SVEUČILIŠTE U SPLITU

FAKULTET ELEKTROTEHNIKE, STROJARSTVA I BRODOGRADNJE

DIPLOMSKI RAD

FORENZIČKA ANALIZA DIGITALNOG VIDEO ZAPISA

MARIJETA BLEKAČIĆ

Split, rujan 2021.

S V E U Č I L I Š T E U S P L I T U
FAKULTET ELEKTROTEHNIKE, STROJARSTVA I BRODOGRADNJE

Diplomski studij: **Računarstvo**

Smjer/Usmjerenje: **/**

Oznaka programa: 250

Akademska godina: 2020./2021.

Ime i prezime: **Marijeta Blekačić**

Broj indeksa: 719-2019

ZADATAK DIPLOMSKOG RADA

Naslov: **Forenzička analiza digitalnog video zapisa**

Zadatak: Proučiti literaturu i istražiti metode forenzičke analize digitalnog video zapisa.Istražiti načine zapisa i ekstrakcije meta informacija u digitalnom video zapisu. Opisati tehnike forenzičke analize digitalnog video zapisa te na primjerima prikazati njihovu primjenu.

Rad predan:

Predsjednik
Odbora za diplomski rad:

Mentor:

prof. dr. sc. Sven Gotovac

izv. prof. dr. sc. Damir Krstinić

IZJAVA

Ovom izjavom potvrđujem da sam diplomski rad s naslovom FORENZIČKA ANALIZA DIGITALNOG VIDEO ZAPISA pod mentorstvom izv. prof. dr. sc. DAMIRA KRSTINIĆA pisala samostalno, primijenivši znanja i vještine stečene tijekom studiranja na Fakultetu elektrotehnike, strojarstva i brodogradnje, kao i metodologiju znanstveno-istraživačkog rada, te uz korištenje literature koja je navedena u radu. Spoznaje, stavove, zaključke, teorije i zakonitosti drugih autora koje sam izravno ili parafrazirajući navela u diplomskom radu citirala sam i povezala s korištenim bibliografskim jedinicama.

Studentica

Marijeta Blekačić

SADRŽAJ

1. UVOD

U današnje vrijeme, digitalna tehnologija je značajno napredovala pa tako i sam digitalni video zapis. Prelaskom s analognog na digitalni video zapis, video i njegova obrada postala je lako dostupna, kako velikim tvrtkama tako i običnim ljudima, pa ih danas svi koristimo na našim mobilnim uređajima ili osobnim računalima u obliku nula i jedinica. Na tržištu se nalaze razni brendovi, pa tako postoje i razni formati video zapisa koje podržavaju ti uređaji.

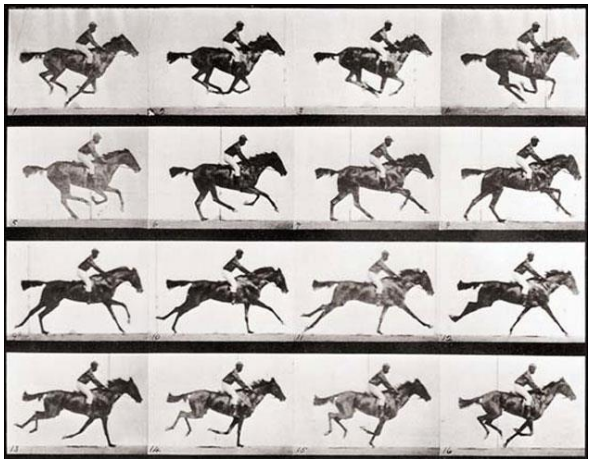
Video zapis je najkompleksniji oblik multimedije, a osim slike, zvuka, grafike i ostalih sadržaja, u sebi sadrži i meta podatke koji ga opisuju. Ti meta podaci mogu se na razne načine ekstrahirati iz video zapisa i na temelju njih možemo saznati razne informacije, od naziva datoteke, vremena snimanja, formata, do GPS oznaka gdje je video snimljen. Ovakvi podaci mogu se i mijenjati kako bi se lažirale informacije.

U ovom diplomskom radu, predstavljeni su najpoznatiji standardi i formati digitalnog video zapisa. Također, prikazani su primjeri operacija na video zapisu kao što su spremanje, editiranje i promjena rezolucije video zapisa, a svi kodovi pisani su u programskom jeziku Python. Osim toga, objašnjeni su načini pristupa meta podacima i mogućnosti koje ti podaci nude.

Za kraj, prikazan je i objašnjen postupak steganografije. Pomoću ove metode moguće je skriti informacije, kao što su slika, tekst ili neki video zapis, u drugi video zapis. Forenzičkom analizom moguće je saznati skrivene poruke unutar video zapisa i na taj način omogućiti tajnu komunikaciju bez sumnje treće osobe.

2. POVIJEST VIDEO ZAPISA

Jedan od prvih video zapisa snimljen je davne 1878. godine pod nazivom *Konj u pokretu* gdje je Eadweard Muybridge, tadašnji engleski fotograf, stavio 24 fotografije konja u niz koji je tada kreirao pokretnu sliku konja u galopu. Bio je to eksperiment kojim se htjelo utvrditi jesu li sve četiri noge konja dignute od tla, a sada ga se smatra prvim nijemim filmom.



Slika 2.1 Konj u pokretu

Međutim, prema Guinnessovoj knjizi rekorda, najstariji preživjeli film je iz 1888. godine kada je francuski izumitelj, Louis Le Prince, režirao *Vrtnu scenu Roundhay* koja traje samo 2,11 sekundi. Koristio je kameru s jedinim objektivom i traku filma, odnosno papir za snimanje. [1]

Slika 2.2. Vrtna scena Roundhay

Ipak, tehnologija nije davala velike mogućnosti, pa je sve do 1900-tih većina filmova bila kraća od minute.

2.1. Analogni video zapis

Tijekom 50-ih godina prošlog stoljeća nastali su analogni video zapisi. Analogni video je video sustav kojeg karakterizira analogni, odnosno kontinuirani, električni signal, čija amplituda i oblik sadrže informacije o osobinama slike koje se pohranjuju na magnetsku traku. [2]

Ono što je potaknulo nagli razvoj ovakvog tipa zapisa je njegova primjena u vojsci gdje se koristio za snimanje izlaznih signala radara. Samim time su se počeli razvijati i novi formati od kojih su se prvi koristili za prijenos uživo. Prvi video snimač je prenosio slike uživo, pretvarajući električne impulse fotoaparata, nakon čega su podaci spremljeni na magnetsku video kasetu. Tadašnje cijene video snimača bile su 50 000 američkih dolara, a video kaseta oko 300 američkih dolara po jednosatnoj roli. Kasnije su cijene padale, pa se 1963. godine pojavljuje i prvi video format, Ampex VR 1500/660, za potrošačko tržište čime se javlja zanimanje i kod ostalih proizvođača kao što su Sony, Hitachi i slično. Time kreće borba za tržište i razvoj raznih video formata. [3]

2.2. Digitalni video zapis

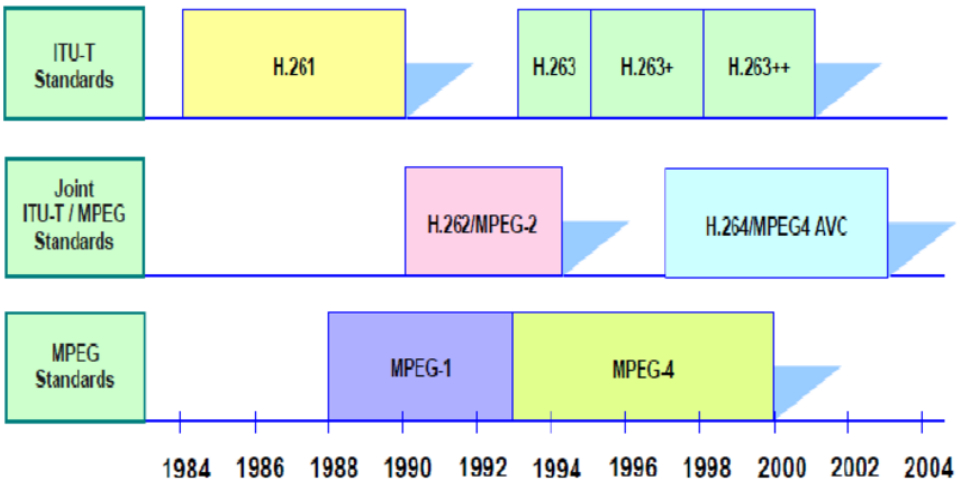
Digitalni video zapis se snima i pohranjuje u digitalnom formatu kao jedinice i nule, za razliku od analognog koji koristi niz fotografija, odnosno koriste se digitalni, a ne analogni signali. Podaci se s jedne strane obrađuju i spremaju u digitalnom obliku kako bi se olakšalo rukovanje s računalima, a s druge strane se gledatelju prikazuju u analognom obliku putem zaslona. [4]

