime polja	Veličina polja	Opis
0	ChunkID	4	Opis "RIFF" odlomka
4	ChunkSize	4	
8	Format	4	
12	Subchunk1ID	4	Sadrži podatke o formatu zvučnog zapisa
16	Subchunk1Size	4	
20	AudioFormat	2	
22	NumChannels	2	
24	SampleRate	4	
28	ByteRate	4	
32	BlockAlign	2	
34	BitsPerSample	2	
36	Subchunk2ID	4	Označava veličinu podatkovnog odlomka i sadrži neobrađene podatke
40	Subchunk2Size	4	
44	Data	Subchunk2Size	

Slika 3.2.3. Struktura *wave* audio formata

Zadana je frekvencija uzorkovanja od 44.1 kHz što znači da postoji 44.100 vremenskih uzoraka po sekundi. Analiza se provodi na svakom uzorku, a zbog frekvencije uzorkovanja svaki uzorak predstavlja 44.100/1. sekundi, odnosno 22.68 mikro sekundi. Time se postiže vrlo visoka preciznost mjerenja i osigurava se točnost konačnih rezultata. Osim toga, ovaj princip postiže nekoliko puta veću brzinu obrade. Iz tekstualne datoteke se učitavaju cjelobrojne vrijednosti koje predstavljaju amplitude uzoraka, spremaju se u niz u memoriji i potom se pokreće analiza.



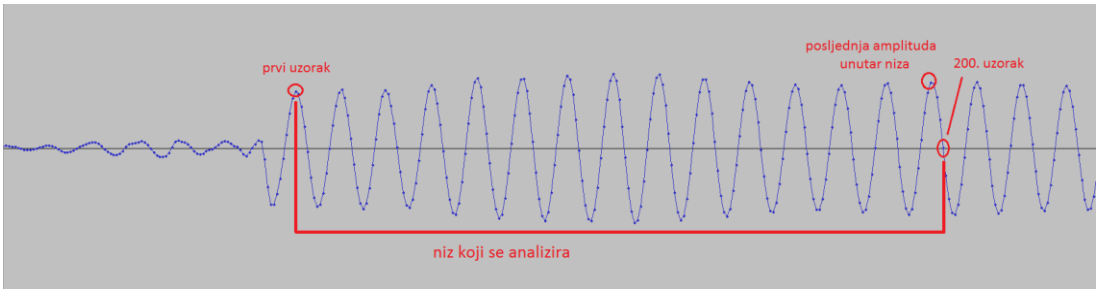
Slika 3.2.4. Izgled uzoraka u vremenskoj domeni gdje svaka točka predstavlja jedan uzorak

Video na kojem se vrši testiranje se snima u kontroliranim uvjetima uz najveću moguću tišinu, što osigurava da će zvučni signal iz videa biti najglasniji zvuk i imati najveću amplitudu. Stoga se prvo prolazi kroz cijeli niz i traži se najveća vrijednost (koja predstavlja najveću amplitudu) koja se zatim sprema u zasebnu varijablu. Nakon što se pronađe najveća amplituda u cijelom zapisu, započinje obrada zvučnog signala uzorak po uzorak. Za svaki uzorak se provjerava vrijednost amplitude, a ako se utvrdi da je njezina vrijednost veća od 50% vrijednosti najveće amplitude u cijelom zapisu, smatra se da je to potencijalno zvučni signal te se pristupa računanju frekvencije. Frekvencija se može dobiti iz perioda sinusnog signala po formuli:

$$f = \frac{1}{T}$$

gdje je *f* frekvencija, a *T* period sinusa. Period sinusa se računa tako da se uzme skup od 200 uzoraka od prve amplitude i provjerava se koliko amplituda postoji unutar tog skupa. Zatim se broj na kojem se pojavljuje zadnja amplituda (u većini slučajeva 200. uzorak nije amplituda pa se uzima redni broj uzorka na kojem je bila posljednja amplituda) dijeli s brojem detektiranih amplituda. Time se dobije prosječno trajanje jednog perioda, a iz njega se dobije frekvencija na

traženom skupu uzoraka. Ako se utvrdi da frekvencija odgovara onoj koju je korisnik zadao (toleriraju se odstupanja do 10%), sa sigurnošću se može tvrditi da je riječ o početku zvučnog signala. Zatim se pali zastavica koja označava da je zvučni signal prisutan.



Slika 3.2.5. Način detekcije zvučnog signala

Primjer je vidljiv na slici 3.2.5. Korišten je zvučni signal koji ima frekvenciju 3 kHz. Pronalaskom prvog uzorka čija amplituda ima dovoljno visoku vrijednost, počinje se analizirati sljedećih 200 uzoraka. Ukupno je detektirano 14 amplituda, a posljednja se nalazi na 197. uzorku. 197 se dijeli sa 14, što daje vrijednost 14.07, odnosno prosječno trajanje jednog perioda unutar niza. Frekvencija uzorkovanja je 44.1 kHz tako da svaki uzorak predstavlja 0.00002268 sekundi, što se množi s trajanjem jednog perioda (14.07) i dobije se vrijednost 0.00031905. Taj broj predstavlja trajanje perioda sinusa izraženo u sekundama i uvrštava se u formulu za računanje frekvencije. Rezultat je 3.13 kHz, što je unutar tolerancije od 10% i zaključuje se da je riječ o početku zvučnog signala.

