Sustav za nadgledanje pogrešaka u prikazu živog video sadržaja

Bošnjaković, Ivan

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

Sveučilišni diplomski studij

SUSTAV ZA NADGLEDANJE POGREŠAKA U PRIKAZU ŽIVOG VIDEO SADRŽAJA

Diplomski rad

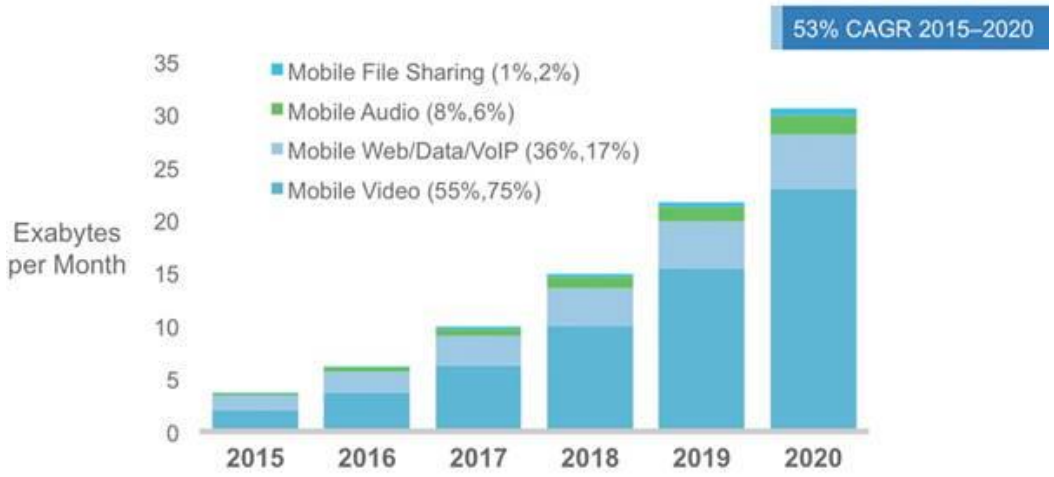
Ivan Bošnjaković

Osijek, 2016.

Sadržaj

1. UVOD

U proteklih nekoliko godina dogodio se značajan napredak u korištenju digitalnog sadržaja, digitalnih fotografija i digitalnog videa. Povećanjem broja mobilnih uređaja korištenje svih vrsta multimedija je drastično poraslo. Prema Cisco-ovim statistikama (Sl. 1.1.) video sadržaj trenutno zauzima više od pola ukupnog podatkovnog prometa pametnih telefona, a procjene za 2020. govore kako će tada gotovo 75 posto podatkovnog prometa biti video sadržaj [1].



Sl. 1.1. Cisco statistike - predviđanje mobilnog prometa [1].

Može se očekivati kako će s vremenom rasti i brzina podatkovnog prometa koju davatelji usluge podržavaju, a time i potražnja za multimedijom visoke kvalitete kao što je FHD (engl. *full high-definition*) video, razlučivosti 1920x1080 elemenata slike ili veće razlučivosti UHD (engl. *ultra-high-definition*), razlučivosti 4K 3840x2160, 8K 7680×4320 elemenata slike i dr. Osim toga pretpostavke su da će s vremenom i mobilni operateri smanjiti cijene usluga, učiniti ih dostupnijim korisnicima, a time će se onda dodatno povećati potražnja za kvalitetnim multimedijskim sadržajem. Sve navedeno nagovještava da će u budućnosti glavnina pojasne širine biti zauzeta preuzimanjem i postavljanjem video sadržaja.

Ovakvo povećanje korištenja multimedijskog sadržaja prirodnim slijedom će dovesti do povećanja interesa za boljom kvalitetom videa ne samo u smislu veće razlučivosti nego za kvalitetnijim video sadržajem općenito. Na kvalitetu video sadržaja može utjecati mnogo toga. Na kvalitetu utječe postupak snimanja videa, kompresija, prijenos videa, reprodukcija itd. U prijenosu i reprodukciji video sadržaja sudjeluju različite strane: mrežni operateri, proizvođači

video opreme, različite tvrtke koje se bave istraživanjem i razvojem kao što su tvrtke za razvoj kodeka i slično.

Kako bi krajnjem korisniku povećali kvalitetu usluge sve strane imaju potrebu za alatima s kojima će provjeravati kvalitetu svojih proizvoda, hardvera, softvera, tijekom faze razvoja, proizvodnje i faze održavanja. Npr. kod mrežnih pružatelja usluga postoji potreba za nadzorom kvalitete usluge QoS (engl. *quality of service*) u stvarnom vremenu kako bi mogli pratiti kvalitetu usluge kod krajnjeg korisnika. Nakon toga u ovisnosti o dobivenim rezultatima pružatelji usluga mogu podešavati parametre mreže kako bi korisniku isporučili uslugu onakve kvalitete kakva je dogovorena prilikom zaključivanja ugovora između korisnika i pružatelja usluga. Osim testiranja i podešavanja mreže nadzor se može provoditi i na drugim dijelovima gdje je ugovorom dogovorena kvaliteta videa. Analiza se također može provoditi i na uređajima za prikazivanje video sadržaja kao što su digitalni TV prijamnici, DTV uređaji i *set-top box* uređaji [2, 3].

Postoje dvije kategorije metoda kojima se može ocijeniti kvaliteta nekog videa:

- subjektivne metode
- objektivne metoda

Subjektivne metode svode se na testiranja u kontroliranim uvjetima gdje više osoba gleda određene video sekvence i u ovisnosti o osobnom dojmu ocjenjuju kvalitetu videa. Takva testiranja kvalitete su vrlo zahtjevna i zahtjevaju mnogo vremena pa nisu pogodna i nemoguće ih je primijeniti kod automatskog testiranja kvalitete videa.

Za razliku od subjektivnih metoda, objektivne metode ocjenjivanja baziraju se na različitim metrikama koje automatski mogu procijeniti kvalitetu videa odnosno percipiranu kvalitetu razmatranog videa.

Objektivne metode ocjene kvalitete videa dijele se u tri skupine: metode ocjene kvalitete videa s potpuno dostupnim referentnim videom (engl. *full reference methods – FR methods*), metode ocjene kvalitete videa s djelomično dostupnim informacijama o referentnom videu (engl. *reduced reference methods – RR methods*) i metode ocjene kvalitete videa bez dostupnih informacija o referentnom videu (engl. *no-reference, NR methods*). Najčešći je slučaj da referentni video nije dostupan pa je ocjena kvalitete moguća samo NR metodom kao što je slučaj kod testiranja kvalitete videa koji je dobiven preko emitirajuće mreže kao što je zemaljska televizija. Pristup bez referentnog testiranja svodi se na pronalazak poznatih svojstva artefakata

koji se mogu pojaviti u video sadržaju kao rezultat različitih distorzija tijekom produkcije i prijenosa videa [3].

Sve veći zahtjevi za kvalitetnim videom povećavaju potrebu i važnost postojanja pouzdanog alata za procjenu kvalitete videa. U zadnjih desetak godina problematikom testiranja video kvalitete su se bavili mnogi znanstvenici i tvrtke koje se bave razvojem videa. Zainteresiranost na ovom području postoji i dalje u znanstvenim krugovima međutim praktična rješenja i programski alati za procjenu video kvalitete su i dalje rijetki, a takvi alati bi bili neprocjenjivi za znanstvenike, korisnike i tvrtke koje se bave istraživanjem i razvojem na području videa.

U ovom radu je prezentirana struktura i mogućnosti aplikacije za detektiranje video artefakata nazvana PQM (engl. *picture quality meter*). Aplikacija se zasniva na bez referentnoj metodi detektiranja artefakata, a može se koristiti za analizu raznih tipova video kontejnera ili za testiranje emitirajućih sustava kao što je zemaljska televizija koristeći *grabber* uređaj. Prezentirana aplikacija omogućava detekciju i nadzor artefakata u stvarnom vremenu i praćenje rezultata analiziranih video sekvenci. PQM aplikacija ima mogućnost proširivanja algoritmima za detekciju artefakata jednostavnim dodavanjem odnosno uklanjanjem odgovarajućih dinamičkih biblioteka. Također prije same analize korisnik ima mogućnost izbora s kojim raspoloživim algoritmima želi provesti analizu. Rezultati analize spremaju se u strukturiranom formatu kako bi se kasnije mogli analizirati.

Rad je strukturiran na sljedeći način. U drugom poglavlju najprije su definirani osnovni pojmovi poput video artefakta te su ukratko opisani najznačajniji video artefakti. U trećem poglavlju dan je pregled postojećih aplikacija za analizu kvalitete videa. Nakon toga, u četvrtkom potpoglavlju opisana je programska implementacija PQM aplikacije, nabrojani su zahtjevi aplikacije, opisan je dizajn i implementacija, u petom poglavlju opisana je gotova PQM aplikacija, a na kraju u zaključku rada dan je osvrt na cijeli rad.

2. DIGITALNI VIDEO ARTEFAKTI

Bez referentno ispitivanje kvalitete videa može se bazirati na analizi elemenata slike ili na analizi kodiranog toka bitova (engl. *bit stream*). Najčešće nije moguće pristupiti toku bitova pa takva analiza nije moguća. U većini slučajeva jedino na čemu se može analizirati video je dobiveni video okvir (engl. *frame*). U tom slučaju primjenom algoritama analiziraju se elementi slike kako bi se pronašle pravilnosti koje definiraju poznate oblike distorzija. Poznate distorzije slike nazivamo video artefaktima. U nastavku su nabrojani najčešći artefakti koji se obično istražuju prilikom NR procjene kvalitete videa:.

- Pojava blokova (engl. *blocking*),
- Zamrzavanje (engl. *freezing*),
- Crni ekran (engl. *black screen and blanking*),
- Gubitak paketa (engl. *packet loss*),
- Šum (engl. *noise*),
- Krivo osvjetljenje i gubitak boja (engl. *brightness and color fading*),
- Kolebanje (engl. *image jitter*),
- Kidanje (engl. *tearing*), te
- Zamagljenost i mreškanje (engl. *blurring and ringing*).

Pojava artefakata na video sadržaju ne mora ujedno značiti da je i video loše kvalitete, objektivno gledajući video može sadržavati određeni broj artefakata, ali da kvaliteta tj. dojam koji se dobiva gledajući ga bude vrlo dobar čak i odličan. Sve ovisi o poziciji artefakata i sadržaju konkretnog videa. Ponekad su artefakti izraženiji, ponekad ne. Zbog toga se samim detektiranjem artefakata ne može odrediti MOS (engl. *mean opinion score*), odnosno dojam koji kvaliteta videa ostavlja na gledatelja. Kako bi se odredio MOS potrebno je uzeti i druge čimbenike u obzir. Detektirani artefakti su samo jedan čimbenik u ocjenjivanju kvalitete videa, ali sama objektivna metoda kao što je detekcija artefakata nikako ne može dati globalnu ocjenu kvalitete videa.

2.1. Pojava blokova

Artefakt pojave blokova događa se kao rezultat kompresije s gubitkom. Npr. kod MPEG-2 ili H.26x video kompresija slika se dijeli na segmente u obliku blokova nad kojima se provodi kompresija. Tijekom kompresije nepovratno se gubi dio informacije što uzrokuje diskontinuitete

na prijelazima između dva bloka. Na taj način dolazi do pojave blok artefakta koji se manifestiraju kao vidljivi blokovi na slici (Sl. 2.1.) [3, 5].

Sl. 2.1. Artefakt - Pojava blokova.

2.2. Zamrzavanje

Do zamrzavanja slike dolazi zbog povremenih problema u prijenosnom kanalu tj. mreži preko koje pružatelj usluga distribuira video signal. Pojava zamrzavanja slike najčešće nastaje kada su oštećenja na dobivenom video signalu prevelika pa dekoder ne može ispravno dekodirati te prikazuje zadnji video okvir koji je ispravno primio. Pojava zamrzavanja slike nastaje i kada prijem video okvira vremenski kasni [3, 4, 6].

2.3. Pojava crnog ekrana

Pojava crnog ekrana (gubitak slike) je artefakt sličan zamrzavanju slike međutim kod ovog artefakta specifično je to što se umjesto zamrznute slike prikazuje ekran u jednoj boji tijekom nekog vremena (Sl. 2.2.). Najčešće u crnoj boji [4].

Sl. 2.2. Artefakt - Pojava crnog ekrana (gubitak slike) [4].

2.4. Gubitak paketa

Gubitak paketa događa se kada je prijenosni kanal na trenutak prekinut zbog pojave smetnji ili je došlo do zastoja na čvorovima unutar mreže. Budući da se jedan okvir videa prenosi preko više paketa, kada dođe do kratkog prekida ili smetnji dio paketa se gubi ili zamijeni. Rezultat ovog artefakta je video okvir s pravokutnim ne pripadajućim dijelovima. Ne pripadajući dijelovi mogu biti dio nekog drugog video okvira kao što je prikazano na slici 2.3. ili nešto neodređeno. Ovisno na koji način uređaj za reprodukciju videa rješava probleme vezane uz gubitak paketa [2, 4].

Sl. 2.3. Artefakt - Gubitak paketa.

2.5. Šum

Šum se događa kada se na nekim elementima video okvira dogodi pomak vrijednosti. Vrijednost elementa video okvira znatno odstupa od okoline u kojoj se nalazi. Rezultat takve razlike je artefakt koji se manifestira kao brašnjavost slike (Sl. 2.4.) [4].

Sl. 2.4. Artefakt - Šum na slici [4].

2.6. Neodgovarajuće osvjetljenje i gubitak boja

Neodgovarajuće osvjetljenje i gubitak boja događa se zbog pogreški u zapisu video okvira u YUV formatu. Neodgovarajuće vrijednosti Y komponente (luminantna) uzrokuju neodgovarajuće osvjetljenje prilikom reprodukcije video sadržaja, dok pogrešne vrijednosti U komponente (krominantna) i V komponente (krominantna) uzrokuju pogrešan prikaz boje (Sl. 2.5.) [4].

Sl. 2.5. Artefakt - Neodgovarajuće osvjetljenje i gubitak boja [4].

2.7. Kolebanje slike

Kolebanje slike rezultat je mehaničke prirode i događa se zbog nedovoljno dobre sinkronizacije, problema s naponom ili zbog nekih drugih poremećaja analogne prirode kao što su interferencije tijekom video prijenosa. Artefakt se manifestira kao nepravilno razbacani elementi slike ili razbacane linije [3, 4].

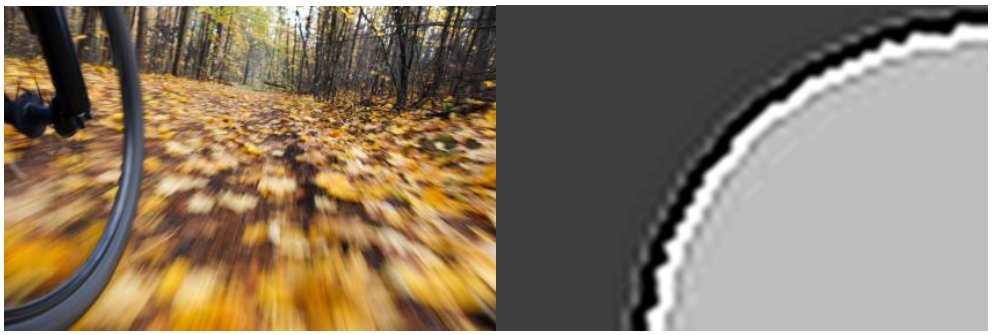
2.8. Kidanje slike

Kidanje slike događa se u situacijama kada se brzina osvježavanja izvora razlikuje u odnosu na brzinu osvježavanja ekrana. Kada se te dvije brzine osvježavanja razlikuju dolazi do nesinkroniziranosti koja se na videu manifestira kao kidanje slike (Sl. 2.6.).

Sl. 2.6. Artefakt - Kidanje slike [4].

2.9. Zamagljenost i mreškanje slike

Artefakti zamagljenosti i mreškanja slike događaju se u videu zbog kvantizacije visoko frekventnih komponenti video okvira. Kao rezultat artefakta zamagljenosti su zamagljeni dijelovi video okvira i gubitak oštrine rubnih dijelova. Kod rubnih dijelova misli se na rubne dijelove komponenti prikazanih na slici. Mreškanje slike manifestira se na rubovima gdje postoji veliki kontrast, a rezultat ovog artefakta je nejasnoća rubnih dijelova tj. mreškanje (Sl. 2.7.) [3, 4].



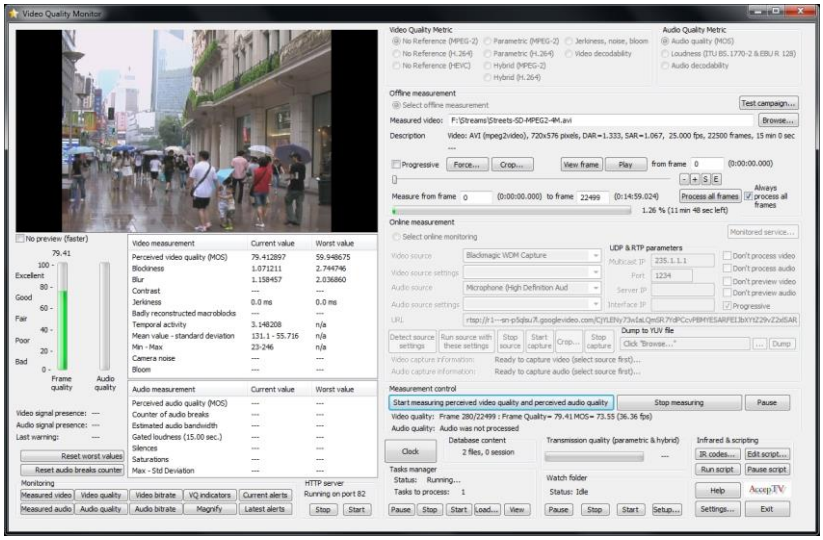
Sl. 2.7. Artefakt - Zamagljenost (lijevo) [9] i mreškanje slike (desno) [10].