Sama svrha digitalnog video zapisa je pohrana na kvalitetniji način u odnosu na analogni zapis. To, između ostalog, znači pohrana veće ili detaljnije količine informacija, tj. „bogatije“. Da bi to uspjeli treba prenijeti veliku količinu podataka. Tu nastaje problem kod digitalizacije video zapisa, odnosno njegovog pretvaranja u elektronički oblik jer se stvaraju veliki dokumenti koje treba skladištiti, prenositi i obrađivati čime se zauzima memorijski prostor računala što je jedan od razloga zašto se u početku digitalni video zapis nije mogao natjecati s analognim video zapisom. Također, problem je bio i nepraktična visoka brzina prijenosa podataka kako bi uopće mogli dobiti razumljivu poruku. Sve je to zahtijevalo ogromnu memoriju i visoke karakteristike komponenti računala. [3]

Međutim, 1970-ih godina počinje se koristiti kodiranje diskretne kosinusne transformacije (DCT), odnosno kompresije s gubitkom, čime je omogućen praktični digitalni video zapis. S vremenom, njegova cijena je bila sve niža, a kvaliteta sve veća od analogne tehnologije. 1997. godine dolazi do pojave DVD-a, a 2006. na tržištu se pojavljuje Blu-ray disk što utječe na pad prodaje video kaseta. Kako je tehnologija napredovala, tako je ljudima bilo sve bliže snimanje i pohranjivanje na osobnim računalima i mobilnim telefonima, ali i prijenos digitalnog video zapisa preko mreže. [3]

3. STANDARDI I FORMATI KOMPRESIJE VIDEO ZAPISA

Razvoj standarda kompresije video zapisa i tehnika koje su se koristile pri standardizaciji dolaze od dviju velikih međunarodnih organizacija ITU-T (*International Telecommunication Union*) i MPEG (*Moving Picture Experts Group*). Slika 3.1. prikazuje njihov razvoj po godinama.



Slika 3.1. Razvoj standarda kompresije video zapisa po godinama

3.1. H.261 standard

Razvoj standarda H.261 počeo je 1984. godine od strane ITU-T organizacije. Iako je završio 1990. godine, njegova zadnja revizija prihvaćena je tek tri godine nakon jer su prvoj verziji nedostajali važni elementi neophodni za potpunu specifikaciju interoperabilnosti. Službeni naziv ovog standarda je *Video kodek za audiovizualne usluge px64Kb/s,* a odnosi se na kodiranje video signala pri brzini od px64kbit/s, pa se često i naziva px64. [5]

Standard H.261 razvijen je za videokonferencije i videotelefoniju preko ISDN[1] linija kojima su brzine prijenosa višekratne od 64kbit/s. Maksimalno kašnjenje između kompresije i dekompresije ne smije biti veće od 150 ms jer bi tada bila ugrožena dvosmjernost, odnosno interaktivnost veze, pa ona ne bi bila primjerena za videokonferenciju ili videotelefoniju. [6]

[1] Linija od dvije vrste kanala, B kanal za prijenos podataka (64kbps) i D kanal (16 ili 64 kbits) za uspostavu i prekid veze, prijenos kontrolnih podataka i slično.

H.261 standard temelji se na:

- kompenzaciji pokreta čime se eliminira vremenska redundancija,
- transformacijskom kodiranju čime se eliminira prostorna redundancija.

Na istom osnovnom principu se temelje svi kasniji video koderi. [6]

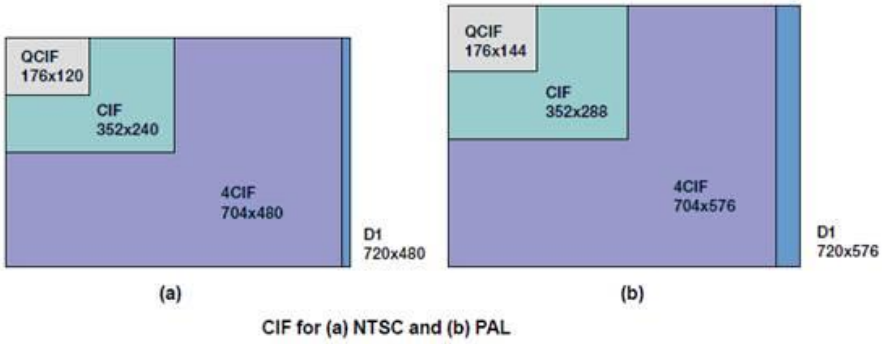
Jedan od problema koji se javljao kod standarda H.261 je usuglašenje standarda koji nisu bili kompatibilni, gdje s jedne strane imamo PAL (*Phase Alternate Line*) i SECAM (*Sequential Color and Memory)* koji su se koristili u europskim zemljama, a s druge strane NTSC (*National Television Standards Committee*) standard koji je korišten u SAD-u, Japanu i drugim zemljama. Rješenje problema pronašli su u preuzimanju vremenske rezolucije iz NTSC standarda i prostorne rezolucije iz PAL i SECAM standarda. Na taj način vrlo je precizno definiran ulazni i izlazni signal. [5]

Ulazni signal ima sljedeće karakteristike:

- 525 ili 625 linija kompozitni neisprepleteni video,
- 8 bita po uzorku,
- 29.97 okvira u sekundi,
- YCbCr model boja. [6]

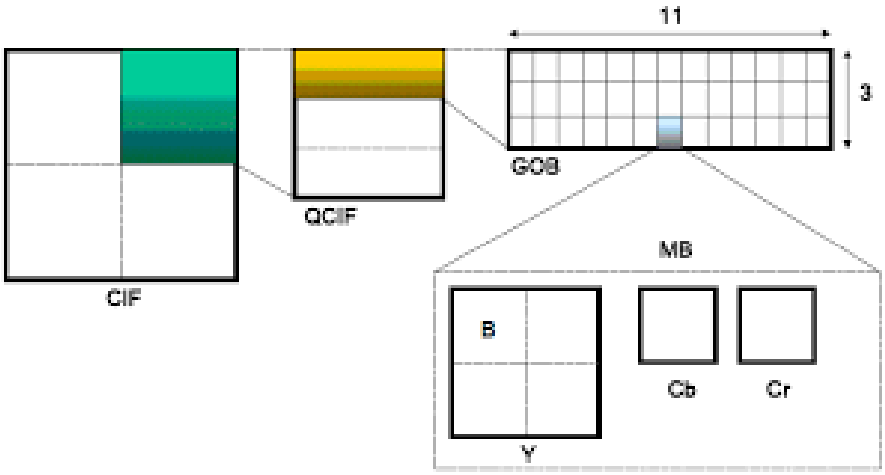
S druge strane, izlazni signal ima dva formata:

- CIF – *Common Intermediate Format* koji koristi 352x288 piksela za komponentu osvjetljenja i 180x144 piksela za komponentu boje,
- QCIF – *Quarter CIF* koji koristi duplo manje piksela u odnosu na CIF format.



Slika 3.2. Formati izlaznog signala

Izvorni videosignal u ova dva formata dijeli se u slike, grupe blokova (GOB – *Groups of Blocks*), makro blokove (MB – *Macro Blocks*) i blokove. Osnovna jedinica je blok i sastoji se od 8x8 elemenata slike, dok je makro blok sastavljen od 4 luminantna bloka (Y) i 2 krominantna bloka za svaku krominantnu komponentu (C_B, C_R). Grupa blokova sastoji se od 3x11 makro blokova. Slika se sastoji od 3 grupe blokova u QCIF formatu, a u CIF formatu od 12 grupa blokova. [6]

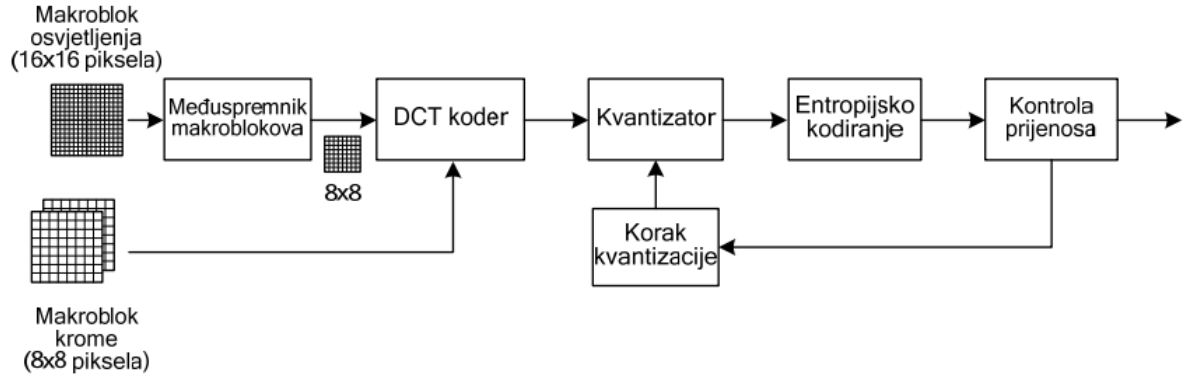


Slika 3.3. Hijerarhijska podjela slike u standardu H.261

Na ovaj način omogućen je visok stupanj sažimanja podataka izvorne slike.