Daljnjim prolaskom kroz niz se radi obrnuti proces te se traže uzorci čija frekvencija neće biti odgovarajuća ili će vrijednost njihove amplitude biti preniska. Kada se pronađe takav uzorak, gasi se zastavica koja označava prisutnost zvučnog signala i ponovno se počinje tražiti početak novog signala koji zadovoljava uvjete.

Nakon prolaska kroz cijeli zvučni zapis, aplikacija računa srednja vremena pojave bljeskova i zvučnih signala te ih sprema u odvojene vektore. Ti se vektori zatim šalju u novu funkciju koja testira sinkronizaciju. Prvo se provjerava sadrže li oba vektora jednak broj elemenata. Ako se utvrdi da njihov broj nije jednak, zaključuje se da broj bljeskova i zvučnih signala nije jednak što znači da je test sinkronizacije pao. Sve se vrijednosti bilježe u izvještaj u CSV formatu, kreira se poseban tekstualni izvještaj i izlazi se iz aplikacije.

CSV izvještaj u prvi redak upisuje vrijeme detekcije prvog zvučnog signala, vrijeme završetka prvog zvučnog signala i srednje vrijeme njegove pojave. U drugi redak se upisuju iste vrijednosti za prvi lijevi bljesak, a u treći redak iste vrijednosti za prvi desni bljesak. Potom se isti zapis ponavlja za svaki idući zvučni signal i bljesak. Izgled izvještaja je vidljiv na slici 3.2.4.

	A	B	C	D
1	A	1020	1140	1080
2	L1	1000	1120	1060
3	L2	1000	1120	1060
4	A	2018	2138	2078
5	L1	2000	2120	2060
6	L2	2000	2120	2060
7	A	4017	4137	4077
8	L1	4000	4120	4060
9	L2	4000	4120	4060
10	A	6018	6138	6078
11	L1	6000	6120	6060
12	L2	6000	6120	6060
13	A	9019	9139	9079
14	L1	9000	9120	9060
15	L2	9000	9120	9060

Slika 3.2.6. Izgled CSV izvještaja nakon završene analize

Drugi izvještaj se zapisuje u tekstualnu datoteku i sadrži objašnjenje zašto je video zapis sinkroniziran ili nesinkroniziran. Ako je broj elemenata u vektorima koji sadrže srednja vremena pojave zvučnih signala i bljeskova različit, u izvještaj se zapisuje da je video zapis nesinkroniziran zato što je detektiran različit broj bljeskova i zvučnih signala. Ako vektori sadrže jednak broj elemenata, provjeravaju se elementi s istim indeksima i provjerava se koliko iznose odstupanja. U slučaju da su sva odstupanja unutar tolerancije, u izvještaj se zapisuje da je video zapis sinkroniziran i javlja se koliko iznosi najveće odstupanje. U suprotnom se u izvještaj zapisuje da je video zapis nesinkroniziran zbog prevelikog odstupanja i zapisuje se u kojem je trenutku ono detektirano.