3. POSTOJEĆE APLIKACIJE ZA ANALIZIRANJE KVALITETE VIDEA

Na tržištu postoji nekoliko aplikacija za bez referentno ocjenjivanje video kvalitete. Neke od istaknutijih su aplikacija *Video Quality Monitor* (VQM), aplikacija koju razvija tvrtka *AcceptTV* i *Video Quality Measurement Tool* (VQMT), aplikacija koja se razvija na Sveučilištu u Moskvi.

3.1. Aplikacija Video Quality Monitor (VQM)

VQM aplikacija (Sl. 3.1.) ima brojne korisne mogućnosti kada se radi o bez referentnoj metodi analiziranja video kvalitete. Aplikacija ima razne mogućnosti kao što je testiranje brzine kodiranja videa (engl. *video encoders benchmarking*) i usporedbu brzina, nadzor videa koji je u prijenosu uživo (engl. *live*), optimizacija video procesiranja, nadzor procesa kodiranja videa u video datoteku, određivanje optimalne brzine prijenosa (engl. *bitrate*) i glasnoće zvuka. Aplikacija podržava sljedeće formate: HEVC (H.265), MPEG-4/AVC (H.264), MPEG-2, nekompresirane formate i druge. Ima mogućnosti za analiziranje prijenosa uživo (engl. *online*) i video datoteka tj. izvan-mrežnu analizu (engl. *offline*). VQM aplikacija kao rezultat daje broj sa MOS skale koji predstavlja doživljenu kvalitetu videa od strane gledatelja. Ova ocjena kvalitete videa temelji se na detekciji nekih video artefakata kao što su: detekcija pojave blokova, zamagljenosti, krivog osvjetljenja i gubitka boja, zamrzavanja, pojavu crnog ekrana i mogućnosti za mjerenje kvalitete zvuka.



Sl. 3.1. Video Quality Monitor [7].

3.2. Aplikacija Video Quality Measurement Tool (VQMT)

VQMT aplikacija je aplikacija za mjerenje video kvalitete koja integrira potpuno referentnu metodu i bez referentnu metodu, podržava razne video formate i formate za sliku. Metrike u bez referentnoj metodi temelje se na detektiranju pojave blokova, detektiranju zamagljenosti, krivog osvjetljenja, detektiranju gubitka okvira, promjenu scene i procjeni šuma. Rezultati analize spremaju se u CVS (engl. *comma-separated values*) formatu, a vizualizacija se sprema u video datoteci. Glavna namjena aplikacije je u području analize kodeka i testiranju propusnosti (engl. *benchmarking*).

3.3. Nedostaci postojećih aplikacija

Spomenute aplikacije imaju korisne mogućnosti, ali postoje i dijelovi na kojima bi se moglo raditi kako bi alati bili korisniji ciljanim korisnicima kao što su tvrtke za istraživanje i razvoj (engl. *research and development, R&D*), tvrtkama za testiranje video opreme i nadzor kvalitete video usluge.

Preciznije govoreći, uočeni su sljedeći nedostatci:

- Opcija za promjenu vrijednosti parametara algoritmima za bez referentno analiziranje,
- Opcija za dodavanje algoritama za analizu video sadržaja,
- Stvaranje plana s kojim bi korisnik bio u mogućnosti postaviti datum i vrijeme kada želi da se započne analiza odabranog video sadržaja,
- Vizualizaciju detektiranih video artefakata u stvarnom vremenu, te
- Pohranjivanje artefakata u strukturiranom formatu radi daljnje analize i istraživanja.

4. PQM APLIKACIJA

PQM aplikacija je aplikacija za bez referentnu detekciju artefakata u video sadržaju. Ona omogućava analizu video sadržaja tj. detektiranje artefakata. Samim detektiranjem artefakata ne može se odrediti kvaliteta videa, ali se detektirani artefakti mogu iskoristiti kao jedan parametar prilikom određivanja kvalitete videa. Sama izrada PQM aplikacije krenula je s ciljem izrade praktičnog alata za korisnike koji imaju potrebu analizirati video sadržaje ili one korisnike koji rade na razvoju algoritama za bez referentnu detekciju artefakata.

U nastavku rada najprije će biti definirani i pojašnjeni svi zahtjevi za aplikaciju koji su postavljeni u fazi planiranja izrade aplikacije.

4.1. Zahtjevi

Zahtjevi su kreirani na temelju iskustva ljudi iz tvrtke RT-RK Novi Sad koji iza sebe imaju godine iskustva na području analize videa. Konzultirajući se s ljudima iz tvrtke i korisnicima što oni očekuju od aplikacije za analizu videa, što bi bilo praktično, bolje, što fali postojećim aplikacijama itd., napravljen je popis zahtjeva koji bi bili interesantni korisnicima. U nastavku teksta nabrojani su i objašnjeni glavni zahtjevi postavljeni prije faze izrade (implementacije) aplikacije u odgovarajućem programskom jeziku.

Aplikacija mora imati mogućnost kreiranja dva tipa projekta: izvan-mrežni projekt i mrežni projekt. Kako je glavna namjena PQM aplikacije detekcija različitih artefakata u videu, projekt mora imati mogućnost pokretanja analize na odabranim video datotekama što podrazumijeva dekodiranje različitih video kontejnera, provedbu analiziranja videa dostupnim algoritmima te spremanje rezultata dobivenih analizom. Izvan-mrežni projekt mora omogućavati analizu video datoteka tj. dekodiranje različitih video kontejnera, analiziranje okvira i spremanje rezultata na disk. Mrežni tip projekta mora omogućavati analizu udaljenog video sadržaja koji se dohvaća pomoću *grabber-a* i spremanje rezultata na disk. Aplikacija također mora imati mogućnost dinamičkog dodavanja algoritamskih biblioteka s kojima je moguće provoditi analizu odabranih video datoteka. Algoritamske biblioteke moraju imati jedinstveno programsko sučelje API (engl. *application programming interface*) kako bi ih bilo moguće dinamički dodavati i uklanjati. Prilikom analize video sadržaja korisnik mora imati mogućnost praćenja rezultata i pregled trenutno analiziranog okvira. Sučelje bi trebalo biti relativno jednostavno, intuitivno korisniku i mora imati funkcionalnosti sa popisa zahtjeva u nastavku.

Glavne funkcionalnosti koje aplikacija mora omogućavati:

- Kreiranje projekta,
- Dodavanje postojećeg projekta,
- Brisanje projekta,
- Otvaranje projekta,
- Dodavanje ulaza projektu,
- Dodavanje algoritamske biblioteke,
- Brisanje algoritamske biblioteke,
- Odabir ulaza za analizu,
- Odabir algoritamskih biblioteka za analizu,
- Analiziranje ulaza,
- Prikaz rezultata,
- Prikaz analiziranog okvira, te
- Spremanje rezultata analize u strukturiranom formatu.

4.1.1. Kreiranje, dodavanje i brisanje projekta

Prilikom kreiranja projekta korisnik u aplikaciji odabire opciju za kreiranje projekta. Odabire putanju gdje želi spremiti projekt na čvrsti disk te odabire naziv projekta i odabire jedan od dva moguća tipa projekta. Ovisno o tipu projekta projekt podržava analizu video datoteka na lokalnom računalu ili video sadržaja koji se dohvaća putem *grabber* uređaja. Nakon što se projekt kreira dodaje se na popis dostupnih projekata unutar aplikacije.

Kod dodavanja postojećeg projekta, korisnik u aplikaciji odabire opciju dodavanja postojećeg projekta pri čemu mora definirati putanju do postojećeg projekta na čvrstom disku. Nakon dodavanja projekta projekt se dodaje na popis dostupnih projekata.

Ako korisnik želi obrisati postojeći projekt, korisnik odabire projekt sa popisa projekata i odabire opciju za brisanje projekta pri čemu će se obrisati svi podaci pohranjeni na disku vezani za ovaj projekt. Osim brisanja projekta postoji i opcija uklanjana projekta sa popisa dostupnih projekata. Ovom opcijom projekt se ne briše sa lokalnog računala nego se prekida veza sa projektom. Projekt postoji na disku, ali više nije vidljiv na popisu dostupnih projekata.

4.1.2. Otvaranje projekta

Korisnik na popisu projekata odabire projekt s kojim želi raditi i može ga otvoriti opcijom za otvaranje projekta. Nakon otvaranja projekta korisnik ima mogućnost dodavanja ulaza, pregledavanja rezultata, odabir ulaza za analizu te pokretanja analize na odabranim ulazima.

4.1.3. Dodavanje ulaza

Ovisno o tipu projekta korisnik može dodati izvan-mrežni i mrežni ulaz. Ako je korisnik kreirao i radi na izvan-mrežnom projektu tada u projekt može dodati samo izvan-mrežni ulaz. Izvan-mrežni ulazi su video datoteke pohranjene na lokalnom računalu. Korisnik dodaje izvan-mrežni ulaz tako što odabere opciju dodavanja ulaza i postavi putanju do video datoteke.

Ako korisnik radi na mrežnom projektu, tada u projekt može dodati samo jedan mrežni ulaz. Mrežni ulaz je definiran mrežnim postavkama za *grabber* uređaj koji se nalazi na udaljenom mjestu i koji preko mreže šalje dohvaćene video okvire aplikaciji na analizu.

4.1.4. Dodavanje i brisanje algoritamskih biblioteka

Algoritamske biblioteke za detektiranje različitih artefakta korisnici mogu dinamički dodavati i uklanjati opcijama za rukovanje datotekama. Algoritamska biblioteka je u obliku DLL (engl. *dynamic-link library*) algoritamskim bibliotekama. Dodavanje odnosno brisanje algoritamskih biblioteka iz PQM aplikacije svodi se na dodavanje odnosno brisanje odgovarajućih DLL datoteka u odgovarajućem direktoriju na disku. Naime, aplikacija dinamički detektira promjene, ako je algoritamska biblioteka dodana dodaje, a ako je obrisana onda uklanja funkcionalnosti iz PQM aplikacije koju dodana/obrisana biblioteka pruža. Algoritamski moduli su definirani preko jedinstvenog API-a kako bi dinamičko dodavanje i brisanje bilo moguće za različite tipove algoritama za detekciju artefakata.

4.1.5. Odabir ulaza, modula i analiziranje ulaza

Nakon što korisnik doda ulaze i algoritamske biblioteke u projekt ima mogućnost odabira jednog ili više ulaza nad kojima želi provesti analizu i odabir jednog ili više algoritama s kojima želi provesti analizu videa. Nakon odabira ulaza i biblioteka korisnik može pokrenuti analizu videa.

4.1.6. Prikaz rezultata i prikaz analiziranog okvira

Nakon što korisnik odabere ulaze, algoritamske biblioteke i odabere opciju za pokretanje analize, analiza je pokrenuta. Na ekranu se prikazuje okvir kojeg aplikacija trenutno analizira, rezultati se ispisuju na ekranu i spremaju se u datoteku u strukturiranom formatu.

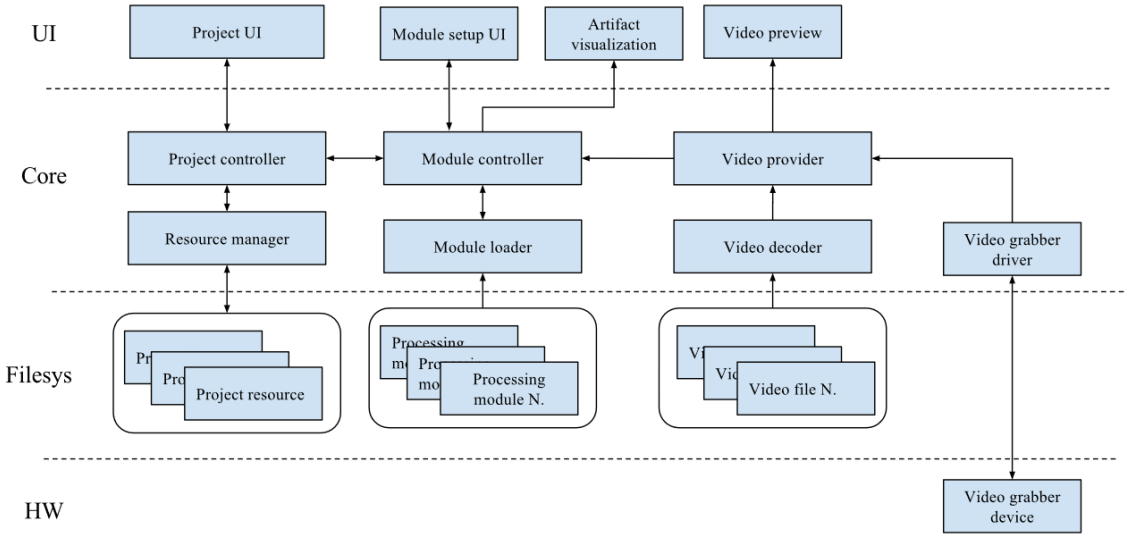
4.1.7. Spremanje rezultata analize u strukturiranom formatu

Kako bi se omogućilo jednostavno učitavanje rezultata u druge alate, spremanje je moguće provesti u standardnim zapisima poput XML-a ili CSV-a.

Ovim popisom su definirani najvažniji zahtjevi aplikacije koji obuhvaćaju temu rada. U nastavku teksta detaljnije je opisan dizajn aplikacije. U potpoglavlju dizajn aplikacije 4.2. dana je gruba slika kako je aplikacija prije implementacije osmišljena. Detaljnija razrada dizajna objašnjena je u potpoglavlju implementacija aplikacije 4.3.

4.2. Dizajn aplikacije

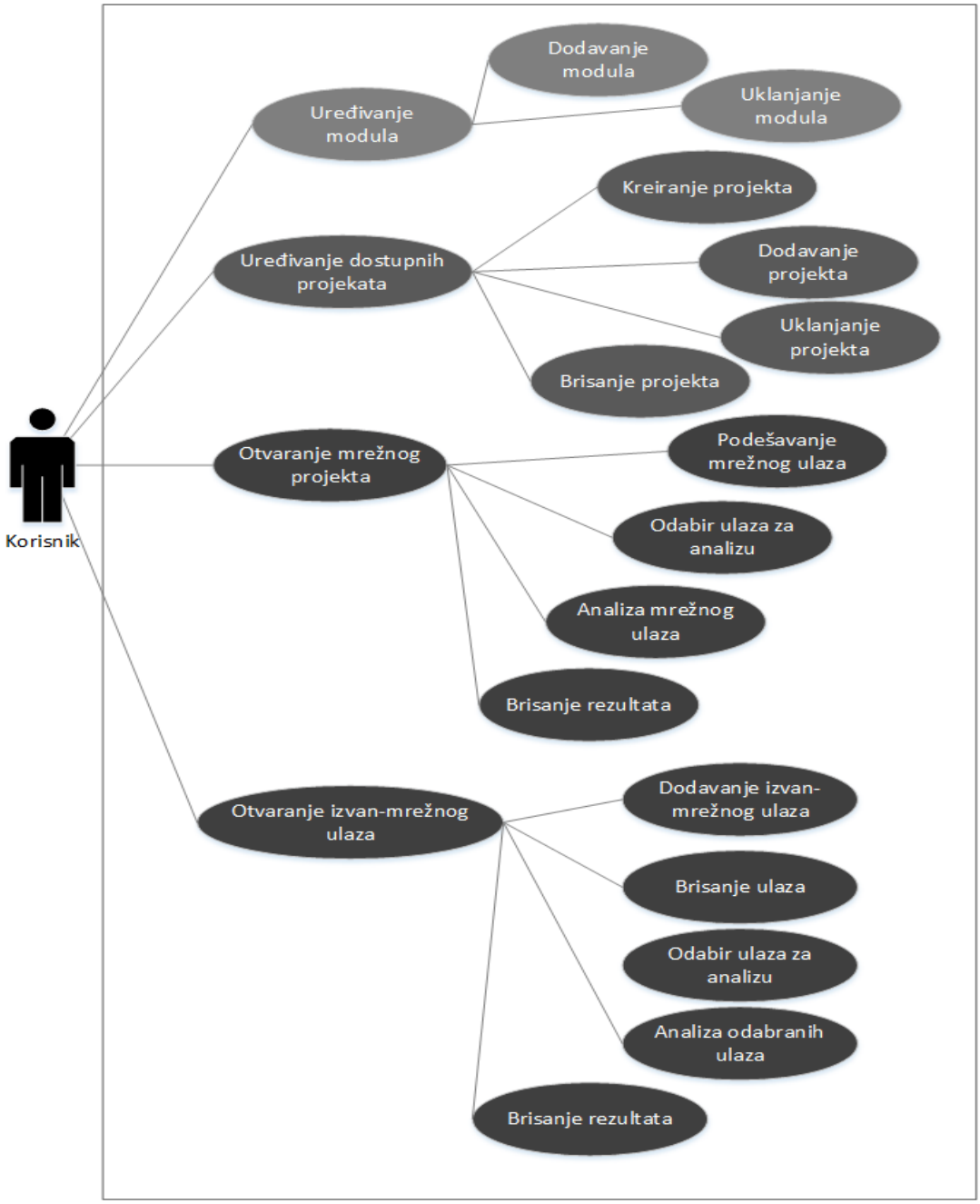
Potpoglavlje o dizajnu aplikacije podijeljeno je u pet točaka: sučelje aplikacije, dijagram slučajeva, funkcionalnosti aplikacije (pozadina aplikacije), povezivanje funkcionalnosti aplikacije sa sučeljem aplikacije i datotečni sustav. Na slici 4.1. prikazana je struktura aplikacije. Struktura aplikacije je podijeljena na četiri glavne razine: razina hardvera, razina datotečnog sustava, razina s funkcionalnostima aplikacije (pozadina aplikacije) i razina korisničkog sučelja aplikacije.