H.261 standard koristi dva tipa kodiranja, a to su kodiranje unutar slike i kodiranje uz predviđanje. Razlikuju se po načinu kodiranja, odnosno okvirima koje koriste.

Kodiranje unutar slike naziva se još i *intra-okvirno kodiranje* jer se koristi I okvir (engl. *Intra frame, I frame*). Ovaj tip kodiranja zasniva se na diskretno kosinusnoj transformaciji (DCT) zbog čega se I-okvir uzima kao mirna slika, a za kodiranje se koristi linearna kvantizacija što znači da se svi koeficijenti kvantiziraju istom vrijednošću koja je konstantna. [6]



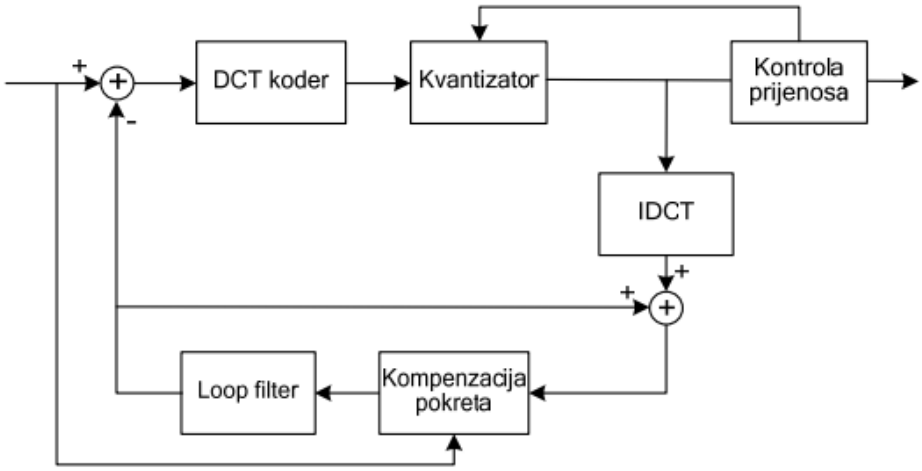
Slika 3.4. Blok dijagram kodiranja unutar slike kod standarda H.261

Kod kodiranja uz predviđanje koriste se P-okviri i za razliku od kodiranja unutar slike, ne kodira se sama slika, nego razlika u slikama, odnosno blokovima. Kodiranje se sastoji od diskretno kosinusne transformacije i diferencijalne impulsne kodne modulacije s predviđanjem pokreta. Proces se sastoji od sljedećih koraka:

1) Na svaki blok veličine 8x8 se primjeni diskretno kosinusna transformacija.
2) Koeficijenti transformacije se linearno kvantiziraju i multipleksiraju.
3) Slika se inverzno kvantizira i transformira primjenom inverzne diskretne kosinusne transformacije.
4) Slika se pohrani u memoriju kako bi se mogla koristiti u kodiranju slika za predviđanje sadržaja slike koja slijedi.

Kako bi se izvršilo predviđanje, uspoređuje se svaki makro blok u trenutno procesiranoj slici s odgovarajućim makro blokom iz prethodne slike. Ako je razlika između ta dva bloka manja od zadane vrijednosti, onda se za taj blok ne prenose podaci. U suprotnom, odnosno ako je ta razlika veća od zadane vrijednosti, onda se na razliku između ta dva bloka primjenjuje diskretna kosinusna transformacija, linearna kvantizacija i multipleksiranje.

Korak kvantizacije je usklađen s ispunjenosti međuspremnika, pa se u trenutcima njegovog preopterećenja on smanjuje kako bi se smanjila i sama količina kodiranih podataka. Kako se procjena pokreta od strane dekodera obavlja na rekonstruiranoj slici, tako se to i radi kod kodera, gdje je postavljena inverzna diskretno kosinusna transformacija (IDCT) kako bi se smanjilo ponavljanje greške. Također, koder sadrži i prostorni filtar za *glađenje* piksela kako bi se smanjila greška procjene pokreta. Slika 3.5. prikazuje shemu kodiranja uz predviđanje.



Slika 3.5. Kodiranje uz predviđanje kod standarda H.261

Osim H.261 standard, za videokomunikaciju i samo uspostavljanje, održavanje, raskidanje veze, multipleksiranje i slično, potreban je još jedan niz standarda. Neki od njih su H.221 koji se odnosi na sintaksu za multipleksiranje audio i video paketa, H.230 koji predstavlja protokol za uspostavu poziva, H.242, G.711, G.722 i ostali. [6]

3.2. H.263 standard

H.263 standard je nastao kao poboljšanje standarda H.261, a prihvaćen je 1996. godine. Dizajniran je za video usluge s niskom brzinom prijenosa informacija. S obzirom da se njegovo razvijanje temeljilo na iskustvima od prijašnjeg standarda H.261, tako je i njihova struktura slična pa video koder H.263 standarda koristi transformacijsko kodiranje za Intra-okvire i prediktivno kodiranje za Inter-okvire. [7]

Za razliku od H.261 standarda koji ima 2 formata, CIF i QCIF, H.263 standard ima 5 različitih formata koji su prikazani u Tablica 3-1.

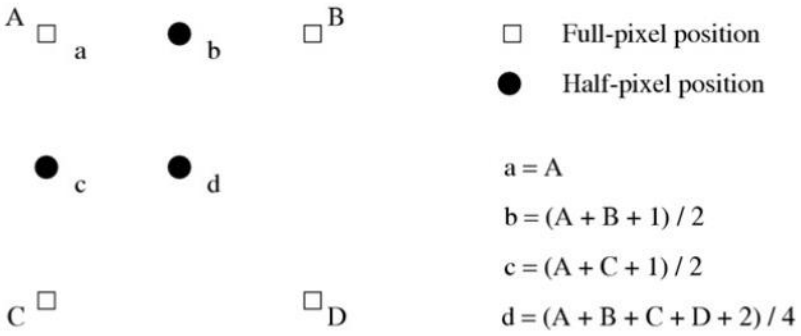
Tablica 3-1 Dimnezije h.263 formata

FORMAT	**OSVJETLJENJE**	**BOJA**
Sub-QCIF	128x96	64x48
QCIF	176x144	88x72
CIF	352x288	176x144
4CIF	704x576	352x288
16CIF	1408x1152	704x576

Karakteristike koje povećavaju efikasnost kompresije H.263 standarda su sljedeće:

- preciznost proračuna vektora pokreta od pola piksela,
- neograničeni vektor pokreta,
- napredno predviđanje,
- aritmetičko kodiranje,
- kombinacija P (*predicted*) i B(*bidirectional predicted*) okvira, odnosno PB okviri. [6]

Za razliku od H.261 standarda, koji koristi *loop* filter kako bi izgladio susjedne okvire, H.263 standard koristi preciznost proračuna vektora pokreta od pola piksela primjenjujući bilinearnu interpolaciju.



Slika 3.6. Bilinearna interpolacija

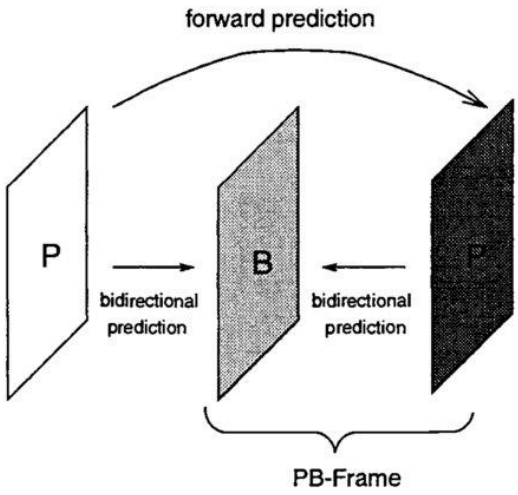
Osim toga, H.263 koristi neograničeni vektor pokreta koji se može pokazivati i izvan slike za razliku od H.261 standarda koji je to dopuštao samo unutar slike. To je moguće korištenjem rubnih piksela, pa se kod malih formata slike poboljšava efikasnost kodera pri pokretima na rubu slike, a s druge strane mogu se koristiti vektori pokreta koji su veći od makro bloka za situacije kada se kamera pomiče.