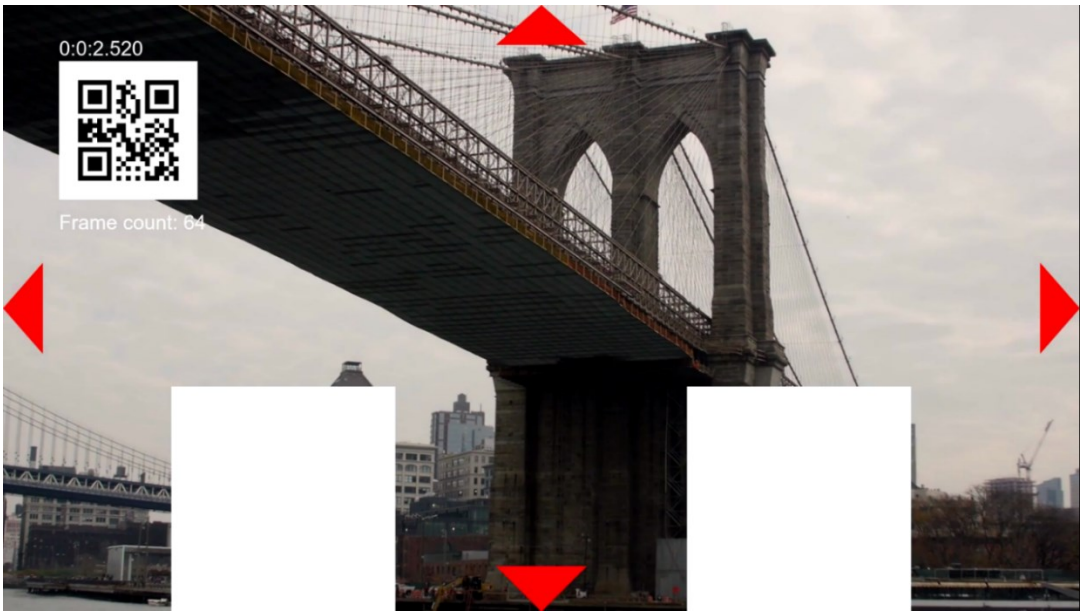
4. Testiranje i rezultati

Testiranje sinkroniziranosti se provodi u strogo kontroliranim uvjetima koji su unaprijed poznati i strogo definirani. Kamera kojom se snima video mora biti bar HD kvalitete (rezolucija 1280x720 piksela) i postavljena na toliku udaljenost od ekrana na kojem se reproducira video da obuhvati crvene strelice vidljive na slici 2.3.3. Testiranjem je utvrđeno da šum i pozadinska buka mogu biti prisutni za vrijeme snimanja, ali zvučni signal kojim se testira sinkronizacija mora biti najglasniji zvuk koji se čuje.

4.1. Kreiranje testnih sekvenci

Izgled video zapisa koji se snimaju kamerom i zatim analiziraju je propisan testnom specifikacijom i objašnjen u poglavlju 2.3. U sklopu ovog rada je izrađeno vlastito rješenje koje kreira tražene video zapise – generator sekvenci (engl. *Sequence generator* – SG).

Za izradu generatora sekvenci je korišten programski jezik JavaScript, okruženje izvršavanja *Node.js* i skup programa *FFmpeg*. Generator sekvenci kao ulaz prima lokaciju video zapisa, a kao rezultat vraća novi video zapis. Korištene su ugrađene funkcije koje *Node.js* nudi za razdvajanje video zapisa u slike, gdje svaka slika predstavlja jedan okvir u videu. U svaki sliku se zatim dodaje vrijeme okvira i QR kod u kojem je ono zapisano. Generator sekvenci koristi brojač okvira kao dodatni mehanizam provjere pa se na svaku sliku zapisuje i broj sadržan u brojaču. Na područje rezervirano za bljeskove se dodaju crni ili bijeli kvadrati, ovisno o tome je li bljesak aktivan ili ne. Korištena je petlja koja prati sliku 2.3.2. za odlučivanje hoće li bljesak biti prisutan ili ne (koriste se zadani periodi pojavljivanja). Kvadrati se uvijek nalaze na mjestu koje je definirano specifikacijom i uvijek su istih dimenzija. Konačno, na rubove ekrana se dodaju 4 crvene strelice koje označavaju područje snimanja, odnosno granicu preko koje ne treba snimati. Ovo je vrlo bitno za ispravno pozicioniranje kamere. Nakon dodavanja svih navedenih značajki na svaku sliku, koriste se *Nodes.js* funkcije iz biblioteke *ImageMagick*. Broj okvira u sekundi ostaje isti.



Slika 4.1.1. Izgled jednog okvira kreiranog korištenjem generatora sekvenci

Tako dobiven video zapis ne sadrži nikakav zvuk pa je potrebno naknadno dodati i zvučne signale u njega. Kreiran je jedan zvučni signal, a potom se koristi *FFmpeg* da bi se isti signal ubacio na više mjesta u video zapisu. Korištena je petlja koja prati isti princip kao i petlja za dodavanje bljeskova, što znači da bi zvučni signali i bljeskovi trebali biti savršeno sinkronizirani.

4.2. Testiranje aplikacije

Aplikacija bi trebala uspješno obavljati provjeru sinkronizacije za svaki video zapis ako su ispunjeni uvjeti koji su prethodno navedeni – kamera visoke razlučivosti s bar dvostruko više okvira u sekundi od videa koji se snima, ispravan položaj kamere i tišina u prostoriji u kojoj se snima.