Sl. 4.1. Struktura aplikacije.

4.2.1. Dijagram slučajeva

Na slici 4.3. prikazan je dijagram slučajeva aplikacije. Korisničko sučelje aplikacije ima opciju za uređivanje algoritamskih biblioteka (modula). Prilikom uređivanja modula može dodavati i uklanjati module koji će biti dostupni u aplikaciji. Korisnik također može uređivati popis dostupnih projekata na listi projekata. Projekte može kreirati, dodavati, uklanjati, brisati. Ima opciju za otvaranje nekog projekta sa popisa projekata. Nakon otvaranja mrežnog projekta, korisnik može podešavati mrežni ulaz, odabrati ulaz za analizu, provesti analizu ulaza s odabranim modulima i obrisati rezultat analize. Ako se radi o izvan-mrežnom projektu, korisnik može dodavati ulaze, odabrati ulaze za analizu, provesti analizu ulaza s odabranim modulima i ima opciju brisanja rezultata analize.



Sl. 4.2. Dijagram slučajeva.

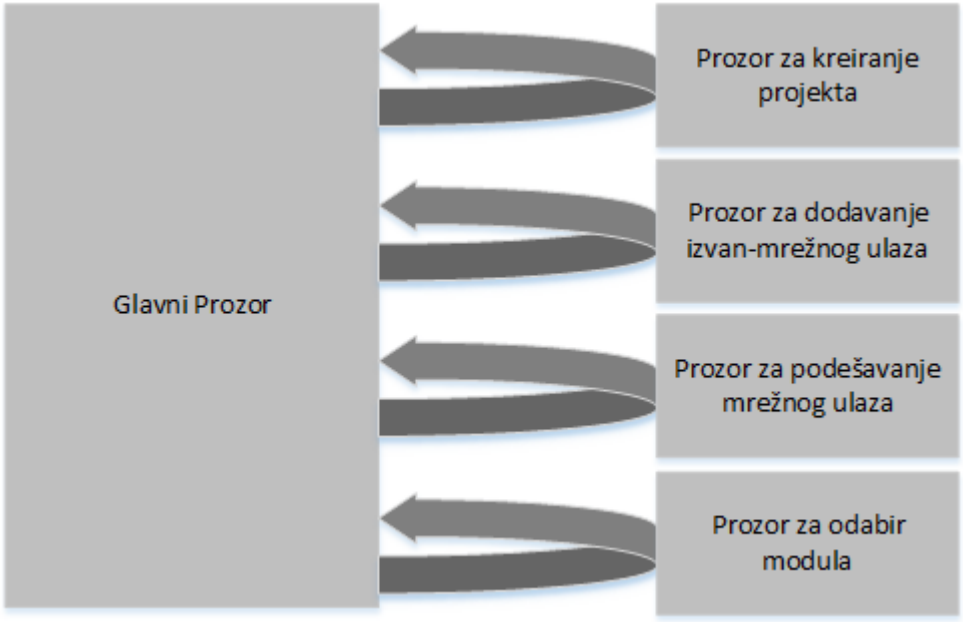
4.2.2. Korisničko sučelje aplikacije

Korisničko sučelje (engl. *graphical user interface - GUI*) aplikacije sastoji se od nekoliko zasebnih prozora. To su redom:

- Glavni prozor,
- Prozor za kreiranje projekta,
- Prozor za dodavanje izvan-mrežnog ulaza,
- Prozor za podešavanje mrežnog ulaza, te

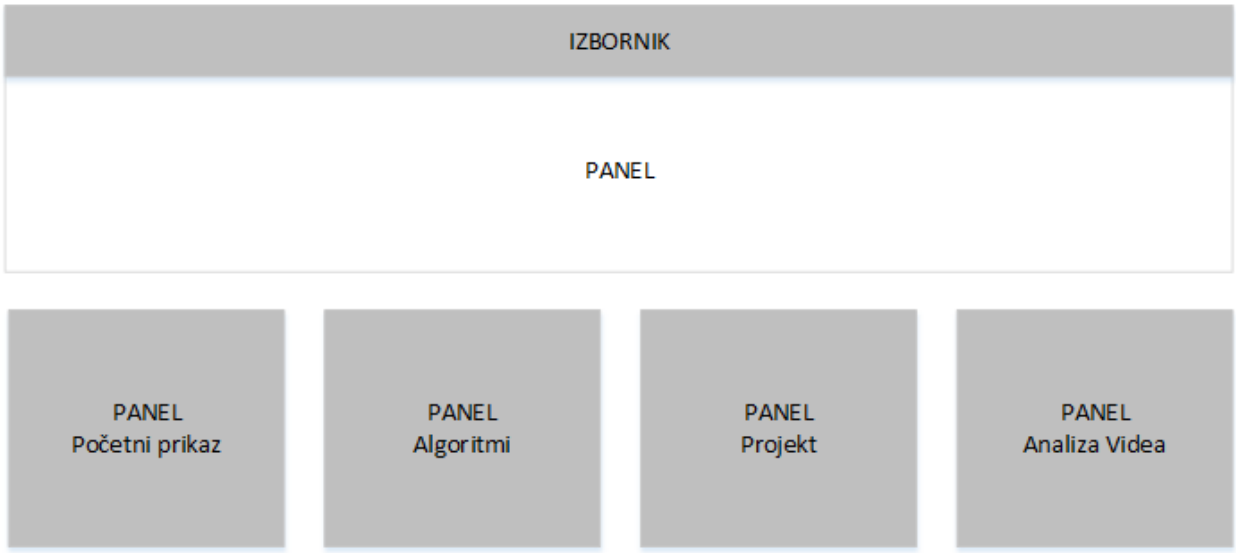
- Prozor za odabir modula koji će se koristiti u analizi.

Prozori za kreiranje projekta, dodavanje izvan-mrežnog ulaza, podešavanje mrežnog ulaza i prozor za odabir modula sporedni su prozori koji se otvaraju po potrebi. Tijekom rada aplikacije glavni prozor je stalno otvoren i u njemu se odvijaju svi glavni dijelovi aplikacije te on rukuje sa sporednim prozorima.



Sl. 4.3. Povezanost korisničkog sučelja aplikacije i smjer komunikacije.

Na slici 4.3. može se vidjeti smjer odvijanja komunikacije između glavnog prozora i ostalih prozora. Glavni prozor otvara sporedne prozore, a sporedni prozori vraćaju definirane strukture podataka glavnom prozoru. Što je detaljnije opisano u potpoglavlju 4.3.



Sl. 4.4. Glavni prozor aplikacije (*multipanel* dizajn).

Glavni prozor aplikacije je podijeljen u dvije glavne cjeline: izbornik i prostor za panele (Sl. 4.4.). U izborniku se nalaze opcije za prebacivanje iz jednog panela u drugi. Opcije u izborniku su definirane u ovisnosti o stanju u kojem se aplikacija nalazi.

Početni izbornik sadrži sljedeće opcije u izborniku:

- *Home* - pritiskom na ovu opciju u glavnom se prozoru prikazuje panel koji definira početni prikaz.
- *File*
    - *New project* - pritiskom na ovu opciju otvara se sporedni prozor za kreiranje projekta.
    - *Add project* - pritiskom na ovu opciju otvara se sporedni prozor za definiranje putanje do postojećeg projekta.
- *PQM*
    - *Algorithms* - pritiskom na ovu opciju u glavnom prozoru prikazuje se panel za dodavanje i brisanje algoritamskih biblioteka.

Nakon što korisnik otvori neki projekt mijenjaju se opcije u izborniku.

Opcije u izborniku nakon što korisnik otvori projekt:

- *File*
    - *Close project* - pritiskom na ovu opciju zatvara se projekt i prikazuje se panel koji definira početni prikaz.
- *Project* - pritiskom na ovu opciju otvara se panel projekt u kojem korisnik ima pregled svih resursa u projektu. Pod resursima projekta podrazumijevaju se svi ulazi i rezultati analiza
- *PQM*
    - *Algorithms* - pritiskom na ovu opciju u glavnom prozoru prikazuje se panel za dodavanje i brisanje algoritamskih biblioteka.
    - *Anaylze input* - pritiskom na ovu opciju otvara se panel u kojem korisnik može pokrenuti i pratiti analizu odabranih ulaza.

Panel koji definira početni prikaz, u nastavku teksta PH (engl. *panel home*), sastoji se od prostora u kojem se nalazi popis projekata povezanih sa aplikacijom. Osim popisa projekata početni prikaz sadrži gumbe, opcije s kojima korisnik može otvoriti, ukloniti ili obrisati projekte.

Panel algoritmi, u nastavku teksta PA (engl. *panel algorithms*), sastoji se od popisa na kojemu je prikazano koje su sve algoritamske biblioteke uključene u PQM aplikaciji. Osim popisa sadrži dva gumba pritiskom na prvi gumb otvara se direktorij u kojem se nalaze algoritamske biblioteke. Ovdje korisnik može dodavati, uklanjati biblioteke koristeći osnovne opcije za dodavanje, uklanjanje datoteka. Osim gumba za otvaranje direktorija s algoritamskim bibliotekama, postoji drugi gumb s kojim korisnik osvježava popis uključenih algoritama.

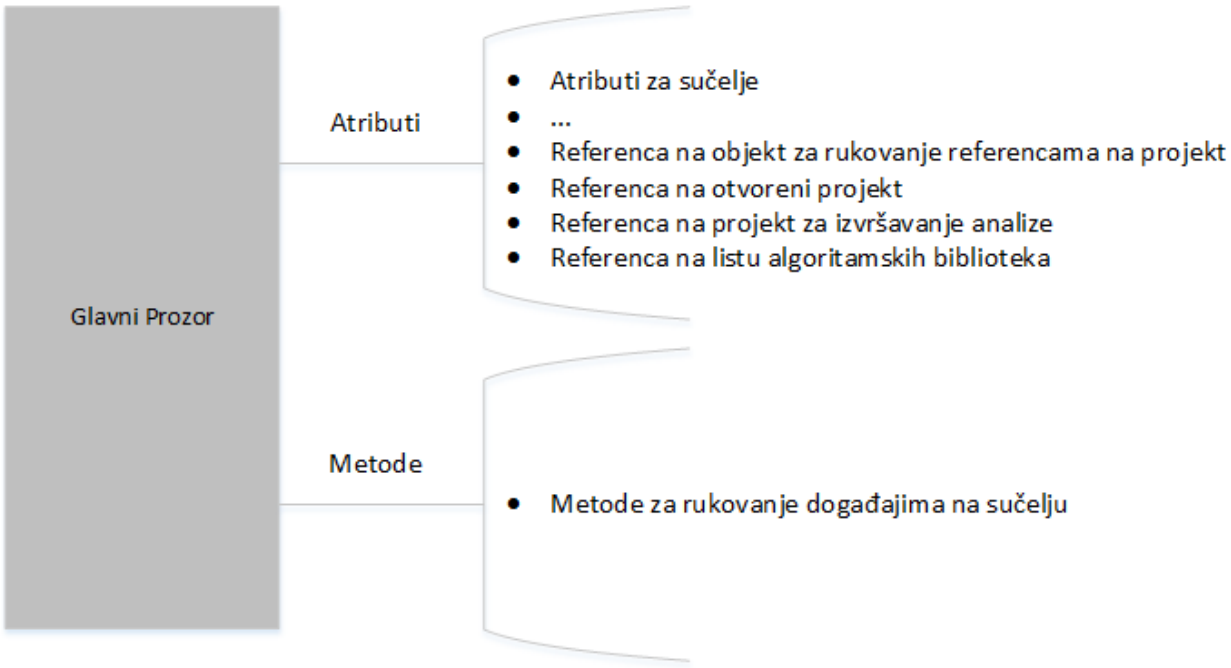
Panel projekt, u nastavku teksta PP (engl. *panel project*), podijeljen je u tri cjeline. Prvu cjelinu čine informacije o projektu. Ovdje je prikazan naziv projekta, putanja do projekta, veličina na projektu. Druge dvije cjeline su dio za rukovanje ulazima i dio za rukovanje rezultatima. Kod dijela za rukovanje ulazima nalazi se popis ulaza i opcije (gumbi) ovisno o tipu projekta. Ako se radi o izvan-mrežnom projektu tada postoji opcija za dodavanje ulaza i opcija za analiziranje odabranih ulaza. Ako se radi o mrežnom projektu tada postoji opcija za izmjenjivanje postavki *grabber-a* i opcija za analiziranje video sadržaja dohvaćenog pomoću *grabber-a*. Za oba tipa projekta isti je dio za rezultate, ovaj dio se sastoji od popisa rezultata i gumba s kojim se prikazuju odabrani rezultati.

U panelu za analizu videa, u nastavku teksta PAV (engl. *panel analyze video*), postoje opcije za pokretanje i zaustavljanje analize i dio u kojemu se vidi okvir ulaza koji se procesira.

Ovime je dan pregled glavnih dijelova aplikacije kada su u pitanju dijelovi za sučelje. U potpoglavlju 4.3. i poglavlju 5. biti će još govora o njima.

4.2.3. Funkcionalnosti aplikacije

Kao glavni dio u kojem se odvija cijeli program je glavni prozor. Glavni prozor je objekt koji ima svoje atribute i metode (Sl. 4.5.). Većina atributa ovog objekta definira izgled aplikacije i oni nisu detaljno opisani u radu.



Sl. 4.5. Atributi i metode glavnog prozora.

Osim atributa koji definiraju sučelje aplikacije postoje i atributi za funkcionalnosti aplikacije kao što su:

- Referenca na objekt koji rukuje referencama na projekt(e) ,
- Referenca na otvoreni projekt,
- Referenca na objekt za izvršavanje analize,
- Referenca na listu algoritamskih biblioteka.

Atribut objekta glavnog prozora je referenca na objekt koji rukuje poveznicama na projekte u nastavku teksta ROPD (engl. *object reference program data*) sadrži listu svih povezanih projekata s aplikacijom. Svaki kreirani projekt ima svoj vlastiti direktorij u kojemu se nalaze informacije o projektu. ROPD zapravo sprema putanje do direktorija određenog projekta i time

omogućava da dodani i kreirani projekti budu dostupni u aplikaciji i nakon što se aplikacija ponovno pokrene.

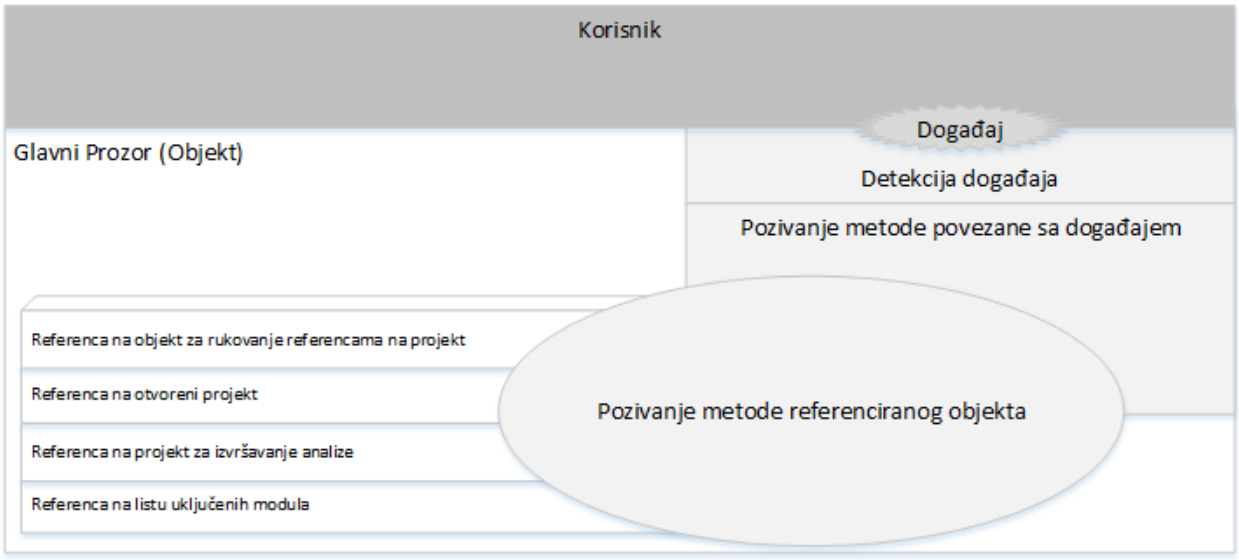
Referenca na otvoreni projekt u nastavku teksta ROP (engl. *reference object project*) je referenca na apstrakciju projekta. Između izvan-mrežnog i mrežnog projekata postoje razlike koje su zamaskirane iza ove apstrakcije. Ako korisnik otvori neki projekt s popisa projekata najprije se dohvaća putanja iz ORPD-a, deserijalizira se objekt koji je serijaliziran u datoteci na definiranoj putanji i njegova se referenca dodjeljuje atributu ROP u glavnom prozoru.

Referenca na objekt za izvršavanje analize u nastavku teksta ROIA (engl. *reference object inputs analyzer*) je referenca na objekt koji izvršava analizu ulaza određenog projekta. Objekt za izvršavanje analize povezuje se na sučelje aplikacije s čime je omogućena kontrola izvršavanja preko glavnog prozora. S druge strane, glavni prozor se opet povezuje na objekt za izvršavanje analize kako bi mu objekt za izvršavanje analize mogao slati rezultate, trenutno analizirani okvir za prikaz i ostale informacije koje glavni prozor prikazuje korisniku tijekom analize određenog ulaza.