Još jedno poboljšanje je i mogućnost korištenja naprednog predviđanja u kojem se mogu koristiti 4 vektora pokreta po jednom makro bloku, za razliku od prijašnjeg jednog u standardu H.261. Na ovaj način se može preciznije predvidjeti jer se ono vrši na blokovima od 8x8 piksela, a ne 16x16.

H.263 standard koristi VLC (*Variable Length Coding*) kodiranje, odnosno kodiranje s promjenjivom dužinom. Ovaj način uklanja redundanciju u video podacima. Koristi tablicu u kojoj se nalaze binarne riječi za svaki simbol, pri čemu se vjerojatniji simboli kodiraju s manje

bita. Na ovakav način se svaki simbol kodira s fiksnim cjelobrojnim brojem bita, a u H.263 standardu se dozvoljava korištenje aritmetičkog kodiranja radi uklanjanja ograničenja fiksnog cjelobrojnog bita za simbole. [5]

Kod H.261 standarda koristili su se I i P okviri, a u H.263 pojavljuje se PB okvir koji se sastoji od dvije slike, tj. od P slike i B slike. One se kodiraju kao jedna cjelina. P-slika se predviđa na temelju prethodne dekodirane I ili P slike, a B-slika se bidirekcionalno predviđa na temelju prethodne dekodirane slike i na temelju trenutne P-slike iz PB okvira.



Slika 3.7. PB okvir

H.263 standard je poboljšan s dvije verzije koje su prihvaćene 1998. godine i 2001. godine, a poboljšanja se odnose na fleksibilnost video formata, skalabilnost, bolji PB okviri i intrakodiranje te uvođenje filtra za uklanjanje efekta bloka.

3.3. MPG (MPEG)

MPEG dolazi od kratice naziva radne grupe, *Moving Picture Ecperts Group*, koja je radila za ISO/IEC na razvoju standardizacije kompresije digitalnog video zapisa. U biti ne postoji razlika između MPG i MPEG formata, već je MPG samo starija ekstenzija za isti format jer se u starijim verzijama Windowsa koristilo proširenje datoteke od tri slova što je rezultiralo kraćenjem formata MPEG na MPG. MPEG je standardizirao razne formate kompresije i standarde od kojih svi koriste algoritme video kompresije s gubitkom temeljene na diskretnoj kosinusnoj transformaciji. [8]

Slika 3.8. MPEG File

3.3.1. MPEG-1

MPEG-1 pojavio se 1993. godine i predstavlja prvu generaciju iz skupine MPEG standarda. Odnosi se na *kodiranje pokretnih slika i pridruženog zvuka za digitalne medije za pohranu do 1,5Mbit/s.* Iako je sposoban za puno veće brzine prijenosa, najčešće je ograničen na 1,5 Mbit/s. Svoju primjenu je pronašao u Video CD-u, a koristio se i kao video zapis niske kvalitete na DVD-u te u uslugama digitalne satelitske/kabelske televizije prije uporabe MPEG-2. Kako bi obavio kompresiju, MPEG-1 potpuno odbacuje ili smanjuje pojedine informacije u frekvencijama i dijelovima slike koje ljudsko oko ne može percipirati. To obavlja u dijelovima videa gdje se uobičajeno javljaju ponavljanja kako bi kao rezultat dobio što bolju kompresiju podataka. Jedan od načina kako da se napravi što bolja kompresija bez velike osjetljivosti na ljudsko oko je *Chroma subsampling* koji se bazira na tome da je ljudsko oko puno osjetljivije na promjene u osvjetljenju nego promjene u boji. Tako se kod ovakve kompresije češće događa neslaganje u boji. [9]

MPEG-1 standard se sastoji od slijedećih dijelova:

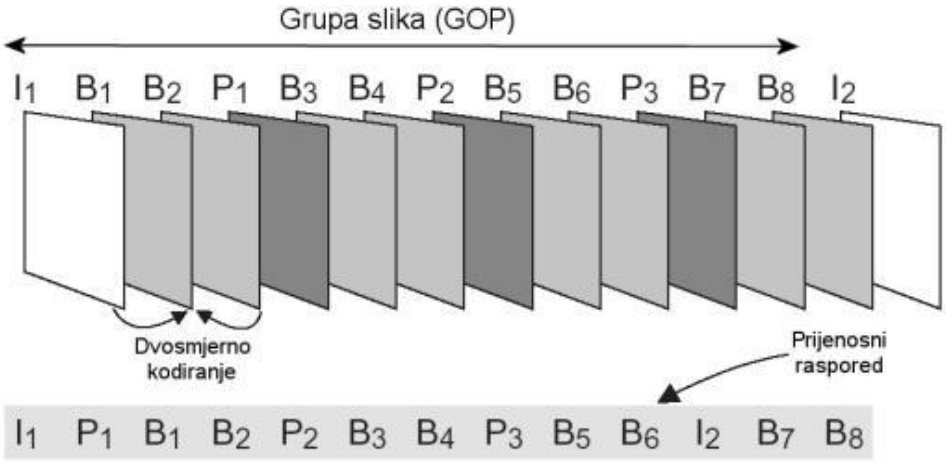
- Sistemski (podrška video i audio kodiranja),
- Video (digitalni medij za pohranu),
- Audio (Određuje visoko kvalitetni audio codec za pohranu audio medija),
- Ispitivanje usklađenosti (Ispitivanje ispravnosti implementiranja standarda),
- Softverska simulacija (Tehnički izvještaj). [10]

Danas je on najraširenij audio/video format koji je kompatibilan i koristi se u velikom broju proizvoda i tehnologija.

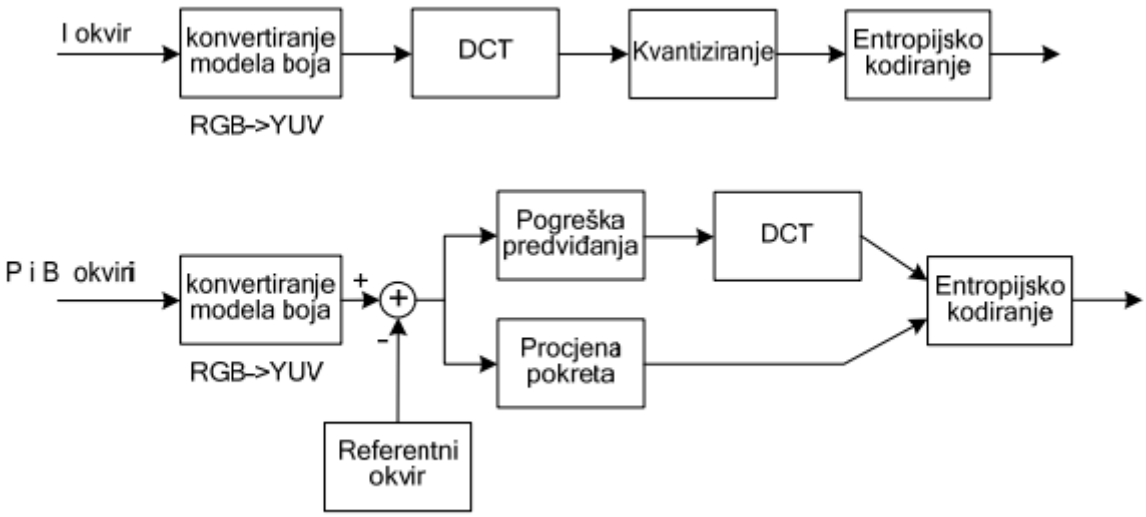
Kao i prethodni standardi, MPEG-1 standard se zasniva na eliminiranju prostorne i vremenske redundancije te se kod njega koriste tri vrste okvira, a to su *Intraframe, Predicted frame* i *Bidirectionally predictive-coded frame*.

I-okvir ili *intraframe* se kodira neovisno o drugim okvirima, koristi DCT kodiranje, ima najnižu kompresiju unutar MPEG-a, te pristupnu točku za slučajni pristup. P-okvir ili *Predicted frame* se kodira na temelju prošlih I ili P okvira, a kompresija je puno veća u odnosu na I-okvire. B-okvir ili *Bidirectionally predictive-coded frame* predstavlja razliku u odnosu na prethodne standarde, a kodira se u odnosu na prošle i buduće I ili P okvire i ima najveću kompresiju od svih tipova kodiranih okvira. Kod kodiranja I-okvira, makro blokovi se moraju kodirati u I modu pa se tako i makro blokovi P-okvira moraju kodirati u P ili I modu, a makro blokovi B-okvira moraju se kodirati u I, P ili B modu. [6]

MPEG-1 standard prilikom kodiranja podijeli video zapis na grupu slika (okvira) čija je veličina određena udaljenošću između dva najbliža I-okvira. Slika 3.9. prikazuje redoslijed okvira u MPEG-1 modu.