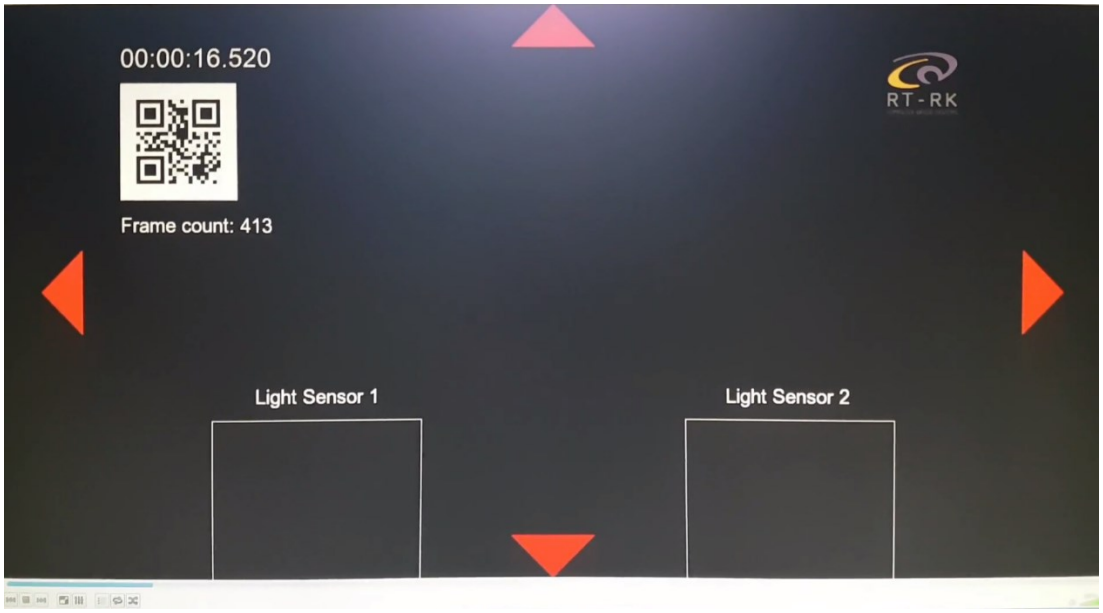
Kreirane su dvije vrste video zapisa – sinkronizirani i nesinkronizirani. Osim razlike u sinkroniziranosti, testni video zapisi imaju i različite frekvencije zvučnih signala. Napravljena su 2 testna slučaja. Jedan testni slučaj čini 5 različitih sinkroniziranih video zapisa, a na svakom je prisutna različita frekvencija zvučnog signala (1, 2, 3, 4 i 5 kHz). Drugi testni slučaj čine nesinkronizirani video zapisi uz iste frekvencije kao i sinkronizirani zapisi.

Testiranje se provodi na oba skupa video zapisa, a smatrat će se uspješnim ako sinkronizirani zapisi zadovolje, a nesinkronizirani ne zadovolje testiranje. Nakon analize svih testnih slučajeva, testovi su ponovljeni uz promijenjene uvjete snimanja – slabije osvjetljenje u prostoriji, veći kut kamere u odnosu na ekran i prisutnost šuma ili buke u prostoriji kako bi se provjerila robusnost aplikacije i njezina tolerancija na nesavršene uvjete snimanja.

Kamera kojom su se snimali video zapisi je kamera na mobilnom telefonu iPhone SE, uz rezoluciju 1080p (full HD) i 60 okvira u sekundi. Frekvencija uzorkovanja zvuka je 48 kHz.

4.3. Rezultati testiranja

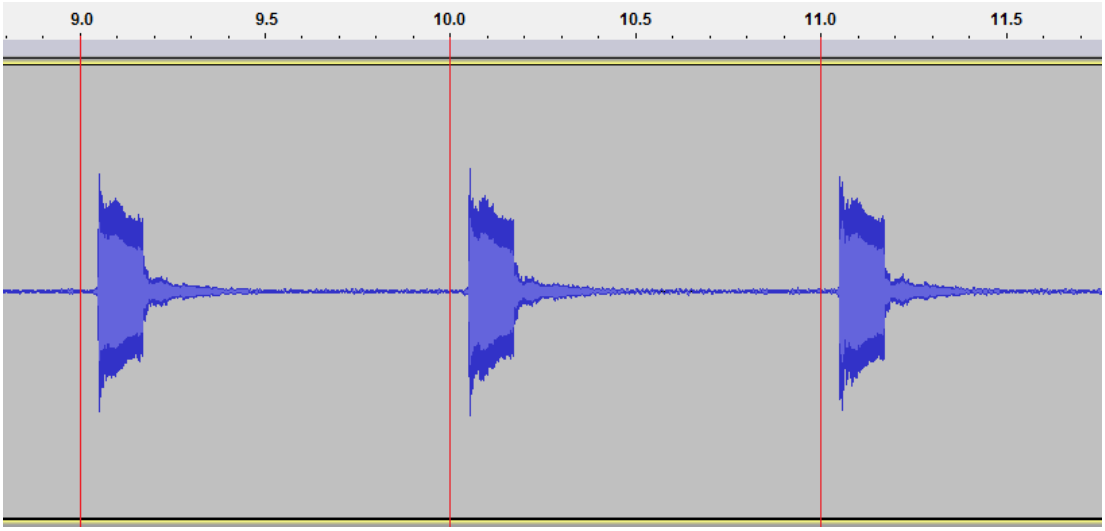
Prvo testiranje je provedeno poštujući sve zadane uvjete. Snimljeni zapis traje samo 14 sekundi i služi za provjeru radi li aplikacija kako je predviđeno. Snimanje je započelo 12 sekundi i 600 milisekundi od početka reprodukcije video zapisa na računalu.