Referenca na listu uključenih modula, u nastavku teksta RML (engl. *modules list*), je lista putanja do algoritamskih biblioteka koji su povezani s aplikacijom. Nakon što se dodaju/obrišu algoritamske biblioteke iz direktorija s algoritamskim bibliotekama te se osvježi popis dostupnih algoritamskih biblioteka sve biblioteke pokušavaju se otvoriti koristeći API za algoritamske biblioteke. Ako otvaranje uspješno prođe, putanja do uspješno otvorene biblioteke sprema se na ovu listu. Kako se biblioteke točno otvaraju koristeći API za algoritamske biblioteke opisano je u točki 4.3.8.

4.2.4. Pozivanje funkcionalnosti aplikacije i sučelja aplikacije

Kao što je bilo predstavljeno u potpoglavlju 4.2.4 funkcionalnosti aplikacije izvode se u četiri referencirana objekta u glavnom prozoru. Povezanost između korisničkog sučelja aplikacije i same funkcionalnosti aplikacije prikazana je na slici 4.6. Nakon što korisnik aktivira neki događaj na sučelju, najprije se poziva metoda glavnog prozora koja je povezana sa sučeljem aplikacije. Unutar metode sučelja, ovisno o stanju aplikacije, pozivaju se metode referenciranih objekata. Dalje se izvode metode referenciranih objekata.



Sl. 4.6. Povezanost sučelja aplikacije i funkcionalnosti aplikacije.

4.2.5. Datotečni sustav

Zbog mogućnosti spremanja rezultata na disk potreban datotečni sustav organiziran na odgovarajući način. Na slici 4.7. prikazana je struktura datotečnog sustava PQM aplikacije.

Sl. 4.7. Datotečni sustav PQM aplikacije.

Datotečni sustav PQM aplikacije sastoji se od direktorija PQM u kojem se nalazi izvršna datoteka PQM aplikacije. Unutar PQM direktorija nalazi se i direktorij sa bibliotekama na koje se aplikacija oslanja. To su direktorij FFMPEG u kojoj se nalaze biblioteke korištene za dekodiranje videa, zatim direktorij Grabber u kojoj se nalaze biblioteke za rad sa *grabber* uređajem i direktorij Moduli u kojoj se nalaze algoritamske biblioteke za analiziranje videa. Unutar PQM direktorija nalazi se i datoteka *PQMProgramData.xml* u kojoj su spremljene poveznice (putanje do projekata). Pojedini projekt nalazi se na proizvoljnoj putanji i čini ga direktorij Projekt u kojoj se nalazi datoteka *project.xml*. U *project.xml* datoteci serijaliziran je

objekt projekt koji je detaljnije opisan u točki 4.3.7. U direktoriju projekt nalazi se i proizvoljan broj direktorija sa rezultatima analize. U direktoriju rezultat nalazi se datoteka *result.xml* u kojoj je serijaliziran objekt rezultat koji je opisan u točki 4.3.6. i datoteka *data.txt* u kojoj su zapisani rezultati analize za svaki algoritam. U *data.txt* zapisani su rezultati u strukturiranom formatu gdje prvih 20 znakova definira redni broj analiziranog okvira, idućih 8 znakova sadrži rezultat 1. algoritma korištenog u analizi, sljedećih 8 znakova 2. itd. Koji je algoritam povezan s kojim rezultatom može se pronaći u datoteci *result.xml* gdje je zapisan popis korištenih algoritama onim slijedom kojim su se zapisivali rezultati u *data.txt*.

4.3. Programska implementacija PQM aplikacije

U prethodnom potpoglavlju opisana je dizajnerska ideja aplikacije, odnosno logička i vizualna skica cijele aplikacije na kojoj se zasniva programska implementacija. Osim navedenih dizajnerskih ideja u nastavku će biti dodane i objašnjene nove jer ih je lakše objasniti izravno kroz implementacijski korak prilikom izrade PQM aplikacije. U ovom potpoglavlju najprije će biti riječi od kojih se sve modula sastoji aplikacija. U svakom potpoglavlju biti će opisan jedan modul, biti će objašnjeno koja je svrha modula. Nakon što bude objašnjen svaki dio zasebno biti će riječi o tome kako su svi dijelovi povezani u jednu cjelinu.

4.3.1. Microsoft Visual Studio 2013 i FFMPEG

Za izradu PQM aplikacije korišteno je Microsoft Visual Studio 2013 razvojno okruženje i FFMPEG biblioteka.

Visual Studio 2013 je programsko razvojno okruženje kojeg je razvio Microsoft za potrebe razvoja različitih programskih rješenja. Visual Studio podržava razvoj Basic, C#, SQL Server, Visual F#, JavaScript, C++ i drugih aplikacija. Za potrebe rada korišten je C++/CLI programski jezik. CLI (engl. *common language infrastructure*) kratica je za sučelje prema .NET programskim jezicima. Ono omogućava kombiniranje više programskih jezika u jednoj aplikaciji. C++/CLI programski jezik omogućava korištenje svih funkcionalnosti iz .NET *framework-a*, mogućnost implementacije dijelova programiranih u C i C++ programskim jezicima. Jednostavnost izrade sučelja i mogućnost izrade modula visokih performansi u C/C++ jeziku razlozi su zašto je ovaj programski jezik odabran za pisanje PQM aplikacije. Važno je napomenuti kako je korištena verzija 4.5 .NET *frameworka.*

FFMPEG je biblioteka sa brojnim funkcionalnostima za rad sa zvukom i videom. Sadrži funkcije za kodiranje, dekodiranje videa, skaliranje i mnoge druge funkcije za obradu zvuka i

videa. U programu je korištena za dekodiranje video datoteka. Biblioteka je otvorenog koda i moguće ju je koristiti u osobne, ali i komercijalne svrhe. Verzija FFMPEG-a korištena u ovom radu je 20160308-git-5061579.

U nastavku slijedi opis svakog zasebnog modula. Budući da je korišten objektno orijentirani pristup jedan modul čini jedna ili više klasa.

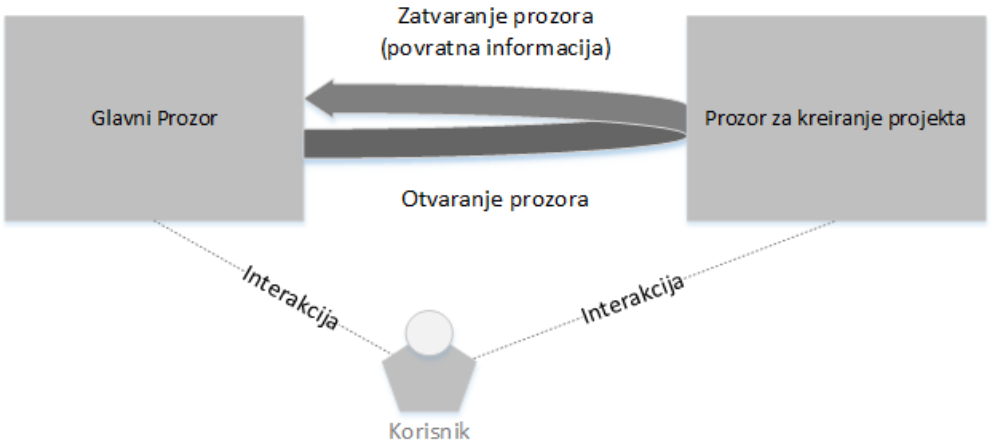
4.3.2. Klase za rad s prozorima

Klasa glavni prozor, u nastavku teksta CFM (engl. *class form main*), je klasa pomoću koje se instancira objekt OFM (engl. *object form main*). OFM je glavni dio aplikacije koji omogućava korisniku da preko sučelja upravlja funkcionalnostima aplikacije. CFM definira prozor. Klasa je napravljena koristeći .NET Windows Formu. Glavnina sučelja napravljena je koristeći Visual Studio alat za dizajniranje Windows Formi. Ovaj alat omogućava dodavanje elemenata sučelja jednostavnim prevlačenjem elemenata iz skupa dostupnih elemenata i pozicioniranjem elementa na Windows Formi. Većina atributa OFM objekta namijenjena je izgledu aplikacije. S njima su definirani elementi sučelja, pozicije elemenata, boja elemenata itd. OFM sadrži i metode koje su izravno povezane sa sučeljem aplikacije koje se aktiviraju na akciju korisnika. Na korisnikovu akciju aktivira se odgovarajuća metoda povezana sa sučeljem, metoda poziva neke druge metode OFM-a za podešavanje sučelja ili pozivanje metoda drugih objekata referenciranih kao atributi u OFM-u. Metode referenciranih objekata izvršavaju glavne funkcionalnosti aplikacije, kao što su rukovanje s poveznicama na projekte, otvaranje i uređivanje resursa otvorenog projekta, rukovanje analizom video sadržaja i rukovanje s uključenim modulima. Što je već spomenuto u točki 4.2.4.

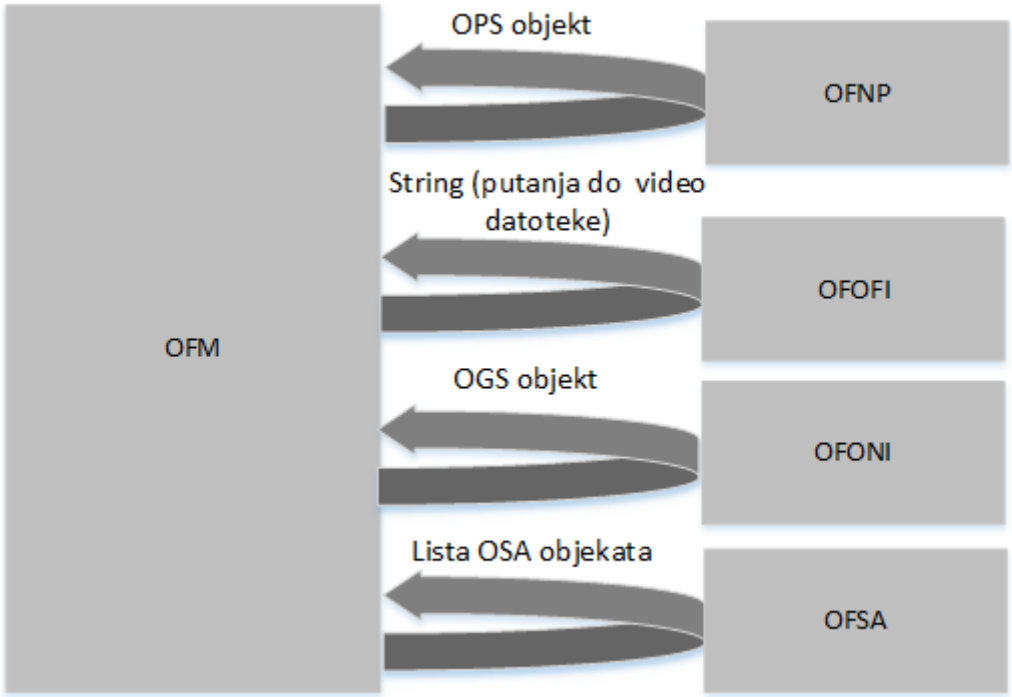
Klasa sa osnovnim informacijama o projektu, u nastavku teksta CPS (engl. *class project structure*), je klasa pomoću koje se instancira objekt koji sadrži osnovne informacije o projektu (engl. *object project structure* - OPS). OPS u atributima sadrži osnove informacije o projektu. To su: tip projekta, ime projekta i direktorij gdje se projekt nalazi. Svrha OPS objekta biti će opisana u nastavku točke i u točki 4.3.7.

Klasa prozora za kreiranje projekta, u nastavku teksta CFNP (engl. *class form new project*), je klasa pomoću koje se instancira objekt (prozor) naziva OFNP (engl. *object form new project*). Ovaj objekt omogućava korisniku kreiranje novog projekta. Nakon što korisnik u OFM-u odabere opciju za kreiranje projekta otvara se OFNP. OFNP je prozor preko kojeg korisnik postavlja direktorij u kojem želi kreirati projekt, postavlja naziv i tip projekta. OFNP nakon

svakog unosa ispituje da li je unos ispravan, npr. nakon što korisnik unese putanju i naziv projekta OFNP ispita da li je u odabranom direktoriju moguće kreirati direktorij koji odgovara nazivu projekta. U slučaju ako već postoji istoimeni direktorij vraća korisniku informaciju da se projekt sa zadanim direktorijem i nazivom ne može kreirati. OFNP je zapravo interaktivni obrazac koji korisniku vraća informaciju je li je unos pravilan ili ne. Prije zatvaranja OFNP sprema sve podatke koje je korisnik unio u jedan atribut koji je referenca, zapravo OPS objekt. Nakon što se zatvori OFNP OFM može pristupiti atributu koji sadrži OPS objekt i na taj način dohvaća podatke koje je korisnik unio.



Sl. 4.8. Povezanost glavnog prozora i prozora za kreiranje projekta.



Sl. 4.9. Povezanost i komunikacija glavnog i sporednih prozora.

Na slici 4.8. i 4.9. prikazana je komunikacija između korisnika, OFM-a i OFNP-a. Ostali dijelovi sa slike biti će objašnjeni u nastavku.

Klasa prozora za dodavanje izvan-mrežnog ulaza u nastavku teksta CFOFI (engl. *class form offline input*) je klasa pomoću koje se instancira objekt (prozor) naziva OFOFI (engl. *object form offline input*). Princip komunikacije OFM-a i OFOFI-a je isti kao komunikacija OFM-a i OFNP-a samo je povratna informacija OFOFI-a drugačija. Dakle, nakon što korisnik odabere opciju za dodavanje izvan-mrežnog ulaza, instancira se i otvara prozor OFOFI. OFOFI je prozor (obrazac) u kojem korisnik unosi podatke za izvan-mrežni ulaz. Izvan-mrežni ulaz je video kontejner na lokalnom računalu pa je jedini podatak o izvan-mrežnom ulazu putanja do video kontejnera. Osim oblika povratne informacije sve se ostalo analogijom može primijeniti kao kod CFNP klase.

Klasa postavke *grabber*-a, u nastavku teksta CGS (engl. *class grabber settings*), je klasa pomoću koje se instancira objekt naziva OGS (engl. *object grabber settings*). Atributi OGS objekta su: IP adresa *grabber-a*, ulaz kojeg *grabber* koristi za dobivanje video signala i port preko kojeg aplikacija komunicira sa *grabber-om*. OGS objekt sadrži samo informacije za povezivanje sa *grabber-om* i nema nikakvih metoda.

Klasa prozora za dodavanje/podešavanje mrežnog ulaza, u nastavku teksta CFONI (engl. *class form online input*), je klasa pomoću koje se instancira objekt (prozor) naziva OFONI (engl. *object form online input*). Ovaj objekt omogućava korisniku podešavanje postavki *grabber-a*. Nakon što korisnik otvori prozor u OFONI-u omogućeno je podešavanje postavki *grabber-a* tako što korisnik u zadanu formu unosi IP adresu *grabber-a*, tip ulaza i port *grabber-a*. Budući da OGS i forma sadrže iste vrijednosti, može se zaključiti da se podaci iz forme zapisuju u OGS objekt isto kao što je bio slučaj za OFNP. Komunikacija između OFM-a i OFONI-a je ista kao što je bila između OFM-a i OFNP-a i OFM-a i OFOFI-a. Dakle OFONI je sučelje/obrazac preko kojeg korisnik podešava postavke *grabber-a*. Nakon što korisnik napravi izmjene postavki i zatvori prozor, OFM iščitava podatke iz OFONI-a tako što dohvaća atribut referencu na OGS objekt. Važno je još napomenuti kako prozor OFONI ima opciju prikaza trenutnog videa sa *grabber-a*. Ovime je omogućeno da korisnik, nakon podešavanja postavki, provjeri postavke *grabber-a*. Ako je sve ispravno uneseno video sadržaj dohvaćen sa *grabber* uređaja biti će prikazan na ekranu.

Klasa odabrani algoritam, u nastavku teksta CSA (engl. *class selected algorithm*), je klasa pomoću koje se instancira objekt naziva OSA (engl. *object selected algorithm*). OSA za atribut

ima naziv algoritamske biblioteke (vidi potpoglavlje 4.1.4.) koju korisnik želi koristiti za analizu videa te ostale atribute koji daju informacije o korištenom algoritmu..

Klasa prozor za odabir algoritama, u nastavku teksta CFSA (engl. *class form selected algorithms*), je klasa pomoću koje se instancira objekt (prozor) naziva OFSA (engl. *object form selected algorithms*). Cilj prozora je omogućiti korisniku odabir algoritamskih biblioteka s kojima želi provesti analizu videa. U prozoru je prikazan popis dostupnih algoritamskih biblioteka s opcijom *checkbox* pomoću koje korisnik odabire one biblioteke s kojima želi provesti analizu odabranih ulaza. Popis se osvježava svaki puta kada se OFSA otvori. Osvježavanje popisa izvodi se tako što se u konstruktoru objekta pozivaju metode koje pretražuju sve datoteke u direktoriju s algoritamskim bibliotekama. Kako se ne bi učitale pogrešne biblioteke koje eventualno mogu dospjeti u direktorij namijenjen algoritamskim bibliotekama, metode sve datoteke provjeravaju preko odgovarajućeg API-a. Ako sve uspješno prođe biblioteka se dodaje na popis algoritama za odabir.

4.3.3. Klasa programska okolina

Klasa programska okolina, u nastavku teksta CPE (engl. *class program enviroment*), je klasa pomoću koje se instancira objekt naziva OPE (engl. *object program enviroment*). Objekt čine atributi kojima je definirana okolina programa.