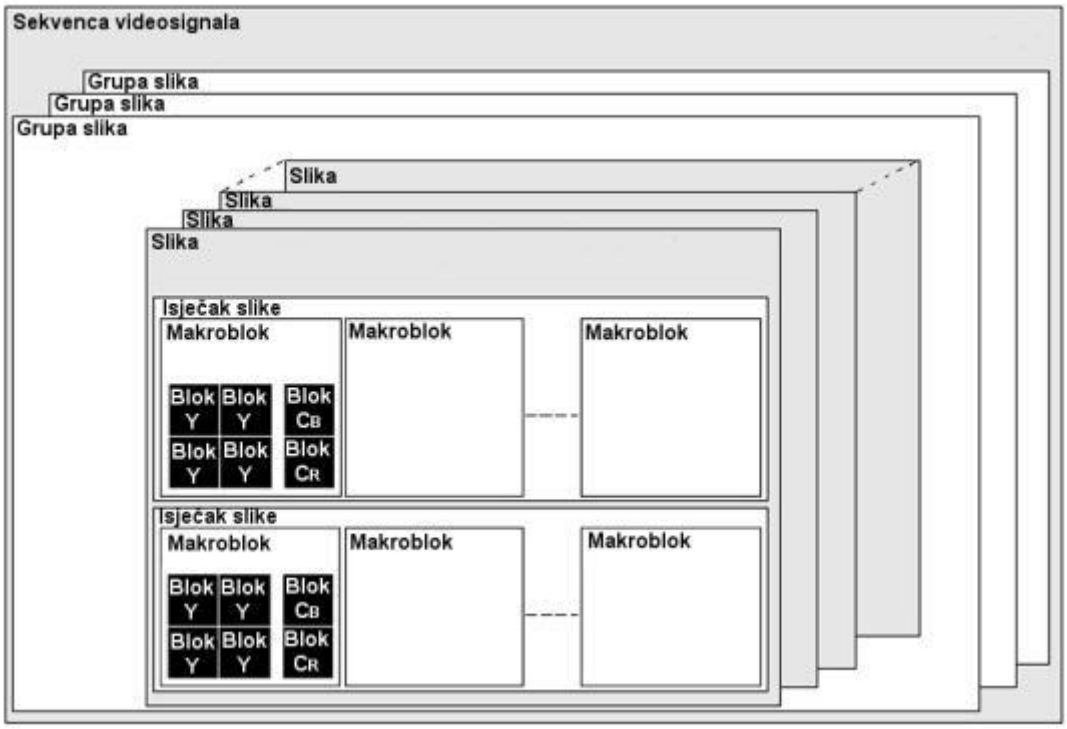


Slika 3.9. Redoslijed okvira u MPEG-1 modu



Slika 3.10. Kodiranje okvira kod MPEG-1 standarda

MPEG-1 standard je organiziran u 6 slojeva. Slika 3.11. prikazuje hijerarhijsku strukturu ovog standarda. Sastoji se od bloka koji predstavlja najmanju jedinicu veličine 8x8 elemenata slike. Iza njega slijedi makro blok kao osnovna jedinica za kodiranje veličine 16x16 elemenata slike. Isječak slike je vodoravni niz makro blokova, a on tvori sliku koja je osnovna jedinica u MPEG kodiranju. Niz od jedne ili više slika je grupa slika koje tvore sekvencu videosignala. [5]



Slika 3.11. Hijerarhijska struktura kod MPEG-1

3.3.2. MPEG-2

MPEG-2 dolazi dvije godine nakon, 1995. i odnosi se na *Generičko kodiranje pokretnih slika i pridruženih audio informacija.* Dosta je sličan MPEG-1, ali je mnogo šireg opsega i puno privlačniji potrošačima zbog svoje visoke razlučivosti i korištenja isprepletenog videa[2] koji koriste analogni televizijski sustavi za emitiranje. Svoju najveću primjenu MPEG-2 pronalazi kod digitalnih televizijskih signala koji se emitiraju zemaljskim, kabelskim i izravnim satelitskim televizijskim sustavima. Svi dekoderi koji su usklađeni sa standardom MPEG-2 mogu bez problema reproducirati MPEG-1 video. [10]

MPEG-2 standard u odnosu na MPEG-1 donosi razna poboljšanja, a najvažnija su sljedeća:

- Skalabilno kodiranje u koje se ubraja SNR skalabilnost, prostorna i vremenska skalabilnost
- Modovi predviđanja za polje/okvir za podršku isprepletenog video ulaza
- DCT kodna sintaksa za polje/okvir
- Korištenje posebnih kvantizacijskih matrica i alternativni redoslijed skeniranja
- Nelinearni kvantizacijski faktori za makro blokove. [6]

Osim toga, znatno je veća veličina okvira i poduzrokovanje može biti 4:2:2 i 4:4:4 što poboljšava kodiranje i podršku isprepletenom videu.

Kao što je navedeno, MPEG-2 koristi modove predviđanja za polja i okvire, točnije dva moda od kojih se jedan odnosi na okvire kao kod MPEG-1 standarda, a drugi se odnosi na polja. Kod takvog predviđanja gornje ili donje polje P-okvira predviđa se na temelju donjeg ili gornjeg polja I ili P okvira. S druge strane, kod B-okvira se svako polje bidirekcionalno predviđa na temelju 4 vektora pokreta, odnosno na temelju donjeg i donjeg polja prošlog i budućeg I ili P okvira.

Za razliku od MPEG-1 koji koristi ZigZag skeniranje, MPEG-2 omogućava alternativno skeniranje na razini sloja slike koje može biti bolje u odnosu na normalno skeniranje, a razlog tome je jer više nula može završiti na kraju korištenjem alternativnog skeniranja. Ono što

[2] Isprepleteni signal sadrži dva polja video kadra snimljena uzastopno.

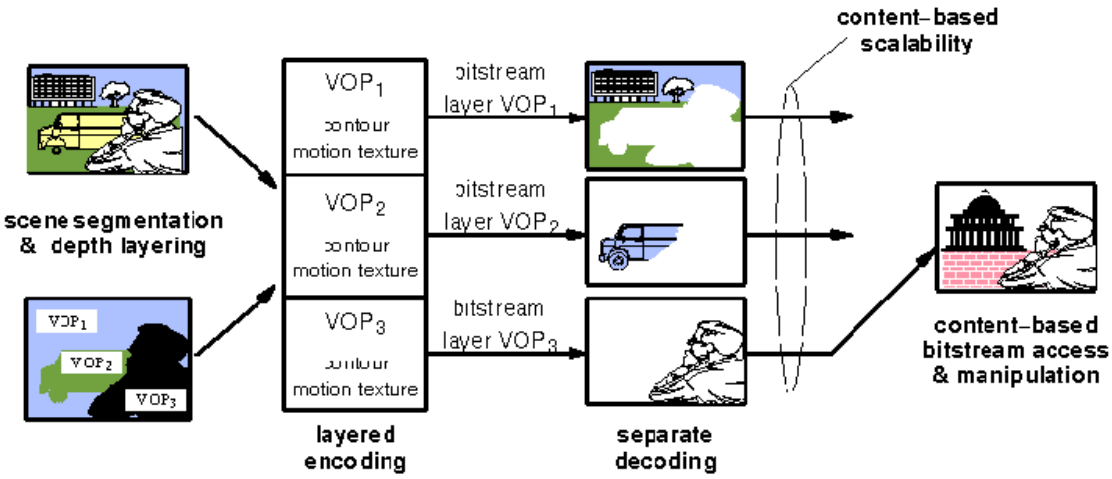
pomaže kod MPEG-2 standarda je i korištenje posebnih kvantizacijskih matrica koje pomažu kod dinamičnih videa, a različite su za intra i ostala kodiranja.

3.3.3. MPEG-4

1998. godine prihvaća se novi standard MPEG-4 koji se odnosi na kodiranje audiovizualnih objekata. On omogućava audio-video kodiranje namijenjeno raznim potrebama komunikacijskih, interaktivnih i difuznih modela servisa. Daje transparentne informacije koje se mogu interpretirati i prevesti u određene signalne poruke za bilo koju mrežu. [11]

MPEG-4 omogućava prikazivanje medija objekata koji se odnose na audio, vizualni i audiovizualni sadržaj, a oni mogu biti snimljeni kamerom ili mikrofonom ili se generirati pomoću računala. Također, omogućava multipleksiranje i sinkronizaciju podataka koji su povezani s medija objektima te opisivanje kompozicije objekata da bi se kreirao složeni medija objekt koji formira audiovizualne scene. S druge strane, MPEG-4 pruža visoki stupanj kompresije i dobru skalabilnost čime postaje više prilagodljiv na različite aplikacije.