Slika 4.3.1. Izgled jednog okvira video zapisa koji je korišten za prvo testiranje

Tolerancija na odstupanje između zvučnog signala i bljeskova je postavljena na 50 milisekundi. Testiranjem je utvrđeno da je video zapis sinkroniziran, a pregledom izvještaja dolazi se do najvećeg odstupanja koje iznosi 45 milisekundi. Odstupanje je prisutno zbog razlike u brzini kojom zvuk i slika dolaze do kamere, odnosno zbog nesavršenosti mjernih

uređaja. Na slici 4.3.2. se može vidjeti da postoji kašnjenje u samom zvučnom zapisu koji je snimljen kamerom, odnosno zvučni signali ne počinju točno u punu sekundu nego kasnije.



Slika 4.3.2. Kašnjenje zvučnog signala u snimljenom video zapisu

4.3.1. Sinkronizirani video zapisi

Izvršeno je 10 testiranja kojima su obuhvaćena oba testna slučaja. Prvi set testova su sinkronizirani video zapisi i očekuje se da će svi uspješno zadovoljiti testiranje, neovisno o frekvenciji zvučnih signala. Prvo je pokrenut test na sinkroniziranom video zapisu na kojem je prisutan zvučni signal s frekvencijom 1 kHz. Aplikacija je uspješno detektirala sve bljeskove i zvučne signale te zaključila da je njihov broj jednak. Potom su provjerena sva odstupanja između srednjih vremena njihove detekcije i utvrđeno je da najveće odstupanje iznosi 47 milisekundi, što je unutar zadane tolerancije. Rezultat je odgovor da su audiosignal i videosignal sinkronizirani.

Isti postupak je ponovljen za sve preostale video zapise iz prvog seta testova i svi su zadovoljili uvjete sinkroniziranosti. Na slici 4.3.3. se vide rezultati tih testova. Kao što se iz slike može vidjeti, odstupanja se razlikuju za najviše 4 milisekunde i ne ovise o frekvenciji zvučnog signala. Aplikacija bi trebala detektirati zvučne signale na bilo kojoj frekvenciji i jednako uspješno raditi neovisno o njoj, a korisniku se prepušta mogućnost odabira te frekvencije. Ograničenje koje pritom postoji je da svi zvučni signali koji se pojavljuju u jednom video zapisu moraju biti na isto frekvenciji (uz dozvoljenu toleranciju od 10%), odnosno nije moguće imati zvučne signale od npr. 1 i 5 kHz u istom video zapisu.

Frekvencija zvučnog signala (kHz)	Video zapis sinkroniziran	Najveće odstupanje (milisekunde)
1	da	47
2	da	46
3	da	47
4	da	48
5	da	44

Slika 4.3.3. Rezultati obrade prvog seta testova

4.3.2. Nesinkronizirani video zapisi

Drugi set testova je proveden na video zapisima koji nisu sinkronizirani, a razlozi nesinkroniziranosti su sljedeći:

- 1. video zapis – broj bljeskova i zvučnih signala nije jednak
- 2. video zapis – bljeskovi kasne 520 milisekundi
- 3. video zapis – zvučni signali kasne 500 milisekundi
- 4. video zapis – lijevi i desni bljeskovi ne počinju u isto vrijeme
- 5. video zapis – kombinacija prethodnih problema

Da bi se aplikacija smatrala uspješnom, svih 5 testova moraju vratiti rezultat da video zapisi nisu sinkronizirani, ali mora i utvrditi koji je razlog nesinkroniziranosti da bi korisnik mogao ispraviti nedostatke.

U prvom video zapisu broj bljeskova i zvučnih signala nije jednak. Ukupno se pojavljuje 28 zvučnih signala i 24 bljeska, pri čemu je frekvencija zvučnih signala 1 kHz. Aplikacija ih je sve uspješno detektirala i spremila vremena njihove pojave u vektore, ali ti vektori ne sadrže jednak broj elemenata. Prvi uvjet koji se provjerava je upravo broj elemenata u vektorima, stoga aplikacija niti ne provjerava kolika su odstupanja nego odmah korisniku javlja da video zapis nije sinkroniziran zbog nejednakog broja bljeskova i zvučnih signala.

U drugom video zapisu lijevi i desni bljesak su međusobno sinkronizirani, ali kasne u odnosu na zvučni signal. Oba bljeska počinju i završavaju 520 milisekundi kasnije tako da je i srednje vrijeme pojave 520 milisekundi veće od očekivanog. Frekvencija zvučnog signala iznosi 2 kHz. Aplikacija je analizirala video zapis i utvrdila da je broj bljeskova i zvučnih signala jednak, a potom je analizirala odstupanja u vremenima njihove detekcije. Utvrđeno je da najveće odstupanje iznosi 476 milisekundi, a razlog zašto ono nije 520 milisekundi je

prethodno objašnjeno kašnjenje zvučnih signala koji su posljedica snimanja kamerom. Tolerancija je postavljena na 50 milisekundi, zbog čega je rezultat analize odgovor da video zapis nije sinkroniziran zbog prevelikog odstupanja u vremenu detekcije. U tekstualnom izvještaju je zapisan rezultat i komentar zašto test nije zadovoljen.