Pod okolinu se programa podrazumijeva:

- Putanja do direktorija u kojoj se PQM aplikacija nalazi,
- Putanja do direktorija u kojoj se nalaze algoritamske biblioteke,
- Putanja do datoteke u kojoj je serijaliziran objekt programski podaci o kojem će biti riječi u točki 4.3.4.,
- Naziv datoteke u kojoj se spremaju informacije o projektu (serijalizirani objekt projekt), te
- Naziv datoteke u kojoj se spremaju informacije o rezultatima (serijalizirani objekt rezultat).

Osim atributa OPE objekt sadrži i statičke metode s kojima se dohvaćaju varijable koje pripadaju programskoj okolini. Atributi ove klase su statičke, postavljaju se prilikom konstrukcije objekta i ne mijenjaju se tijekom izvođenja programa.

4.3.4. Klasa programski podaci

Klasa programi podaci, u nastavku CPD (engl. *class program data*), je klasa pomoću koje se instancira objekt naziva OPD (engl. *object program data*). Svrha OPD objekta je čuvanje poveznica na projekte. OPD je zapravo objekt koji sadrži listu OPS objekata. Budući da se u OPS objektu nalazi putanja do direktorija s projektom potrebno je dodati OPS objekt projekta na listu OPS objekata u OPD objektu. Dakle, prilikom svakog kreiranja i dodavanja projekta poveznica na projekt u obliku OPS-a objekta dodaje se u OPD objekt. Pošto se OPD objekt na kraju programa serijalizira, a na početku programa deserijalizira, omogućeno je da svi kreirani i dodani projekti budu dostupni i nakon ponovnog pokretanja aplikacije.

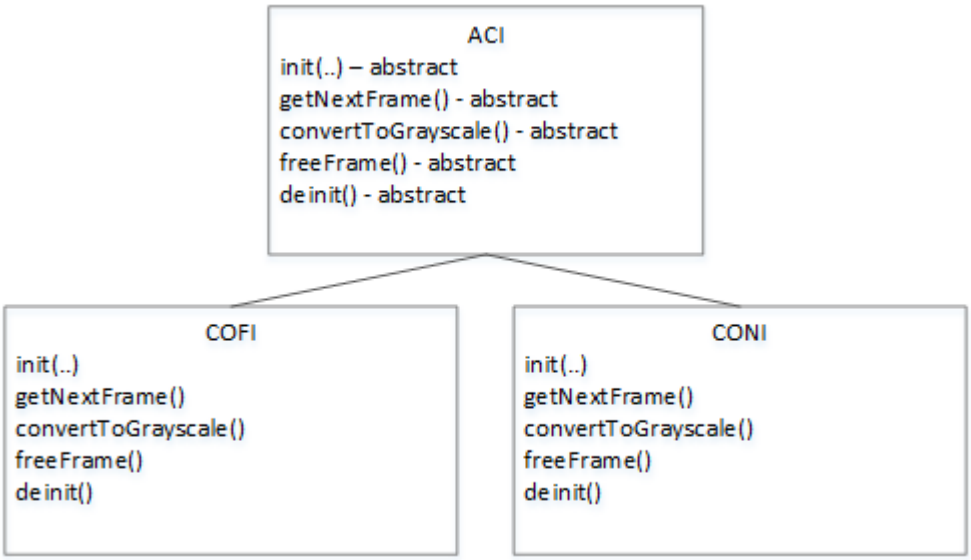
4.3.5. Klase za rad s ulazima

Apstraktna klasa ulaz, u daljnjem tekstu ACI (engl. *abstract class input*), je apstrakcija koja generalizira moguće ulaze svakog projekta (vidi potpoglavlje 4.1.3.). Atributi klase su: jedinstveni broj ulaza, referenca na nit u kojoj se procesira ulaz, referenca na semafor koji govori koliko je slobodnih mjesta u među-spremniku, referenca na semafor koji daje broj okvira u među-spremniku i *mutex* koji sprječava istovremeni pristup među-spremniku. ACI sadrži apstraktne metode za inicijalizaciju ulaza, de-inicijalizaciju ulaza, dohvaćanje okvira, oslobađanje dohvaćenog okvira, dohvaćanje visine okvira, širine okvira te metodu koja se treba izvoditi u posebnoj niti u kojoj će se dekodirati ili dohvaćati okviri, apstraktne metode za konvertiranje okvira u *bitmap* i *grayscale* format.

Klasa izvan-mrežnog ulaza, u nastavku teksta COFI (engl. *class offline input*), nasljeđuje ACI, prepisuje apstraktne metode definirane u ACI i dodaje svoje atribute i metode. Pomoću klase COFI instancira se objekt naziva OOFI (engl. *object offline input*) pomoću kojega se dekodira video kontejner koji se nalazi na lokalnom računalu na putanji definiranoj u atributu OOFI-a. OOFI dekodira okvire, sprema ih u međuspremnik, čini ih dostupnim ostalim dijelovima koda tako što se pozivaju metode OOFI-a za dohvaćanje okvira i konverziju u ovisnosti o potrebi.

Klasa mrežni ulaz, u nastavku teksta CONI (engl. *class online input*), nasljeđuje ACI, prepisuje apstraktne metode definirane u ACI-u. Pomoću klase CONI instancira se objekt naziva OONI (engl. *object online input*). OONI kao atribut ima referencu na OGS. Koristeći postavke definirane u objektu OGS instancira se objekt *grabber* koji se može promatrati kao vritualni uređaj koji omogućava dohvaćanje video okvira pozivanjem funkcije za dohvaćanje okvira. Okvir se sprema u među-spremnik i dostupan je drugim dijelovima koda isto kao i okvir OOFI

objekta. Za konvertiranje i dohvaćanje okvira iz drugih dijelova koda vrijedi analogija sa konvertiranjem i dohvaćanjem okvira iz OOFI objekta.



Sl. 4.10. Polimorfizam ulaza.

Na slici 4.10. prikazano je kako COFI i CONI klase nasljeđuju ACI klasu. Ovakvim pristupom omogućen je polimorfizam nad objektima ulaza.

4.3.6. Klasa rezultat

Klasa rezultat, u nastavku teksta CR (engl. *class result*), je klasa pomoću koje se kreira instanca objekta naziva OR (engl. *object result*). Atributi ove klase su putanja do datoteke u kojoj se zapisuju rezultati analize kako je opisano u potpoglavlju 4.2.5, lista algoritama korištenih za analizu videa, vrijeme početka analize i vrijeme kraja analize.

4.3.7. Klase za rad s projektima

Kao što je već rečeno u potpoglavlju 4.1. postoje dva tipa projekta izvan-mrežni projekt i mrežni projekt. To su dva projekta koji imaju svoje sličnosti i razlike. Kako bi se ta dva projekta generalizira primjenom polimorfizma napravljena je apstraktna klasa koju će izvan-mrežni i mrežni projekti naslijediti. Ovakvo generaliziranje potrebno je napraviti kako bi se u OFM mogle implementirati opcije otvaranja projekta, dodavanja ulaza itd. Kako je točno izvedeno povezivanje OFM-a i projekata opisano je u potpoglavlju 4.3.10.

Apstraktna klasa projekt, u nastavku teksta ACP (engl. *abstract class project*), je čista apstraktna klasa koju nasljeđuju projekti.

Atributi klase su:

- Referenca na OPS,
- Jedinstveni identifikacijski broj zadnjeg ulaza,
- Referenca na listu ulaza,
- Jedinstveni identifikacijski broj zadnjeg rezultata, te
- Referenca na listu rezultata.

Čiste virtualne metode su:

- Metoda za dodavanje ulaza,
- Metoda za dodavanje rezultata,
- Metoda za računanje veličine projekta na disku,
- Metoda za spremanje projekta,
- Metoda za dohvaćanje popisa ulaza,
- Metoda za dohvaćanje popisa rezultata,
- Metoda za dohvaćanje ulaza preko jedinstvenog identifikacijskog broja, te
- Statička metoda za deserijalizaciju projekta sa diska.

Klasa izvan-mrežni projekt, u nastavku teksta COFP (engl. *class offline project*), i klasa mrežni projekt u nastavku teksta CONP nasljeđuju ACP i prepisuju čiste virtualne metode. Pomoću COFP-a instancira se objekt naziva OOFP (engl. *object offline project*), a pomoću CONP-a instanicra se objekt naziva CONP. Ovakvim nasljeđivanjem iste apstraktne klase omogućeno je da se u OFM-u referencom na objekt koji je naslijedio ACP primjeni svojstvo polimorfizma što je bitno prilikom kreiranja sučelja aplikacije.

4.3.8. Klase za rad s dinamičkim bibliotekama

Klasa modul, u nastavku teksta CM (engl. *class module*), je klasa pomoću koje se instancira objekt naziva OM (engl. *object module).* OM je omot (engl. *wrapper*) oko .dll biblioteke koji definira jedinstveni API za različite tipove algoritama. Dakle, ovaj objekt rukuje sa modulom na najnižoj razini dll-a, on otvara biblioteku pregledava da li je biblioteka napisana prema definiciji API-a i ako je sve u redu omogućava kreiranje viših slojeva apstrakcije. Za atribute OM ima pokazivač na biblioteku i objekt definiran API-em. Prva sljedeća viša apstrakcija opisana je u nastavku.

Klasa instanca modula, u nastavku teksta CMI (engl. *class module instance*), je klasa koja nasljeđuje CM i pomoću nje se instancira objekt naziva OMI (engl. *object module instance*). U

OM-u je napravljena apstrakcija za rukovanje bibliotekom na najnižoj razini, a u OMI-u je napravljena apstrakcija druge razine. U OM-u su dohvaćene funkcije biblioteke, a u OMI-u je napravljena instanca objekta koristeći API.

Klasa algoritam, u nastavku teksta CA (engl. *class algorithm*), je klasa koja nasljeđuje CMI i pomoću nje se instancira objekt naziva OA (engl. *object algorithm*). Klasa algoritam treća je razina apstrakcije. Ovom apstrakcijom radi se sloj na OMI-u. OM ne mora uvijek biti algoritamska biblioteka. OM može podržavati različite funkcionalnosti. Da li je OM modul za detekciju artefakata može se utvrditi tek nakon što se pregledaju vrijednosti pojedinih varijabli modula. Objekt OA prilikom svoje konstrukcije pretražuje instancu modula tražeći varijable koje definiraju API algoritama.

API za algoritam čini:

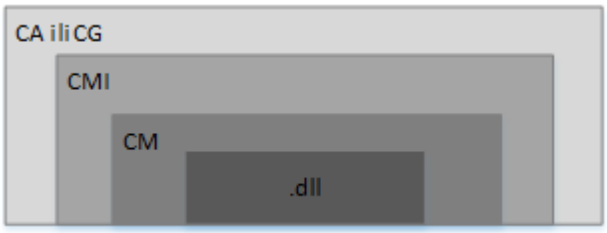
- Varijabla kojom je definiran format okvira kojeg algoritam može obraditi,
- Varijabla preko koje algoritam prima širinu okvira,
- Varijabla preko koje algoritam prima visinu okvira,
- Varijabla preko koje algoritam prima redni broj okvira,
- Varijabla preko koje algoritam prima adresu memorije na kojoj se nalazi video okvir,
- Varijabla za pokretanje analize, te
- Varijabla preko koje algoritam vraća vrijednost analize.

Ako konstrukcija OA uspješno prođe znači da je sve ispravno i da se OA može koristiti za analizu okvira. Za ispravno izvršavanje analize potrebno je još samo da dio koda koji poziva OA metode za analizu poštuje redoslijed izvršavanja.

Redoslijed izvršavanja gledajući sa strane dijela koda koji poziva OA metode je sljedeći:

1. Utvrditi format okvira kojeg OA može obraditi
2. Konvertirati okvir u odgovarajući format
3. Postaviti širinu okvira na OM
4. Postaviti visinu okvira na OM
5. Postaviti redni broj okvira na OM
6. Postaviti adresu pokazivača okvira na OM
7. Postaviti varijablu za pokretanje analize
8. Pročitati rezultat

Klasa *grabber*, u nastavku teksta CG (engl. *class grabber*), nasljeđuje CMI i pomoću nje se instancira objekt naziva OG (engl. *object grabber*). Prve dvije apstrakcije su iste i za OG i za OA samo što prilikom treće apstrakcije konstrukcije OG objekta pretražuje se OMI i traže se varijable koje čine API za *grabber*. Ako je OG uspješno konstruiran, moguće je pozivati funkcije s kojima se postavljaju postavke *grabber-a* (IP adresa, video ulaz, port) i moguće je pokrenuti i zaustaviti dohvaćanje video okvira pomoću *grabber* uređaja.



Sl. 4.11. Slojevi apstrakcije .dll biblioteke.

Na slici 4.11. prikazani su slojevi apstrakcije oko .dll biblioteke.

4.3.9. Klasa analiza ulaza

Klasa analiza ulaza, u nastavku teksta CIA (engl. *class inputs analyzer*), je klasa pomoću koje se instancira objekt naziva OIA (engl. *object inputs analyzer*). OIA objekt omogućava analizu odabranih ulaza s odabranim algoritamskim bibliotekama. Atributi ovog objekta su referenca na objekt klase ACP, lista jedinstvenih identifikatorskih brojeva kojima su definirani ulazi u referenciranom projektu koje je korisnik odabrao za analizu, zatim lista OA objekata, zastavice za kontrolu tijeka analize (zastavica za pokretanje i zaustavljanje analize), događaji na koje se mogu pretplatiti ostali dijelovi programa, događaj za ispis poruke (ispis rednog broja trenutno analiziranog okvira i ispis rezultata) te događaj za slanje okvira u *bitmap* formatu.

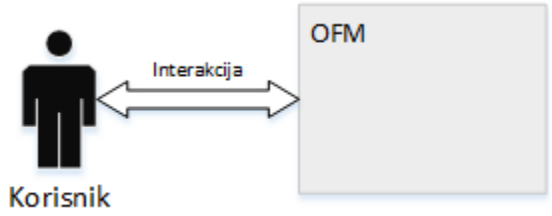
Konstruktor ove klase kao parametre prima referencu na projekt nad kojim je potrebno provesti analizu, zatim listu ulaza koje je potrebno analizirati i listu OSA objekata. Prilikom konstruiranja objekta, referenca na projekt i referenca na listu ulaza pridjeljuje se atributima OIA, a za svaki objekt iz liste OSA objekata konstruira se OA i konstruirani objekt dodaje se na listu OA objekata.

Nakon što je OIA objekt uspješno konstruiran spreman je za analizu. Dio koda koji kreira OIA objekt može se još pretplatiti na događaje objekta kako bi dobivao povratne poruke. Nakon pokretanja analize pokreće se metoda OIA objekta u zasebnoj petlji. U toj metodi postoje dvije petlje. Prva petlja ide po odabranim ulazima, a druga se petlja izvršava sve dok se ne analizira

cijeli ulaz. Ovo vrijedi za izvan-mrežni projekt dok kod mrežnog projekta postoji samo druga petlja jer takav projekt ima samo jedan ulaz. U drugoj petlji dohvaćaju se okviri sa *grabber-a* sve dok korisnik ne zaustavi analizu. Rezultati analize za svaki okvir se zapisuju u tekstualnu datoteku koja se nalazi na putanji definiranoj u atributu OR objekta. Nakon što je analiza gotova OR objekt se dodaje na listu rezultata referenciranog projekta.

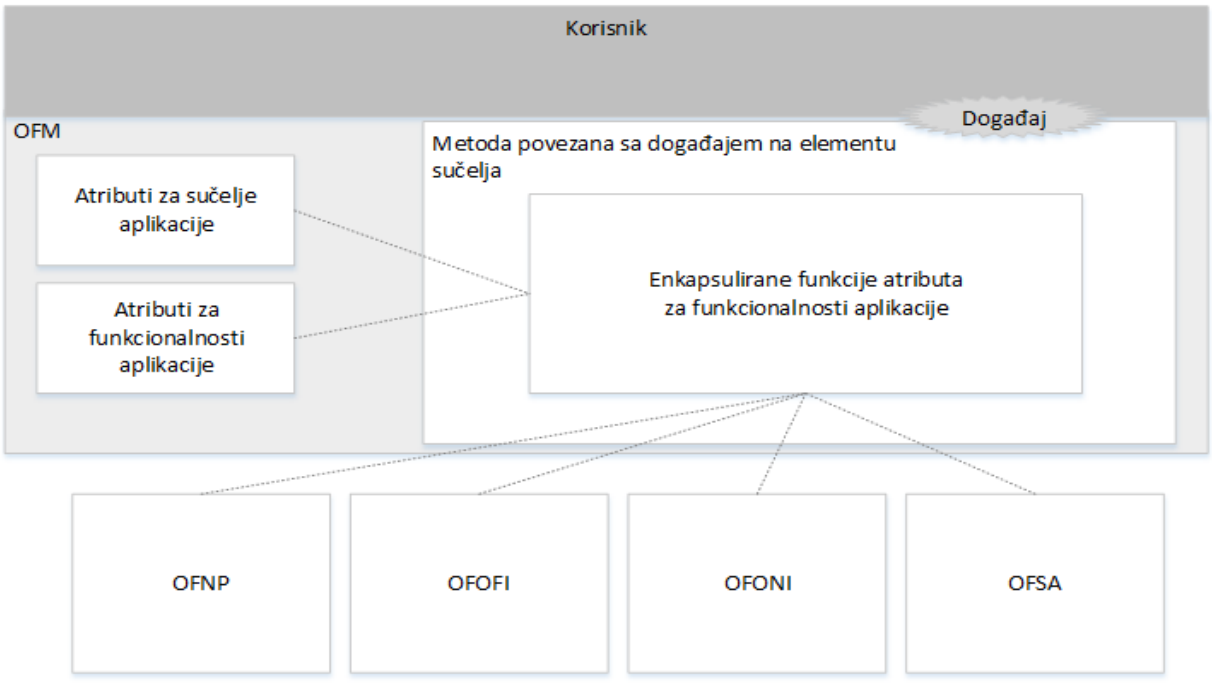
4.3.10. Povezanost svih instanciranih objekata

U ovom odjeljku je opisana povezanost svih klasa, tj. svih objekata koji se instanciraju tijekom rada aplikacije i na koji način oni komuniciraju



Sl. 4.12. Interakcija korisnika i OFM-a (osnovni prikaz).