Ovaj standard koristi koncept *Video object plane* (VOP) koji je prikazan na slici ispod gdje vidimo da se svaki okvir ulaznog signala dijeli u određene VOP-ove koji pokrivaju dio slike. Niz takvih VOP-ova naziva se video objekt (VO) i on se može kodirati u zasebni bitstream. U MPEG-4 standardu specificirana je sintaksa za multipleksiranje i demultipleksiranje koja omogućava prijemniku da poveže različite video objekte u jedan okvir što nije moguće u MPEG-1 i MPEG-2 standardima. [6]



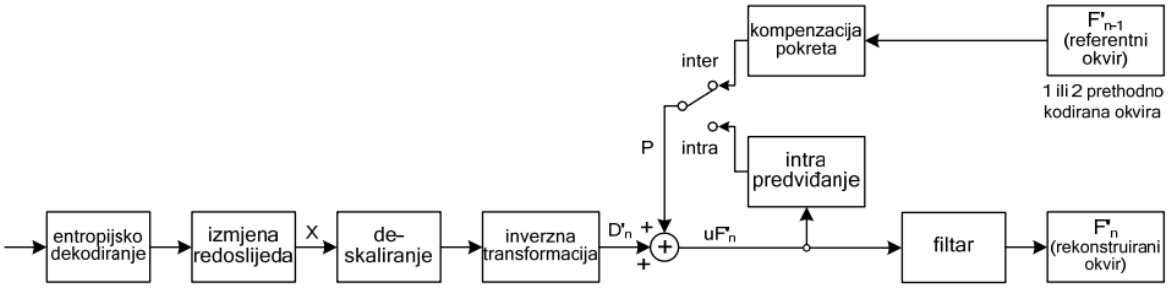
Slika 3.12. Video object plane koncept

3.4. H.264/MPEG AVC

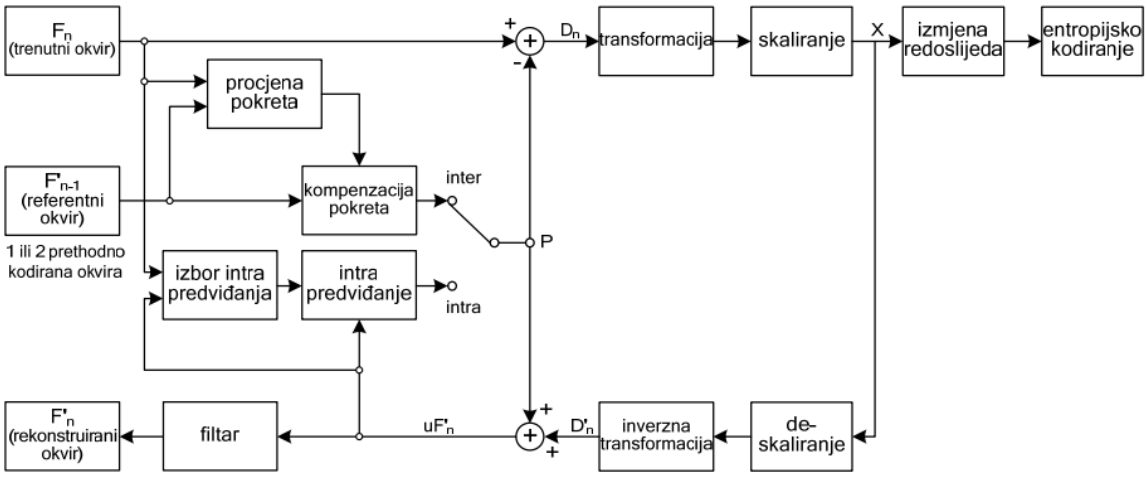
Standard H.264 poznat je i pod imenom MPEG AVC (MPEG *Advanced Video Coding*) i trenutno je najbolji video koder po stupnju kompresije. Predstavlja koder opće namjene koji se kreće od mobilnog videa s niskim brzinama prijenosa do televizije visoke rezolucije. Ideja je bila stvoriti standard koji može pružiti puno bolju kvalitetu video zapisa pri znatno manjih brzinama prijenosa u odnosu na prethodne standarde.

Takve karakteristike u kombinaciji s istim ili malim promjenama složenosti dizajna postignute su korištenjem cjelobrojne diskretno kosinusne transformacije (DCT), segmentacijom promjenjive veličine bloka i predviđanjem među slikama.[12] U usporedbi s MPEG-2 standardom kojem je vrlo sličan, H.264 kodira na brzini od jedne trećine ili jedne polovine za istu kvalitetu video zapisa. S druge strane, H.264 standard se uvelike razlikuje od MPEG-4 video kodera jer MPEG-4 stavlja naglasak na fleksibilnost, a H.264 na efikasnost i pouzdanost. Odnosno, MPEG-4 omogućava veliki spektar video podataka kao što su pravokutni okviri, video objekti, mirne slike te kombinaciju prirodnih i sintetičkih vizualnih informacija, dok se H.264 fokusira na efikasnost kodiranja pri čemu koristi hibridni pristup eliminacije redundancije podijeljen u četiri kategorije, a to su percepcija, prostor, vrijeme i statistika. [6]

Kao i prethodni standardi, H.264 definira sintaksu kodiranog toka bitova i metodu dekodiranja toka bitova, a ne koder i dekoder kao takav. Slika 3.13. prikazuje shemu H.264 dekodera, a Slika 3.14. shemu H.264 kodera.



Slika 3.13. Blok shema H.264 dekodera



Slika 3.14. Blok shema H.264 kodera

3.5. AVI

AVI (*Audio Video Interleave*) je format spremanja multimedijske datoteke koji je predstavljen 1992. godine od strane Microsofta. Ovakav tip datoteke može sadržavati audio i video podatke koji se mogu sinkronizirano reproducirati. [13] Slika 3.15. prikazuje ikonu AVI formata.

Slika 3.15. Ikona AVI formata

AVI je podformat formata RIFF *(Resource Interchange File Format)* koji dijeli podatke u blokove ili dijelove, gdje je svaki blok identificiran oznakom *FourCC*[3]. AVI datoteka se dijeli na dva obavezna dijela i jedan opcionalni. Prvi dio označen je oznakom *hdrl* i predstavlja zaglavlje datoteke gdje se nalaze svi meta podaci o videozapisu kao što je širina, visina i fps. Drugi dio je označen oznakom *movi* i on sadrži stvarne audio i video podatke. Treći dio, koji

[3] FourCC je znak od četiri bajta koji se koristi za jedinstvenu identifikaciju formata podataka

je izboran, označen je oznakom *idx1,* a sadrži listu blokova i njihove lokacije unutar datoteke. [14]

3.6. MOV

MOV format sprema datoteku u QTFF (*QuickTime File Format*) formatu koji je razvijen od strane Apple-a 1991. godine. Ideja razvijanja ovog formata je mogućnost podržavanja različitih vrsta medijskih zapisa radi prilagodbe prijenosa multimedijskih sadržaja između operativnih sustava, uređaja i aplikacija. Ovaj format pohranjuje jedan ili više zapisa na kojem se nalaze različite vrste podataka, kao što su zvuk, video i tekst. Najčešće se koristi za spremanje filmova i drugih video datoteka. Slika 3.16. prikazuje ikonu MOV formata. [15]

Slika 3.16. Ikona MOV formata

3.7. MP4

MP4, punog naziva *MPEG 4, dio 14* je multimedijski format razvijen u 2001. godine od strane ISO-a. Razvijen je kao proširenje MP3 formata koji se koristi za audio zapise. MP4 se koristi kao format za spremanje video i audio zapisa, ali ga je moguće koristiti i za spremanje drukčijih podataka kao što je tekst i slike. [16]

Nakon pojave na tržištu, ovaj format su počele koristiti mnoge tvrtke kako bi potrošačima omogućili lakši pristup digitalnom zvuku, pa je Apple iTunes trgovina postala glavno maloprodajno središte za kupnju pjesama. MP4 format omogućava kodiranje više meta podataka i na taj način čini datoteke mnogo korisnijima. Međutim, prelazak s MP3 na MP4 format je imao dosta problema zbog kompatibilnosti i mogućnosti reprodukcije na raznim uređajima. Kako MP4 uključuje zaštitu autorskih prava, tako se javio problem kod reprodukcije

datoteka koje su zaštićene autorskim pravima i onih koje nisu. Ali, većina problema kod ovog formata je zastarjela tijekom godina, dolaskom novih uređaja na tržište koji su bili prilagođeni ovom formatu. [17]

4. OSNOVNE OPERACIJE NA VIDEO ZAPISU

4.1. Učitavanje, prikaz i spremanje videa

U programskom jeziku Python video se učitava pomoću paketa *numpy* i *cv2*. Numpy je paket koji se koristi za znanstveno računanje u Pythonu. Njegovi nizovi omogućavaju napredne matematičke i druge vrste operacija na velikom broju podataka. Cv2 se koristi za analize slika i videa kao što je prepoznavanje lica, uređivanje fotografija, optičko prepoznavanje znakova i slično.