U trećem video zapisu zvučni signali kasne 500 milisekundi u odnosu na bljeskove, a njihova frekvencija iznosi 3 kHz. Svi su zvučni signali uspješno detektirani pa se pristupa analizi srednjih vremena detekcije. Najveće odstupanje iznosi 547 milisekundi kao posljedica kašnjenja od 500 milisekundi i 47 milisekundi zbog snimanja kamerom. U izvještaj se zapisuje da je video zapis nesinkroniziran zbog prevelikog odstupanja.

Četvrti video zapis sadrži jednak broj bljeskova i zvučnih signala, ali desni bljesak uvijek započinje u trenutku kada lijevi bljesak završava, odnosno ima kašnjenje od 120 milisekundi. Frekvencija zvučnih signala iznosi 4 kHz. Na slici 4.3.4. se vidi da je lijevi bljesak završio, a desni traje. Aplikacija je detektirala sve bljeskove i zvučne signale i utvrdila da je njihov broj jednak, ali je utvrdila da postoji odstupanje između lijevih i desnih bljeskova koje uvijek iznosi točno 120 milisekundi. Sve su vrijednosti zapisane u CSV izvještaj, a u tekstualni izvještaj se zapisuje da je video zapis nesinkroniziran zato što bljeskovi nisu sinkronizirani.



Slika 4.3.4. Kašnjenje desnog bljeska u odnosu na lijevi bljesak

Peti video zapis sadrži bljeskove koji kasne u odnosu na zvučne signale i desni u odnosu na lijevi te njihov broj nije jednak. Frekvencija zvučnih signala iznosi 5 kHz. Obzirom da broj

bljeskova i zvučnih signala nije jednak, aplikacija odmah zaključuje da video zapis nije sinkroniziran i završava analizu.

Frekvencija zvučnog signala (kHz)	Video zapis sinkroniziran	Razlog nesinkroniziranosti
1	ne	Nejednak broj bljeskova i zvučnih signala
2	ne	Preveliko kašnjenje bljeskova
3	ne	Preveliko kašnjenje zvučnih signala
4	ne	Nesinkroniziranost bljeskova
5	ne	Nejednak broj bljeskova i zvučnih signala

Slika 4.3.5. Rezultati obrade drugog seta testova

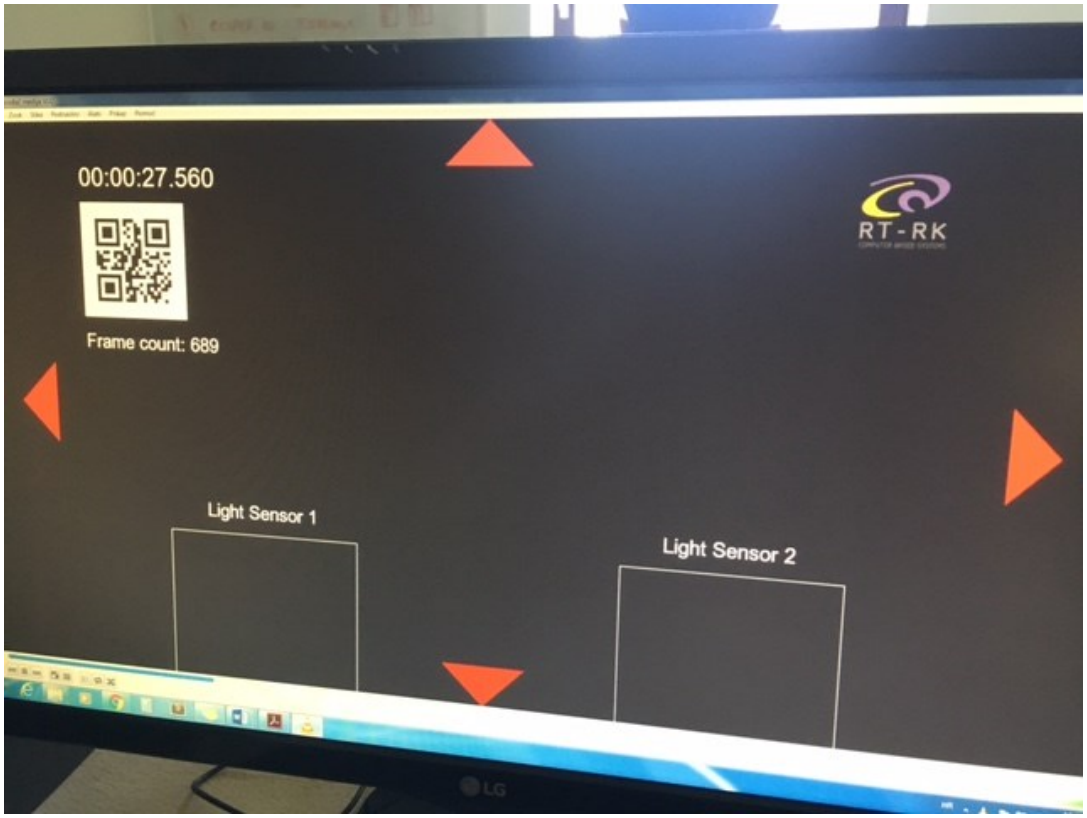
Na slici 4.3.5. se mogu vidjeti rezultati svih testova. Aplikacija je za sve video zapise iz ovog seta testova utvrdila da su nesinkronizirani i uspješno je prepoznala razloge, stoga se može smatrati da je ispunila osnovne zahtjeve. Bitno je utvrditi kada neki video zapis nije sinkroniziran, ali je jednako bitno i utvrditi razloge kako bi korisnik mogao ispraviti nedostatke.