Nakon pokretanja aplikacije korisniku je prikazan glavni prozor OFM (Sl. 4.12.). Koristeći opcije u glavnom prozoru korisnik može izvoditi slučajeve definirane u potpoglavlju 4.2.1. Nakon što korisnik odabere neku od opcija poziva se OFM metoda koja izmjenjuje sučelje aplikacije u ovisnosti o stanju u kojem se aplikacija nalazi.



Sl. 4.13. Interakcija korisnika i OFM-a (detaljniji prikaz).

Primjerice prilikom pokretanja aplikacije prikazan je početni prikaz, panel PH, a ako korisnik odabere opciju za uređivanje modula PH se skriva i prikazuje se PA. Ovo je samo jedan primjer kako se izmjenjuju paneli na OFM-u. Na slici 4.13. prikazan je tijek od stvaranja događaja od strane korisnika do pokretanja pozadinskog dijela aplikacije. Dakle, korisnik kreira neki događaj na sučelju, pritisne ili odabere određenu opciju, a nakon toga se poziva metoda povezana s tim elementom sučelja i događajem. Nakon što se metoda pozove izvodi se blok koda metode povezane sa sučeljem i događajem. Unutar tog koda nalaze se funkcije koje pokreću rad u pozadini aplikacije. Na primjer nakon što korisnik odabere opciju za kreiranje projekta, poziva se metoda koja rukuje događajem kreiranja projekta, unutar metode kreira se objekt OFNP i otvara se prozor. Nakon otvaranja prozora definiranog OFNP-om blokira se OFM. Nakon što je korisnik obavio što je želio, zatvara prozor i nastavlja se izvođenje metode koja ga je otvorila. U toj metodi provjerava se da li je korisnik unio podatke kao što je to opisano u točki 4.3.2. prozor OFNP, ako je, dohvaća se OPS objekt pomoću kojeg se stvara novi projekt u OFM-u. Nakon što se kreira novi projekt na listu povezanih projekata OPD objekta dodaje se OPS objekt.

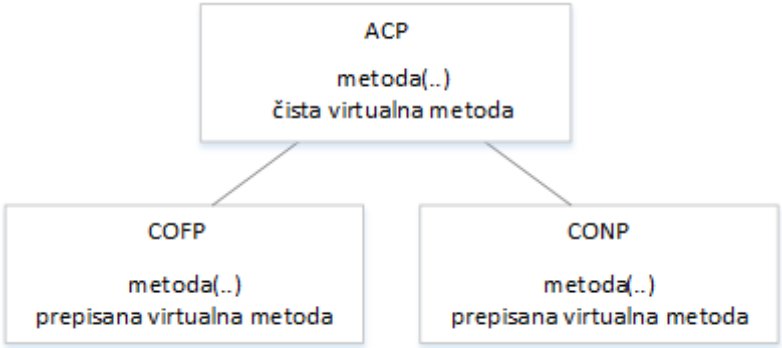
Ovaj princip rada koristi se i prilikom dodavanja izvan-mrežnog ulaza, mrežnog ulaza i označavanja algoritama za analizu. Otvara se prozor OFNP, OFOFI, OFONI ili OFSA. Korisnik popunjava formu i nakon što je napravio sve što je želio pritišće gumb s kojim potvrđuje izmjene. Nakon potvrđivanja objekt (prozor) provjerava što je korisnik unio u formu, kreira strukturu podataka ovisno o unesenim podacima i zatvara se.

Četiri su važna atributa OFM-a (vidi točku 4.2.3) Nakon što se konstruira objekt OFM samo je ROPD definiran, svi ostali atributi su nedefinirani. ROPD pokazuje na objekt OPD. OPD se samo prilikom prvog pokretanja programa konstruira koristeći konstruktor. Prilikom svih sljedećih pokretanja programa OPD se deserijalizira sa diska. (vidi točku 4.3.4.)

Nakon što je deserijaliziran OPD i konstruiran OFM, korisnik vidi sučelje aplikacije (panel PH) na kojemu se nalazi lista svih povezanih projekata. Dodavanje projekta moguće je napraviti odabirom opcije za dodavanje projekta. Nakon što je projekt kreiran i dodana je poveznica u OPD, on se nalazi na popisu na panelu PH. Sada korisnik može odabrati opciju za otvaranje novog projekta. Nakon što je opcija pritisnuta izvodi se slijed prema slici 4.13. Unutar metode povezane s elementom i događajem poziva se funkcija s kojom se dohvaća redni broj označenog projekta na listi projekata. Redni broj projekta na listi projekata odgovara indeksu OPS objekta u listi projekata u OPD projektu. Nakon što je odgonetnut redni broj projekta na listi projekata dohvaća se OPS objekt, putanja do projekta definirana u ovom objektu koristi se za

deserijalizaciju objekta (projekta) iz datoteke na definiranoj putanji. Ako je deserijalizacija uspješna ROP referenca unutar OFM objekta postavlja se na deserijaizirani objekt (projekt). Ovim postupkom projekt je otvoren potrebno je još samo podesiti sučelje koje odgovara otvorenom projektu. Nakon što je sučelje podešeno prikazan je panel PP.

Referenca ROP prikazuje na objekt čiste apstraktne klase ACP, dakle referenca ne zna o kojem se objektu, tj. projektu radi samo zna da oba projekta nasljeđuju ACP klasu i da imaju iste metode kao ACP klasa (Sl. 4.14.). Primjenom ovog svojstva, svojstva polimorfizma pozivamo uvijek istu metodu s istim parametrima, a kod koji se izvodi je potpuno drugačiji.



Sl. 4.14. Svojstvo polimorfizma na projektu.

Kada je u aplikaciji instanciran OFM u kojem je referenciran ROPD i ROP preostale su nedefinirane reference ROIA i RML. RML referenca je nedefinirana sve dok aplikacija ne zatreba listu algoritamskih biblioteka. Ako korisnik odabere opciju analize ulaza ili opciju za uređivanje modula aplikacija mu mora pokazati sve dostupne module. Kako se provjerava dostupnost modula i kreira lista modula opisano je u odjeljku 4.2.3.

Nakon što je korisnik dodao algoritamske biblioteke u direktorij s bibliotekama i neke ulaze u projektu moguće je analiziranje ulaza. Analiziranje ulaza, gledajući s korisničke strane sastoji se od tri koraka. Odabir ulaza za analizu, odabir algoritamskih biblioteka s kojima će se analizirati ulazi i pokretanje analize. Korisnik najprije odabire ulaze koje želi analizirati tako što na popisu ulaza označi ulaze, nakon označavanja ulaza odabire opciju za analizu označenog. Nakon što je opcija pritisnuta pojavljuje se prozor OFSA za odabir modula s kojima će se analiza provesti, korisnik označi one module koje želi koristiti tijekom analize i potvrđuje odabrano. Nakon što korisnik potvrdi odabrano OFSA kreira listu OSA objekata i čini ju dostupnom OFM-u. OFM dohvaća listu s OSA objektima i kreira listu u kojoj se nalaze jedinstveni identifikatori ulaza. Lista s jedinstvenim identifikatorima ulaza kreira se tako da se za svaki označeni ulaz gleda redni broj ulaza na popisu ulaza. Redni broj svakog ulaza se onda koristi za dohvaćanje

objekta ulaz iz liste ulaza u otvorenom projektu gdje redni broj odgovara indeksu elementa liste. Nakon što je lista s jedinstvenim identifikatorima ulaza kreirana i lista sa OSA objektima dostupna, poziva se konstruktor za kreiranje OIA objekta. Konstruktor OIA objekta kao parametre prima referencu na projekt čiji se ulazi analiziraju, listu sa OSA objektima i listu sa jedinstvenim identifikatorima ulaza. U konstruktoru se zatim za svaki OSA objekt konstruira OA objekt prema potpoglavlju 4.3.8. gdje je opisano konstruiranje OA objekta i konstruirani objekt se doda na listu algoritama spremnih za analizu. Nakon što je OIA objekt konstruiran OFM se pretplaćuje na događaje OIA objekta i OIA objekt se pretplaćuje na događaje OFM objekta. Pretplaćivanjem OFM-a na OIA omogućeno je da nakon pokretanje analize OIA može pozivati OFM metode koje će korisniku prikazivati rezultate izvođenja i koje će signalizirati OFM-u stanje analize. S druge strane pretplaćivanjem OIA na OFM omogućeno je upravljanje analizom preko sučelja aplikacije.

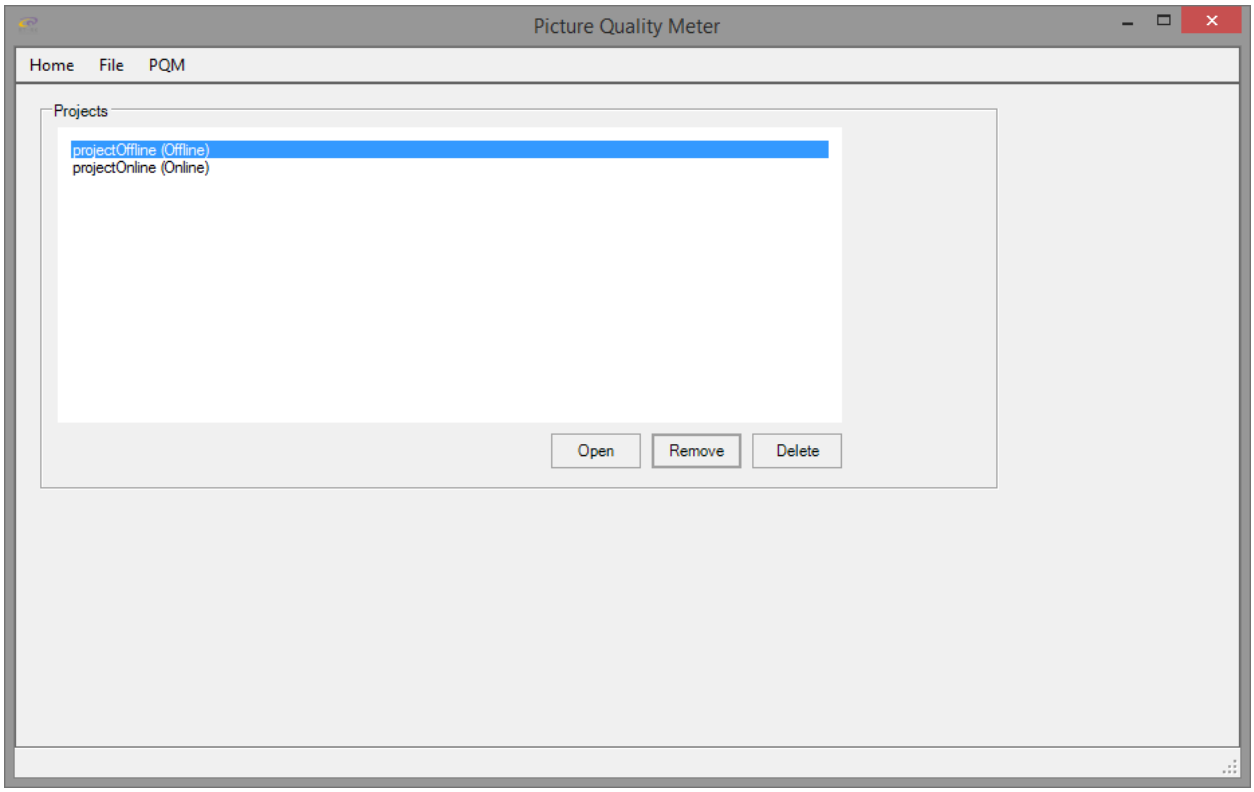
Nakon što korisnik pokrene analizu, kreira se nova nit u kojoj se izvršava analiza onako kako je opisano u potpoglavlju 4.3.9. Dakle, unutar OIA metode za analizu kreira se objekt OR zatim se inicijalizira prvi ulaz sa liste odabranih ulaza. Iznad dva različita ulaza postoji apstraktna klasa koja kao u slučaju s projektima omogućava primjenu svojstva polimorfizma, tj. programski postupak analize je neovisan o tipu projekta. Najprije se ulaz inicijalizira, zatim se iz liste algoritama određuje koji točno format okvira je potreban pojedinom algoritmu za analizu. Zatim se postavlja visina i širina okvira koji je predmet analize. Nakon toga se poziva funkcija ulaza koja za povratnu vrijednost vraća pokazivač na okvir odgovarajućeg formata. Pokazivač se dalje prosljeđuje svakom objektu OA. Osim pokazivača OA algoritmima se postavlja i redni broj okvira. Nakon što je sve podešeno svakom OA objektu sa liste poziva se funkcija koja kreira nit u kojoj se izvršava analiza. Nakon što su svi algoritmi pokrenuti, OIA metoda čeka da svaka niti završi sa svojim radom. Nakon što svaki algoritam provede analizu, poziva se metoda OA objekta koja vraća rezultat i OIA metoda ga zapisuje u strukturiranom formatu u datoteku s rezultatima koja se nalazi na putanji definiranoj u OR objektu (vidi točku 4.2.5). Nakon što je završena analiza ulaza OR objekt se dodaje na listu rezultata referenciranog projekta. Cijeli postupak od kreiranja OR objekta do dodavanja na listu rezultata referenciranog projekta, u slučaju analize izvan-mrežnih ulaza, ponavlja se onoliko puta koliko je ulaza označeno prije analize.

Kada se provede analiza svih označenih ulaza određenog projekta, korisnik može pregledati datoteku s rezultatima tako što se vrati na PP i ondje na listi rezultata označi rezultat koji želi pregledati. Nakon zatvaranja programa najprije se serijalizira projekt na tvrdi disk, a

zatim se serijalizira OPD objekt. Ovime je omogućeno da ponovnim pokretanjem aplikacije resursi i dalje budu dostupni.

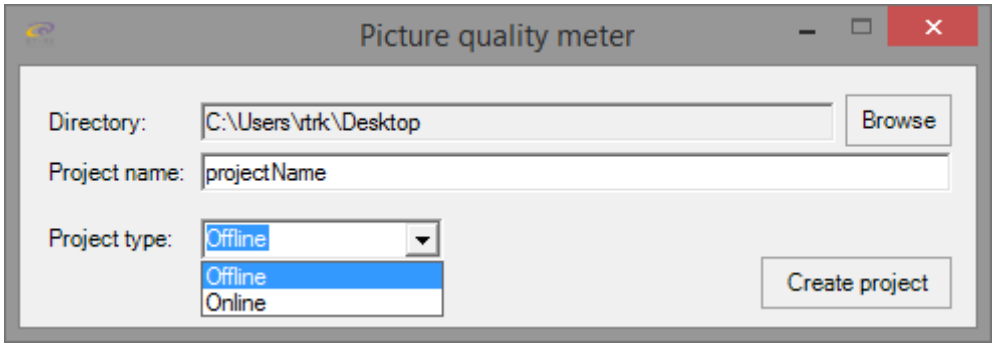
5. PRIKAZ RADA PQM APLIKACIJE

U ovom poglavlju biti će prikazan rad gotove PQM aplikacije. Nakon što se aplikacija pokrene pojavi se prozor sa slike 5.1.



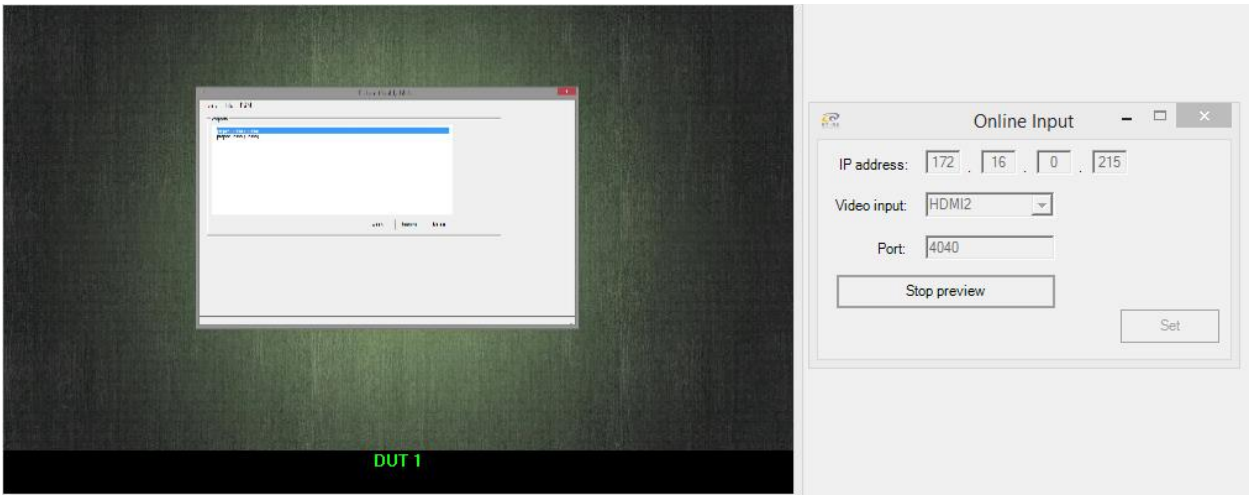
Sl. 5.1. Početni prozor PQM aplikacije.

Na slici 5.1. vidljiv je izbornik sa panelom PH. Na PH nalazi se lista projekata povezanih s aplikacijom. Korisnik može odabrati jedan od projekata sa liste povezanih projekata i odabrati neku od opcija: otvaranje, uklanjanje, brisanje projekta.