Za instaliranje paketa *numpy* u terminal se upiše naredba *py -m pip install opencv-python*, a za instalaciju paketa cv2 koristi se naredba *py -m pip install opencv-python*. Nakon što su uključeni paketi, funkcijom *cv.VideoCapture* stvaramo objekt koji može primati index ili naziv video datoteke ovisno je li želimo snimiti video zapis kamerom uživo ili učitati video zapis s računala.

Funkcija *isOpened()* vraća *True* ili *False* ovisno o tome je li se video učitao bez pogreške. Metoda *imshow* prikazuje video, ali u sivoj boji kako smo i postavili u kodu. Pomoću metode *cv2.waitKey* odabiremo odgovarajuće vrijeme jer ako je premalo video će biti prebrz, a s druge strane, ako je preveliko, video će biti prespor.

```
import numpy as np
import cv2 as cv
cap = cv.VideoCapture('video.mov')
if not cap.isOpened():
    print("Cannot open camera")
    exit()
while True:
    ret, frame = cap.read()
    if not ret:
        print("Can't receive frame (stream end?). Exiting ...")
        break
    gray = cv.cvtColor(frame, cv.COLOR_BGR2GRAY)
    cv.imshow('frame', gray)
    if cv.waitKey(25) and 0xff == ord('q'):
        break

cap.release()
cv.destroyAllWindows()
```

Slika 4.1. Funkcija za učitavanje i prikaz video zapisa u programskom jeziku Python

Slika 4.2. prikazuje kod koji koristimo ako želimo spremiti video koji smo snimili koristeći kameru od računala.

```
cap = cv.VideoCapture(0)
fourcc = cv.VideoWriter_fourcc(*'DIVX')
out = cv.VideoWriter('output.mp4', fourcc, 20.0, (640,  480))
while True:
    ret, frame = cap.read()
    cv.imshow('frame', frame)
    out.write(frame)
    if cv.waitKey(1) and 0xff == ord('x'):
        break
    if cv.getWindowProperty('frame', cv.WND_PROP_VISIBLE) < 1:
        break

cap.release()
out.release()
cv.destroyAllWindows()
```

Slika 4.2. Spremanje videa snimljenog kamerom računala

Kod spremanja koristi se metoda *VideoWriter* gdje navodimo naziv datoteke koju spremamo, *FourCC* kod, broj sličica u sekundi (*fps*) i veličinu okvira. *FourCC* kod označava specifikaciju kompresije i dekompresije videa, a koriste se DIVX, XVID, MJPG, W264, WMV1, WMV2 i sl.

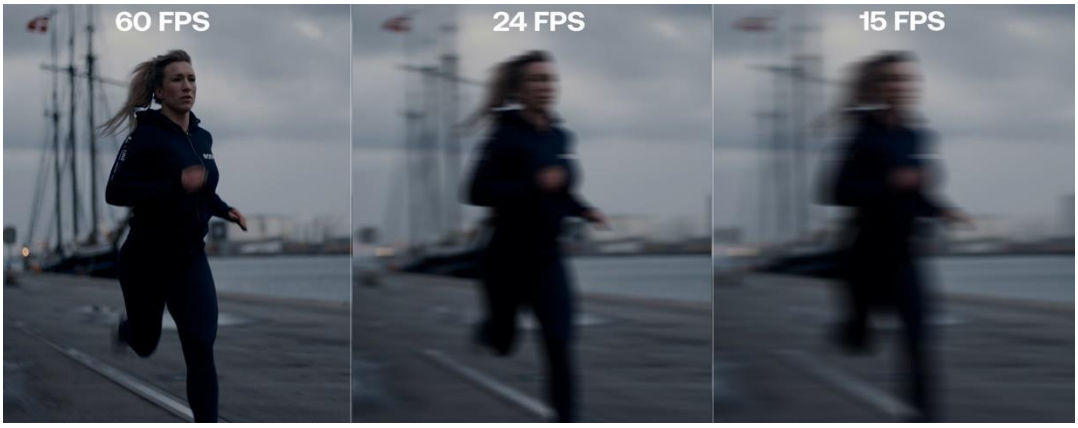
4.2. Okviri

S obzirom da se video zapis sastoji od okvira (engl. *frames*) koji se mijenjaju određenom brzinom kako bi dobili pokretnu sliku, onda video možemo podijeliti u okvire i spremiti ih u određenu mapu kako bi ih kasnije mogli mijenjati, uređivati ili koristiti u drugim kodovima. Dakle, video zapis nije kontinuirano snimanje. Brzina snimanja video zapisa izražava se u FPS (*Frame Per Seconds).*

FPS je različit ovisno o namjeni video zapisa koji se snima, pa tako imamo podjelu:

- Kinematografija – 24 fps
- Video zapisi, oglasi i Youtube – 30 fps za društvene medije, 60 fps ako ih treba uređivati
- Stariji izgled videa – 12-16 fps

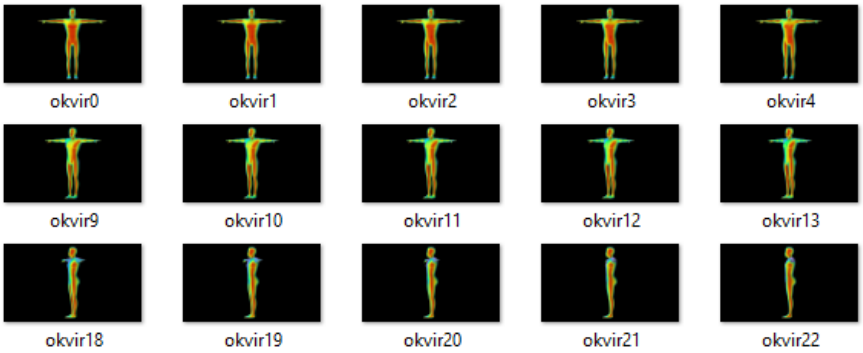
Ako je video zapis snimljen i reproduciran pri 24 fps, onda to znači da u jednoj sekundi imamo 24 različita okvira koji se prikazuju. Video zapisi uživo i oni koji u sebi imaju puno kretanja, kao što su video igre i sportski događaji snimaju se s većim fps-om. Ako želimo snimiti usporeni video onda koristimo 60 fps ili više. [18] Za ubrzana videa kao što je *time-lapse* koristimo malo okvira, npr. 8-10 fps jer bi bilo besmisleno koristiti više okvira. Slika 4.3. prikazuje razliku između 60, 24 i 15 fps.



Slika 4.3. Primjer korištenja različitog fps-a pri snimanju video zapisa

4.3. Spremanje okvira

Kako bi okvire mogli uređivati ili ih koristiti u drugim kodovima pojedinačno, možemo ih spremiti u određenu mapu. Prvo je potrebno uvesti pakete *cv2* i *os*. Nakon toga, stvara se objekt klase *VideoCapture* iz kojega ćemo dobiti okvire. Za ulaz postavljamo putanju do našeg video zapisa. Potrebno je provjeriti postoji li mapa u koju ćemo spremiti okvire i ako ista ne postoji, potrebno ju je napraviti. Kako bi mogli pratiti broj trenutnog okvira koji obrađujemo, definiramo varijablu *currentframe* i postavljamo je na nulu. Nakon toga pokreće se petlja koja završava kada više nema okvira za čitanje. U petlji se poziva metoda *capture.read()* koja vraća dvije vrijednosti. Prva se odnosi na to jesmo li uopće mogli pročitati okvir, a druga vrijednost na stvarni okvir. Ako je okvir pročitan, definiramo putanju, odnosno mjesto gdje ćemo ga spremiti. Spremanje radimo pomoću metode *imwrite()*. Nakon što smo spremili okvir, povećavamo varijablu *currentframe* i prelazimo na sljedeći okvir. Slika 4.4. prikazuje primjer spremanja okvira iz videa.