4.3.3. Video zapisi snimani u lošijim uvjetima

Da bi se testirala robusnost aplikacije, snimljen je novi set testnih video zapisa koji su snimani u lošijim uvjetima. Snimljena su 3 video zapisa pod većim kutom kamere u odnosu na ekran, 3 video zapisa u prostoriji sa slabijim osvjetljenjem, 3 video zapisa prilikom čijeg snimanja je u prostoriji bila prisutna buka i jedan video zapis u kojem je kamera blago pomicana u obje strane tijekom snimanja. Cilj ovog seta testova je utvrditi otpornost aplikacije na različite uvjete snimanja.

Od korisnika se očekuje da kamera snima ekran pod kutom od 0 stupnjeva i cijelo vrijeme bude fiksirana. Snimljena su 3 video zapisa pod većim kutom i na svima je provedeno testiranje. Prvi video zapis je sniman pod manjim kutom i testiranje je bilo uspješno. Detektirani su svi bljeskovi i uspješno su pročitani svi QR kodovi. Drugi video zapis je sniman pod većim kutom i ovaj put aplikacija nije uspješno detektirala bljeskove. Po testnoj specifikaciji, bljeskovi se uvijek nalaze na istom mjestu, a u ovom slučaju su bljeskovi bili izvan traženog područja. Svi su QR kodovi uspješno očitani. Na slici 4.3.6. se može vidjeti kut snimanja. Treći video

zapis je sniman uz kut veći od 45 stupnjeva i u ovom slučaju bljeskovi nisu detektirani, ali ni QR kodovi više nisu uspješno očitani. Može se zaključiti da aplikacija radi uz manji nagib kamere, ali kut snimanja ne smije prelaziti 15 stupnjeva u bilo koju stranu. Snimljen je i jedan video zapis u kojem je kamera blago pomicana u sve strane, ali nikad nije ostvaren prevelik nagib. Aplikacija je uspješno detektirala sve bljeskove i očitala sve QR kodove pa se može zaključiti da kamera ne mora biti fiksirana cijelo vrijeme snimanja dok god ne prelazi prevelik kut.



Slika 4.3.6. Snimanje pod prevelikim kutom

Iduća stvar koja se provjerava je snimanje uz slabije osvjetljenje. Snimljena su 3 video zapisa u potpuno zamračenoj prostoriji i na svima je provedeno testiranje. Aplikacija je unatoč osvjetljenju uspješno detektirala bljeskove i očitala sve QR kodove, pa se može zaključiti da osvjetljenje prostorije ne utječe na rad aplikacije.

Posljednji set testova je proveden na 3 video zapisa koja su snimana u prostoriji u kojoj je prisutna određena buka. Prilikom snimanja prvog video zapisa, zvučnici na računalu su maksimalno pojačani, a u pozadini se čuju tihi ljudski glasovi. Zvučni signal iz videa je pritom najglasniji zvuk koji je prisutan. Aplikacija je uspješno detektirala sve zvučne signale i vremena

njihove pojave. Drugi video zapis je sniman bez prisutnosti pozadinske buke, ali se u jednom trenutku emitira zvuk glasniji od zvučnog signala iz videa. Aplikacija je u ovom slučaju detektirala dio zvučnih signala, ali ne sve. Problem nastaje zbog prevelike amplitude puštenog zvuka. Treći video zapis je sniman dok je u pozadini puštena glazba koja je tiša od zvučnih signala. U ovom slučaju većina zvučnih signala nije detektirana, a razlog leži u izobličenju sinusne funkcije zbog prisutnosti drugih glasnih zvukova.

Opis video zapisa	Frekvencija zvučnog signala (kHz)	Video zapis sinkroniziran	Razlog nesinkroniziranosti
Kamera pomaknuta kutom manjim od 15 stupnjeva	3	da	Nema
Kamera pomaknuta kutom od 30 stupnjeva	3	ne	Nema
Kamera pomaknuta kutom od 45 stupnjeva	3	ne	QR kodovi nisu pročitani zbog prevelikog kuta snimanja
Kamera pomicana u tijeku snimanja	3	da	Nema
Malo slabije osvjetljenje u prostoriji	3	da	Nema
Slabije osvjetljenje u prostoriji	3	da	Nema
Potpuni mrak u prostoriji	3	da	Nema
Tihi glasovi prisutni za vrijeme snimanja	3	da	Nema
Glasni zvuk prisutan za vrijeme snimanja	3	ne	Glasni zvuk ima znatno veću amplitudu od zvučnog signala
Glazba se čuje tijekom snimanja	3	ne	Glazba uzrokuje distorziju sinusnog signala

Slika 4.3.7. Rezultati obrade video zapisa snimanih u lošijim uvjetima

Zaključak posljednjeg seta testova je da u prostoriji smije biti prisutna određena pozadinska buka, ali ona ne smije biti glasna inače će dovesti do ometanja rada aplikacije. Svi rezultati su vidljivi na slici 4.3.7.