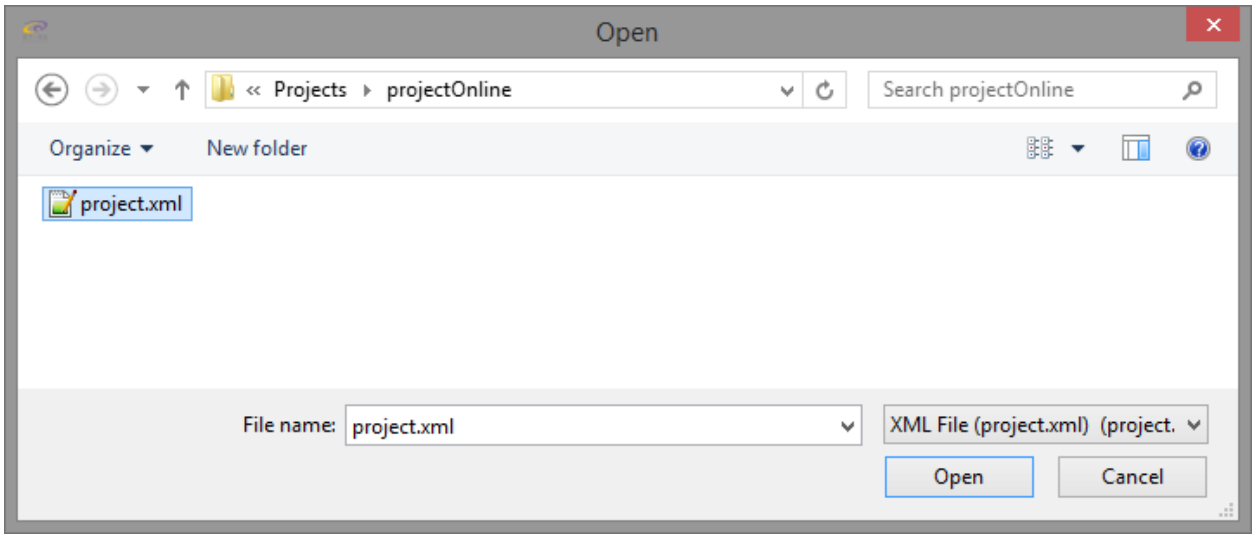
Sl. 5.2. Prozor za kreiranje projekta.

Ako korisnik u izborniku sa slike 5.1 odabere opciju kreiranja novog projekta izbacuje se prozor OFNP sa slike 5.2. U ovom prozoru korisnik definira putanju do projekta, postavlja naziv projekta i definira tip projekta. Nakon što sve unese odabire opciju za kreiranje projekta.



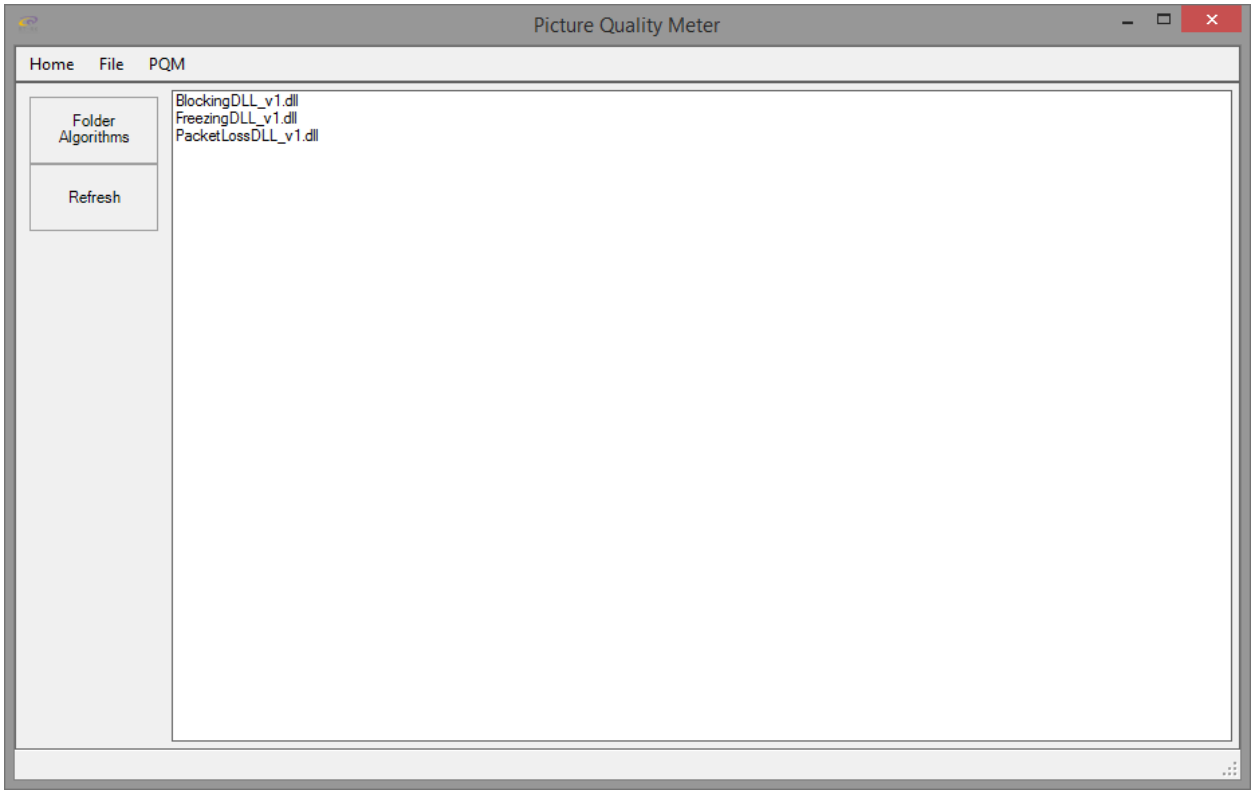
Sl. 5.3. Postavljanje postavki za mrežni ulaz.

Ako je korisnik za tip projekta postavio mrežni projekt, u sljedećem koraku mu se nudi mogućnost podešavanja mrežnog ulaza. Ova opcija nije obavezna, korisnik ju može preskočiti i kasnije definirati mrežni ulaz. Prilikom definicije mrežnog ulaza na prozoru OFONI korisnik unosi IP adresu na kojem se *grabber* nalazi, zatim definira tip ulaza na *grabber-u* i port preko kojeg se obavlja komunikacija. Nakon što su opcije postavljene korisnik prtiskom na gumb *Start preview* otvara novi prozor u kojem je prikazana slika sa *grabber-a*. Ako je korisnik pravilno podesio mrežni ulaz, tada je na zaslonu računala vidljiv video sadržaj koji dohvaća *grabber*. Pritiskom na dugme *Stop preview* prozor sa pred-pregledom se zatvara i korisnik potvrđuje postavke pritiskom na dugme *Set*.



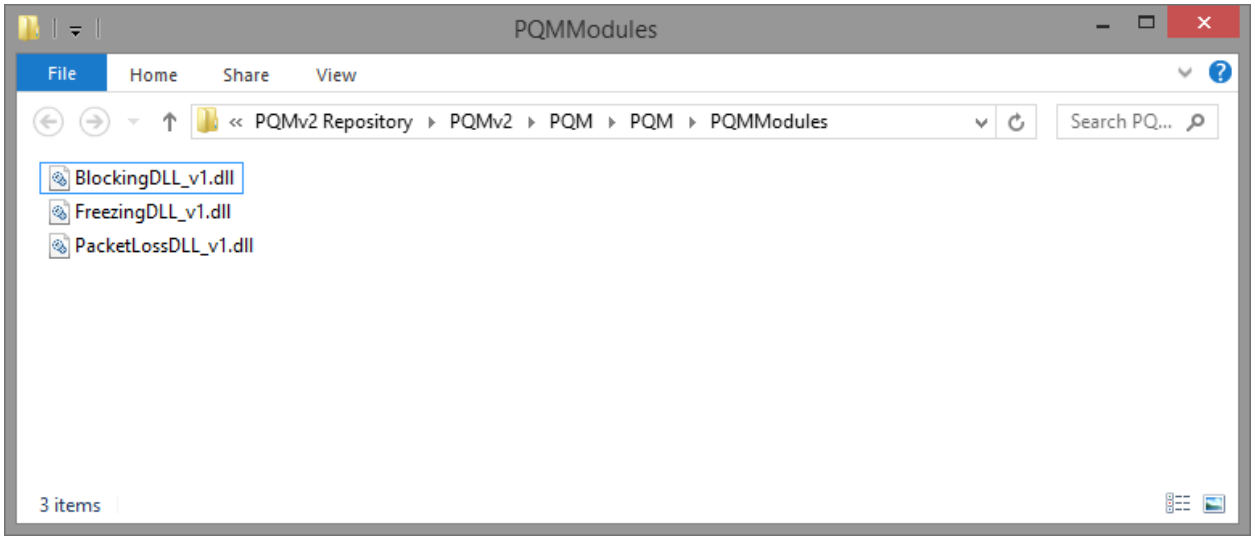
Sl. 5.4. Definiranje putanje i učitavanje postojećeg projekta u aplikaciju.

U slučaju da se projekt nalazi na računalu, ali nije dostupan u aplikaciji, npr. ako korisnik prebacuje projekte s jednog računala na drugo. Pritiskom na opciju dodavanja projekta otvara se pretraživač datoteka (Sl. 5.4.) preko kojeg korisnik dolazi do direktorija s projektom, označava datoteku *project.xml* u kojoj se nalazi serijalizirani projekt i otvara ju. Ako je datoteka ispravnog formata i deserijalizacija se uspješno obavi, projekt se povezuje s aplikacijom.

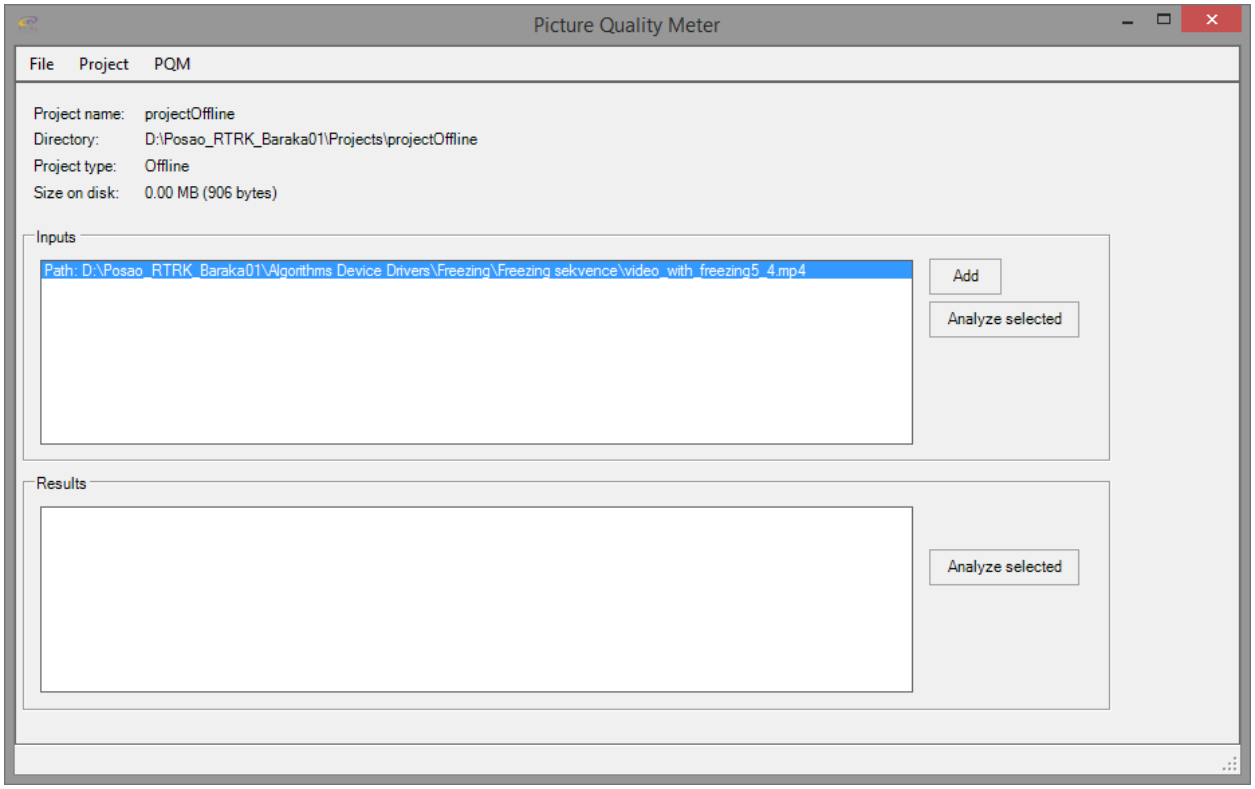


Sl. 5.5. Pregled uključenih algoritamskih biblioteka.

Kada korisnik odabere opciju za pregled algoritamskih biblioteka prikazuje se panel PA sa popisom svih dostupnih biblioteka (Sl. 5.5.). Osim pregleda dostupnih algoritamskih biblioteka korisnik ima opciju za uređivanje modula. Nakon što pritisne opciju za uređivanje biblioteka otvara se direktorij u kojem se algoritamske biblioteke nalaze (Sl. 5.6.). U tom direktoriju korisnik može dodavati i brisati module opcijama za dodavanje i brisanje datoteka.

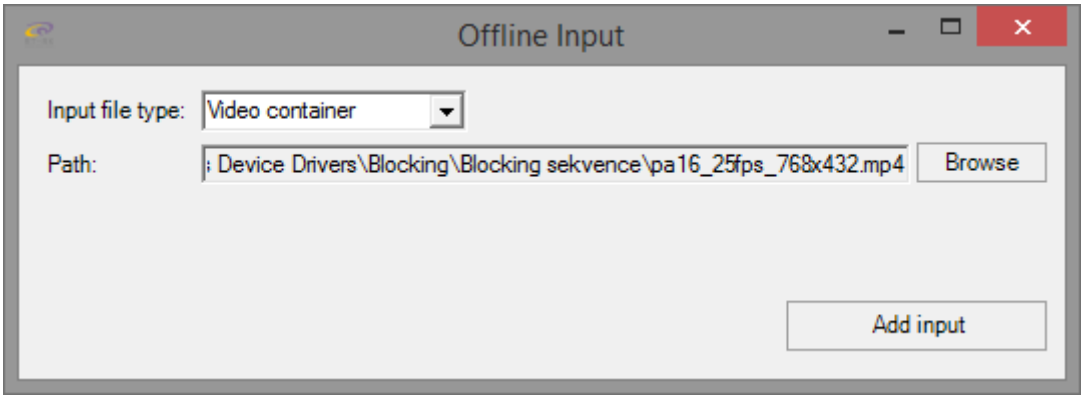


Sl. 5.6. Direktorij s algoritamskim bibliotekama uključenim u aplikaciju.



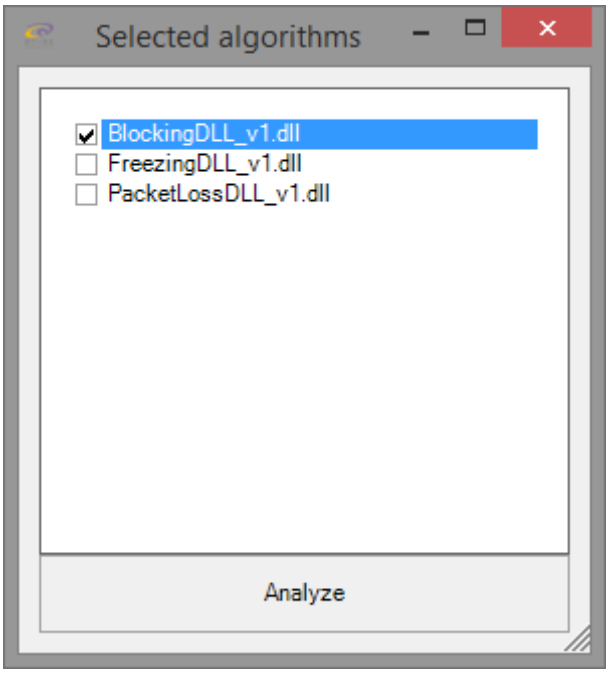
Sl. 5.7. Pregled ulaza i rezultata kod izvan-mrežnog projekta.

Nakon što korisnik kreira ili otvori projekt otvara se PP na kojem se nalaze informacije o projektu: naziv projekta, direktorij u kojem se nalazi projekt, tip projekta i veličina projekta na tvrdom disku (Sl. 5.7.). Osim informacija o projektu na PP se nalaze i dvije liste. Lista s popisom ulaza i lista s popisom rezultata. Pokraj liste s popisom ulaza nalaze se dvije opcije, opcija za dodavanje ulaza i opcija za analizu odabranih ulaza. Kraj popisa s rezultatima nalazi se opcija za analizu rezultata. Ovo se odnosi na izvan-mrežni projekt, a ako se radi o mrežnom projektu kraj popisa ulaza umjesto opcije za dodavanje ulaza nalazi se opcija za podešavanje postavki *grabber* uređaja.



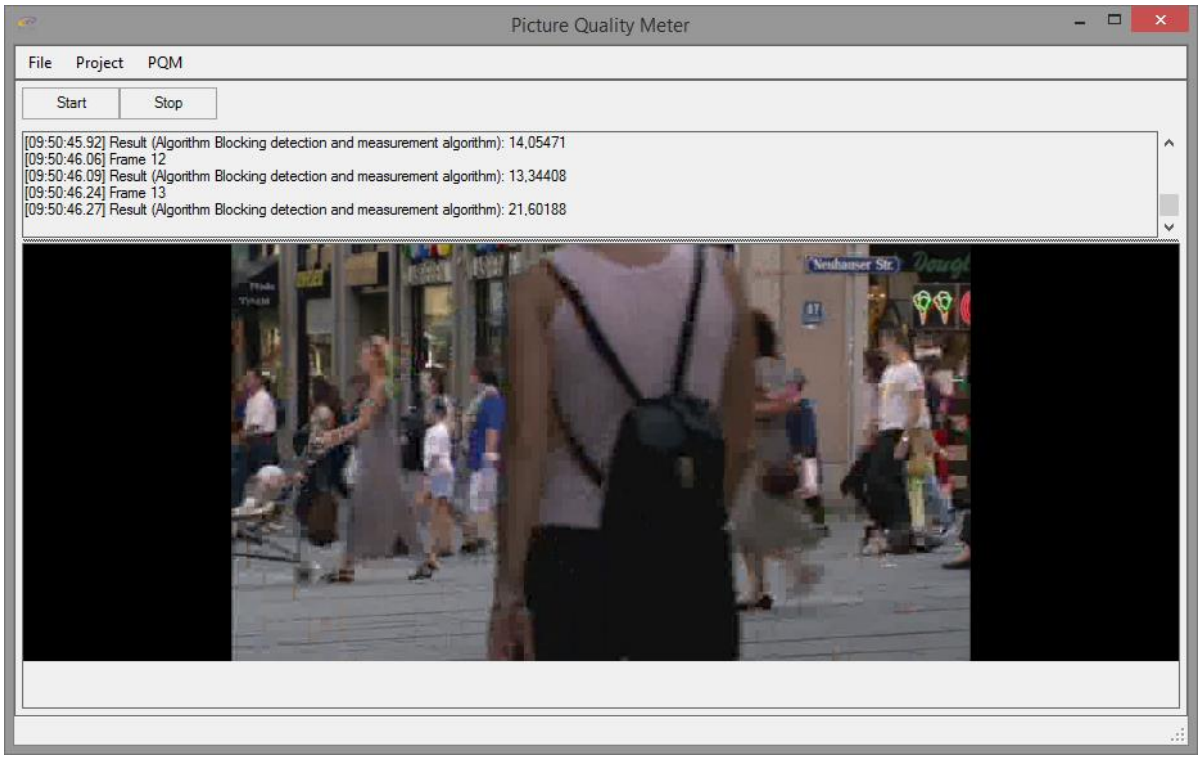
Sl. 5.8. Dodavanje izvan-mrežnog ulaza.

Kada korisnik otvori izvan-mrežni projekt i odabere opciju za dodavanje novog ulaza otvori se prozor OFOFI (Sl. 5.8.) na kojem korisnik odabire tip ulaza (za potrebe rada postoji samo opcija za dodavanje video kontejnera). Osim tipa ulaza korisnik postavlja i putanju do datoteke. Nakon što korisnik sve postavi i pritisne gumb *Add input*, ulaz je dodan te se pojavljuje na listi s ulazima. Kada korisnik otvori izvan-mrežni projekt i odabere opciju za izmjenu mrežnih postavki, otvara se OFONI prozor sa slike 5.3. u kojem korisnik postavlja postavke *grabber-a*.



Sl. 5.9. Odabir modula za analizu.

Nakon što korisnik odabere, sa liste ulaza, ulaze koje želi analizirati i pritisne gumb za pokretanje analize, pojavljuje se prozor OFSA sa slike 5.9. Na ovom prozoru nalazi se popis modula uključenih u aplikaciji. Korisnik odabire one module koje želi koristiti u analizi i nakon što označi sve module potvrđuje odabrano. Nakon ovog koraka odabrani ulazi i moduli su spremni za analizu, može pokrenuti analizu (Sl. 5.10.) pritiskom na gumb *Start*.



Sl. 5.10. Analiza ulaza.

Nakon što korisnik pokrene analizu na panelu PAV prikazuju se rezultati analize i prikazan je trenutno analizirani okvir. Nakon što analiza završi rezultat analize vidljiv je na panelu PP na listi rezultata. Korisnik sada može odabrati rezultat i otvoriti datoteku s rezultatima (Sl. 5.11.).

```
[Frame:          1][  4.71]
[Frame:          2][  3.82]
[Frame:          3][  3.15]
[Frame:          4][  3.12]
[Frame:          5][  3.21]
[Frame:          6][  3.20]
[Frame:          7][  3.33]
[Frame:          8][  3.38]
[Frame:          9][  3.35]
[Frame:         10][  3.68]
```

Sl. 5.11. Izgled datoteke s rezultatima.

6. ZAKLJUČAK

Razvojem pametnih telefona korisnicima je omogućeno da pregledavaju, snimaju, dijele video sadržaje u svakom trenutku. Zbog toga je proteklih nekoliko godina značajno porastao udjel prometa kojeg video sadržaj zauzima u globalnim podatkovnim mrežama. Osim porasta učestalosti pregledavanja i distribuiranja video sadržaja porasla je i potražnja za video sadržajima veće kvalitete. Zbog sve većih zahtjeva pojavljuju se problemi na infrastrukturi koja distribuira sadržaj korisnicima. Potrebna je sve veća propusnost pojasa pa se koriste sve veći stupnjevi kompresije. Veći stupnjevi kompresije za sobom vuku potrebnu za sve složenijim hardverom i softverom i veću mogućnost grešaka tj. pojave video artefakata. Budući da je hardver i softver sve složeniji mogućnost grešaka prilikom razvoja sve je veća. Zbog toga tvrtke koje se bave izradom hardvera i softvera, ovog tipa, imaju sve veću potrebu za kvalitetnim alatima s kojima će provoditi testiranje proizvoda. U ovom radu je prezentiran jedan od takvih alata. Prezentirana je PQM aplikacija koja ima mogućnost analize video sadržaja tj. detekcije video artefakata. Važno je naglasiti kako PQM aplikacija ne ocjenjuje kvalitetu video sadržaja nego samo detektira video artefakte. PQM aplikacija ima mogućnost analize izvan-mrežnog video sadržaja (video datoteka) i mrežnog sadržaja (živog video sadržaja dohvaćenog korištenjem *grabber* uređaja). Aplikacija omogućava kreiranje dva tipa projekta. Svaki tip projekta podržava jedan tip ulaza, izvan-mrežni projekt podržava izvan-mrežne ulaze, a mrežni projekt mrežne ulaze. Korisnik može dodavati i podešavati ulaze, može odabrati više ulaza za analizu koji će se slijedno analizirati. Rezultati se spremaju na disk i dostupni su korisniku za daljnju analizu. Aplikacija omogućava i dinamička proširivanja u vidu algoritamskih biblioteka za detekciju artefakata. Algoritamske biblioteke za analizu mogu se dinamički dodavati i brisati. Aplikacija sprema stanje rada pa nakon što korisnik ponovno pokrene aplikaciju, svi projekti, ulazi i rezultati koji su bili prije isključivanja aplikacije dostupni su i nakon što je aplikacija ponovno pokrenuta.

Tijekom izrade diplomskog rada pojavilo se nekoliko problema, primjerice s korištenjem gotovih biblioteka te problemi vezani uz dizajn aplikacije u okviru Visual Studia. Svi problemi su uspješno riješeni te je predloženo rješenje, tj. PQM aplikacija u potpunosti funkcionalna.

Zbog objektno orijentiranog pristupa programskoj implementaciji, PQM aplikacija se može relativno lagano proširivati dodatnim mogućnostima. Dodatne mogućnosti koje bi krajnjem korisniku omogućile lakši uvid u dobivene rezultate kao što je dodavanje interaktivnog grafa s

rezultatima, mogućnost vizualizacije artefakata, zatim spremanje pozicija artefakata, spremanje dijela videa dohvaćenog sa *grabber-a* kada su detektirani artefakti i brojne druge mogućnosti.

LITERATURA

[1] Cisco, “Cisco Visual Networking Index: Global Mobile Data Traffic Forecast Update, 2015–2020”, Link:http://www.cisco.com/c/en/us/solutions/collateral/service-provider/visual-networking-index-vni/mobile-white-paper-c11-520862.html (Zadnji pristup: 29.06.2016.)

[2] N. Teslic, V. Zlokolica, V. Pekovic, T. Teckan, and M. Temerinac, “Packet-loss error detection system for DTV and set-top box functional testing,” IEEE Trans. Consum. Electron., vol. 56, no. 3, pp. 1311–1319, Aug. 2010.

[3] M. Shahid, A. Rossholm, B. Lövström, and H.-J. Zepernick, “Noreference image and video quality assessment: a classification and review of recent approaches,” EURASIP J. Image Video Process., vol. 2014, no. 1, p. 40, 2014.

[4] RT-RK, “BBT”, Link: http://bbt.rtrk.com/audiovideo-analysis/. (Zadnji pristup 29.06.2016.)

[5] C. Chen and J. A. Bloom, “A Blind Reference-Free Blockiness Measure,” in Advances in Multimedia Information Processing-Pcm 2010, Pt I, vol. 6297, 2010, pp. 112–123.

[6] S. Wolf and M. Pinson, “A no reference (NR) and reduced reference (RR) metric for detecting dropped video frames,” Fourth Int. Work. Video Process. Qual. Metrics Consum. Electron., pp. 15–16, 2009.

[7] AcceptTV, “Video Quality Monitor”,Link: http://www.acceptv.com/en/products_vqm.php (Zadnji pristup: 29.06.2016.)

[8] MSU, “Video Quality Measurement tools”,
Link: http://compression.ru/video/quality_measure/index_en.html(Zadnji pristup: 29.06.2016.)

[9] Artefakt - Zamagljenost, Link: http://www.jhlabs.com/ip/gaussian_blur.jpg (29.06.2016.)

[10] Artefakt - Mreškanje, Link: https://en.wikipedia.org/wiki/Ringing_artifacts (29.06.2016.)

[11] .NET C++/CLI dokumentacija,
Link: https://msdn.microsoft.com/en-us/library/68td296t.aspx (07.09.2016.)

[12] FFMPEG dokumentacija, Link: http://www.ffmpeg.org/doxygen/3.0/ (07.09.2016.)

KRATICE

ACI - (engl. *abstract class input*) apstraktna klasa ulaz

ACP - (engl. *abstract class project*) apstraktna klasa projekt

CA - (engl. *class algorithm*) apstraktna klasa algoritam

CFM - (engl. *class form main*) klasa glavni prozor

CFNP - (engl. *class form new project*) klasa prozora za kreiranje projekta

CFOFI - (engl. *class form offline input*) klasa prozora za dodavanje izvan-mrežnog ulaza

CFONI - (engl. *class form online input*) klasa prozora za dodavanje/podešavanje mrežnog ulaza

CFSA - (engl. *class form selected algorithms*) klasa prozora za odabir algoritama

CG - (engl. *class grabber*) klasa *grabber*

CGS - (engl. *class grabber setting*) klasa postavke *grabbera*

CIA - (engl. *class inputs analyzer*) klasa analiza ulaza

CM - (engl. *class module*) klasa modul

CMI - (engl. *class module instance*) klasa instanca modula

COFI - (*engl. class offline input*) klasa izvan-mrežni ulaz

COFP - (engl. *class offline project*) klasa izvan-mrežni projekt

CONI - (engl. *class online input*) klasa mrežni ulaz

CONP - (engl. *class online project*) klasa mrežni projekt

CPD - (engl. *class program data*) klasa programski podaci

CPE - (engl. *class program enviroment*) klasa programska okolina

CPS - (engl. *class project strucutre*) klasa sa osnovnim informacijama o projektu

CR - (engl. *class result*) klasa rezultat

CSA - (engl. *class selected algorithm*) klasa označeni rezultat

CSE - (engl. class program enviroment) klasa programska okolina

OA - (engl. *object algorithm*) objekt algoritam

OFM - (engl. *object form main*) objekt glavni prozor

OFNP - (engl. *object form new project*) objekt prozor za kreiranje projekta

OFOFI - (engl. *object form offline input*) objekt prozor za dodavanje izvan-mrežnog ulaza

OFONI - (engl. *object form online input*) objekt prozor za dodavanje/podešavanje mrežnog ulaza

OFSA - (engl. *object form selected algorithms*) objekt prozor za za odabir algoritama

OG - (engl. *object grabber*) objekt *grabber*

OGS - (engl. *object grabber settings*) objekt postavke *grabber-a*

OIA - (engl. *object inputs analyzer*) objekt analiza ulaza

OM - (engl. *object module*) objekt modul

OMI - (engl. *object module instance*) objekt instanca modula

OOFI - (engl. *object offline input*) objekt izvan-mrežni ulaz

OOFP - (engl. *object offline project*) objekt izvan-mrežni projekt

OONI - (engl. *object online input*) objekt mrežni ulaz

OONP - (engl. *object online project*) objekt mrežni projekt

OPD - (engl. *object program data*) objekt programski podaci

OPE - (engl. *object program enviroment*) objekt programska okolina

OPS - (engl. *object project structure*) objekt sa osnovnim informacija o projektu

OR - (engl. *object result*) objekt rezultat

OSA - (engl. *object selected algorithm*) objekt označeni algoritam

PA - (engl. *panel algorithms*) panel algoritmi

PAV - (engl. *panel analyze video*) panel analiza videa

PH - (engl. *panel home*) panel početni prikaz

PP - (engl. *panel project*) panel projekt

PQM - (engl. *picture quality meter*) aplikacija za analizu videa

RML - (engl. *reference algorithms list*) referenca na listu s algoritmima

ROIA - (engl. *reference object inputs analyzer*) referenca na objekt za analizu videa

ROP - (engl. *reference object project*) referenca na objekt projekt

ROPD - (engl. *reference object program data*) referenca na objekt programski podaci

SAŽETAK

Potražnja za videom i aplikacijama kojima je osnova multimedijski sadržaj u velikom je porastu u zadnjih nekoliko godina. Zbog sve veće potražnje za multimedijom pogotovo video multimedijom postoji potreba za pouzdanim alatom s kojim će se procijeniti kvaliteta videa koji se distribuira do krajnjeg korisnika. Iako u znanstvenim krugovima, pitanje analize videa zauzima mnogo pažnje alati za analizu videa su još uvijek vrlo rijetki. U ovom radu predstavljena je aplikacija koja omogućava bez referentnu detekciju video artefakata u stvarnom vremenu i koja je namijenjena za korisnike koji rade na produkciji videa, distribuciji videa, proizvodnjom hardvera, odnosno pripadajućeg softvera i znanstvenim istraživanjima. Aplikacija ima mogućnost dodavanja i brisanja algoritamskih biblioteka koje se koriste za analizu videa. Omogućava analizu rezultata i zapis strukturiranog rezultata na disk koji se mogu koristiti u daljnjoj analizi videa i uređaja za reprodukciju video sadržaja.

Ključne riječi: aplikacija, bez-referentno, analiza videa, detektiranje artefakata

ABSTRACT

Picture quality meter - No-reference video artifact detection tool

The demand for a wide range of video and multimedia applications is growing extremely fast in recent years. In such a diverse usage of digital video, there is a need for reliable tool for testing video quality by different parties involved in video content delivery to end users. While video quality methods are gaining quite considerable attention in scientific research, practical tools for video quality are still rare. This thesis presents an application for real-time no-reference video artifact detection that can be used in different scenarios like video equipment testing, network monitoring, video hardware/software development and scientific research. The application has some unique features allowing user customization in terms of used no-reference methods, giving the user deeper insight into the obtained results and easier usage when dealing with in field video equipment testing.

Keywords: application, no-reference, video analysis, artifact detection

ŽIVOTOPIS

Ivan Bošnjaković rođen 16. srpnja 1992. godine u Aarau, Švicarska. Do 1999. godine živi u Švicarskoj s obitelji, nakon čega se vraća u Hrvatsku i kreće u prvi razred osnovne škole. Nakon završetka osnove škole upisuje Prirodoslovno-matematičku gimnaziju u Našicama. Tijekom srednje škole pokazuje interes za računalnim programiranjem. Sudjeluje na školskim i županijskim natjecanjima iz informatike i na Hrvatskom otvorenom natjecanju u informatici. Nakon završetka srednje škole 2011. iste godine upisuje preddiplomski studij na Elektrotehničkom fakultetu u Osijeku kao redovan student. Preddiplomski studij završava u roku 2014. godine sa završnim radom na temu Aplikacija za crtanje i prikaz jednostavnih 3D modela. Tijekom diplomskog studija radio je na projektima vezanim uz kolegije, radio je na projektu izrade aplikacije za naručivanje u kafićima u sklopu kolegija Internet objekata, zatim u sklopu kolegija, Internet programiranje i Vizualizacija podatka, radio je na projektu za prikaz statističkih podataka o populacijama svjetskih država, u sklopu kolegija Računalni sustavi stvarnog vremena radio je na projektu izrade meteorološke stanice, u sklopu kolegija Razvoj mobilnih aplikacija radio je i na projektu za android sustave i dr. Na kraju prve godine diplomskog studija prijavljuje se za stipendiju tvrtke RT-RK koju i dobiva. Početkom druge godine diplomskog studija kao stipendist započinje i sa radom u tvrtki RT-RK. Ondje produbljuje znanje iz programiranja, sluša tečajeve iz osnovnog i višeg C programiranja. Radi na izradi softvera za *Set-top box* uređaje u sklopu kolegija Digitalna videotehnika. Početkom drugog semestra započinje s radom na projektu tvrtke, započinje s radom na PQM (engl. *picture quality meter*) aplikaciji, aplikaciji za analizu video sadržaja. Sa znanstvenim radom pod nazivom *Picture quality meter - No-reference video artifact detection tool* sudjeluje na Zinc, IEEE konferenciji u Novom Sadu u lipnju 2016. Ovaj diplomski rad je rezultat rada na tom projektu.

_____

Potpis