5. Zaključak

U diplomskom radu je osmišljen i implementiran algoritam za testiranje sinkronizacije audia i videa na digitalnim prijemnicima za HbbTV. Algoritam je prvotno napravljen u MATLAB okruženju, a njegova implementacija je odrađena u programskom jeziku C++. Aplikacija koja je nastala kao rezultat ovog rada zadovoljava testnu specifikaciju i uspješno analizira sinkroniziranost. Testiranja su provedena na različitim video zapisima, od kojih su na nekima audiosignal i videosignal bili sinkronizirani, a na nekima ne. Aplikacija je uspješno prepoznala koji su video zapisi nesinkronizirani i detektirala je razloge. Kao jedna od ideja za proširenje funkcionalnosti je da aplikacija svaki video zapis čiji format nije podržan automatski pretvori u .mp4 format.

Diplomski rad je realiziran u suradnji s tvrtkom „Institut RT-RK Osijek d.o.o.“ uz pomoć sumentora iz tvrtke. Aplikacija je integrirana u njihovo testno sučelje kao dinamički povezana C++ biblioteka i služi kao dio šireg skupa testova za HbbTV.

Literatura

[1] HbbTV testna specifikacija https://www.hbbtv.org/wp-content/uploads/2016/12/HbbTV-TG-00065-000-HbbTV_Test_Spec_v9.0_doc_v1.0.pdf

[2] HbbTV logo https://www.hbbtv.org/resource-library/

[3] Izgled HbbTV sučelja http://www.astra2sat.com/hbbtv/

[4] https://www.hbbtv.org/

Sažetak

U sklopu ovog diplomskog rada je napravljena aplikacija za testiranje sinkroniziranosti audiosignala i videosignala u sklopu postupka certifikacije za HbbTV standard. Opisana je i izrada video zapisa na kojima se vrši testiranje. U uvodnim poglavljima su pojašnjeni HbbTV standard i testna specifikacija koju je potrebno zadovoljiti da bi uređaj prošao postupak certifikacije. U završnom poglavlju su definirani testni slučajevi i navedeni rezultati testiranja.

Za izradu aplikacije je korišten programski jezik C++ i biblioteke otvorenog koda na integriranom razvojnom okruženju Visual Studio, a za izradu testnih video zapisa je korišten programski jezik JavaScript i okruženje izvršavanja Node.js. Aplikacija uspješno obavlja testiranje sinkronizacije na velikom broju video zapisa snimanih različitim uređajima. Korisnik kao povratnu informaciju dobije odgovor o uspješnosti testiranja, a u posebnom izvještaju se spremaju vremena detekcije svih zvučnih signala i bljeskova.

Ključne riječi: HbbTV, C++, JavaScript, Node.js, testni slučaj, testna specifikacija, sinkronizacija

Abstract

As a part of this thesis, an application that tests audio and video synchronization as part of certification process for HbbTV standard was developed. The creation of video sequences on which the tests are performed is explained. In introductory chapters, HbbTV standard and test specification are clarified. In final chapter, test cases are defined and test results are mentioned.

C++ programming language and open source libraries for Visual Studio integrated development environment have been used for the development of the application, while JavaScript programming language and runtime environment Node.js have been used for the creation of test sequences. The application successfully performs the testing of synchronization on large number of video sequences, filmed with different devices. The user receives the response whether the testing was successful as a feedback, while detection times of all sound signals and flashes are stored in a special report.

Keywords: HbbTV, C++, JavaScript, Node.js, test case, test specification, synchronization

Životopis

Toni Livaja je rođen 11. rujna 1992. u Osijeku. Pohađao je Osnovnu školu Tenja koju završava s odličnim uspjehom. Završio je III. gimnaziju Osijek, koja radi po programu prirodoslovno-matematičke gimnazije. U tijeku osnovnoškolskog i srednjoškolskog obrazovanja je sudjelovao na desetak županijskih natjecanja iz više predmeta te na jednom državnom natjecanju. 2011. godine upisuje sveučilišni preddiplomski studij računarstva na Elektrotehničkom fakultetu u Osijeku, a 2015. diplomski studij računarstva. U tijeku studija je četiri puta predstavljao fakultet na međunarodnom natjecanju Elektrijada i ostvario zapažene uspjehe, za što je dobio nagradu fakultetskog vijeća 2016. godine. U tijeku diplomskog studija je bio stipendist tvrtke Institut RT-RK Osijek d.o.o.