Slika 4.4. Okviri iz videa

```
import cv2
import os

#Otvori video
video= cv2.VideoCapture('C:/video.mp4')

#Provjeri postoji li file za spremanje okvira
try:
    if not os.path.exists('okviri'):
        os.makedirs('okviri')
except OSError:
    print ('Error: Creating directory of data')

currentframe = 0
while(True):
    ret1,frame1 = video.read()
    if ret1:
        name1 = './okviri/okvir' + str(currentframe) + '.png'
        cv2.imwrite(name1,frame1)
        currentframe += 1
    else:
        break

video.release()
cv2.destroyAllWindows()
```

Slika 4.5. Kod za spremanje okvira iz videa

4.4. Promjena rezolucije videa

Ako želimo saznati parametre veličine videa kao što su širina, visina i fps to se saznaje preko funkcije *get*.

```
width = cap.get(cv2.CAP_PROP_FRAME_WIDTH )
height = cap.get(cv2.CAP_PROP_FRAME_HEIGHT )
fps =  cap.get(cv2.CAP_PROP_FPS)
print('Širina: ',width)
print('Visina: ',height)
print('Fps: ',fps)
```

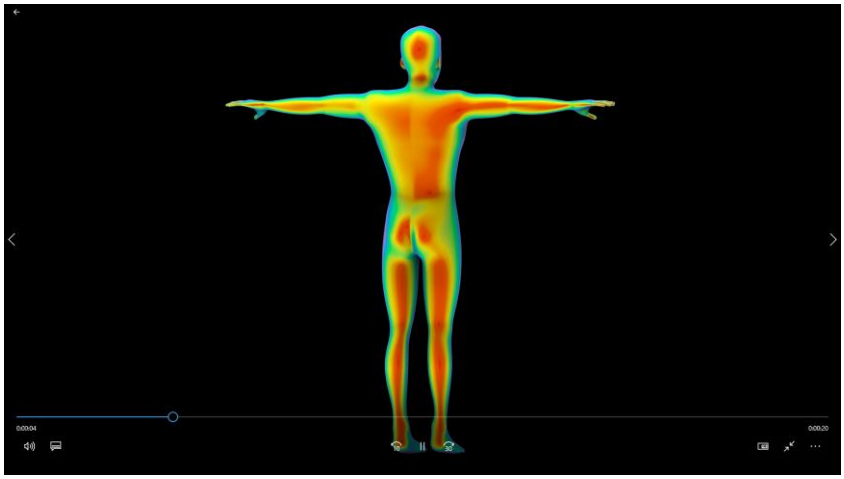
Slika 4.6. Širina, visina i fps video zapisa

Ovi parametri se mogu mijenjati. Da bi to napravili uključit ćemo novi paket *MoviePy*, koji se koristi za uređivanje videa kao što je rezanje, povezivanje, umetanje naslova, sastavljanje video zapisa i slično. Prvo je potrebno instalirati *MoviePy* u terminal pomoću naredbe *py -m pip install moviepy*, nakon čega se u kod uključuje paket *moviepy* pomoću linije koda *import moviepy.editor as mp.*

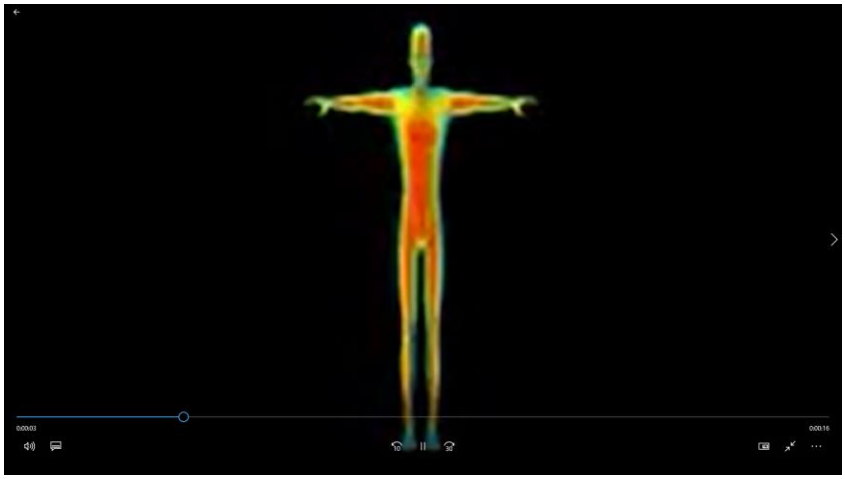
```
cap = mp.VideoFileClip("video.mp4")
capl=cap.resize((100,100))
capl.write_videofile("movie_resized.mp4")

print(capl.size)
```

Slika 4.7. Promjena rezolucije

Slika 4.8. i Slika 4.9 prikazuje promjenu koja se dogodila.



Slika 4.8. Izvorna rezolucija video zapisa

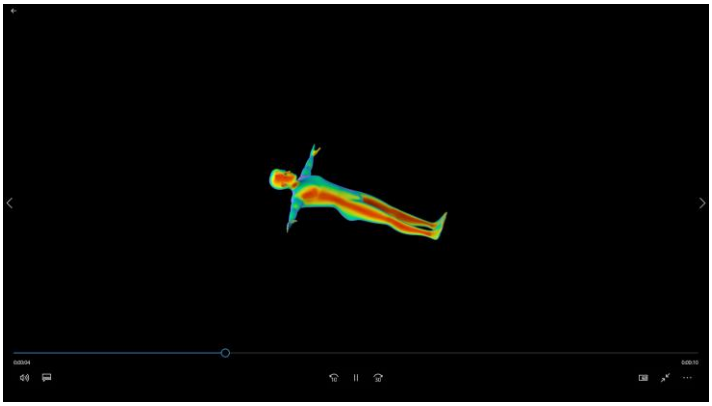


Slika 4.9. Rezolucija nakon promjene

Slika 4.11. prikazuje da video možemo i rotirati na sličan način.

```
cap = mp.VideoFileClip("video.mp4")
cap2 = cap.add_mask().rotate(72)
cap2.write_videofile("movie_rotated.mp4")
```

Slika 4.10. Kod za rotiranje videozapisa



Slika 4.11. Rotirani video zapis

4.5. Prostor boja

Kod video signala modeli boja se zasnivaju na osvjetljenju i razlici boja što znači da jedna komponenta predstavlja osvjetljenje, a druge dvije se odnose na boju, tj, razliku boja.

Osvjetljenje predstavlja crno-bijeli dio signala i odgovara Y komponenti u CIE[4] sustavu boja, ali ne u potpunosti jer se ovdje radi o sumi nelinearnih, gama korigiranih, RGB komponenti. Zbog toga se Y označava s Y' i naziva gama korigirano osvjetljenje *luma*. Informacije o boji dobivaju se izbacivanjem informacije o osvjetljenju što je najlakše napraviti oduzimanjem.

Y'UV model boja, kao i ostali modeli boja računaju komponentu osvjetljenja Y' kao

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

Za računanje razlike boja koristi se skaliranje

$$U = 0.492(B' - Y')$$

$$V = 0.877(R' - Y')$$

Y'IQ model boja koristi I i Q komponente umjesto U i V komponenti, koje su rotirane za 33°, pa imamo sljedeće formule

$$I = 0.596R' - 0.275G' - 0.321B'$$

$$Q = 0.212R' - 0.523G' + 0.311B'$$

Y'CbCr model boja se koristi kod 8-bitnog kodiranja boja kod većine standarda za kompresiju video i slikovnih signala kao što su JPEG, MPEG i sl. Y' komponenta može poprimiti jednu od 220 razina (16-235), a Cb i Cr komponente mogu poprimiti 4 razine više (16-240).
Ovaj model koristi sljedeće formule:

$$Y' := 219Y' + 16$$

$$Cb = 224(0.564(B' - Y)) + 128$$

$$Cr = 224(0.713(R' - Y)) + 128$$

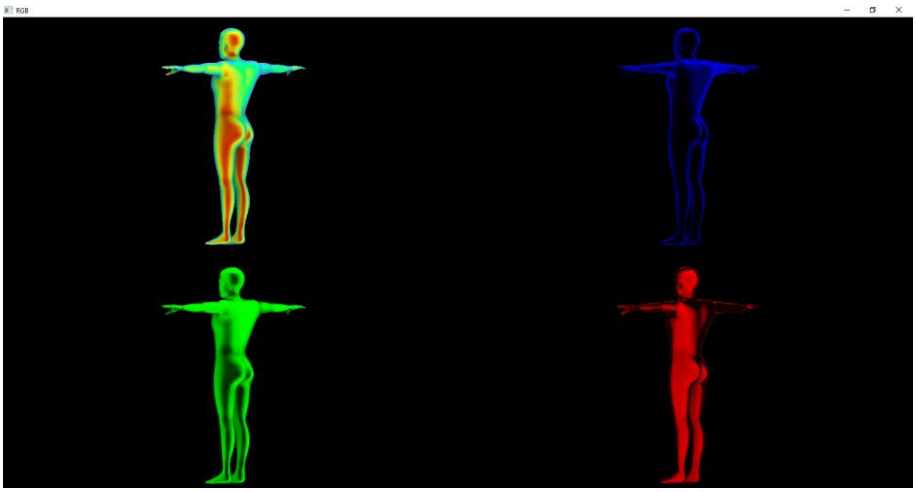
Prilikom učitavanja slike funkcijom *imread()* redoslijed boja nije RGB kao što je inače, nego je BGR, odnosno *blue, green, red*. To znači da se prilikom kodiranja treba paziti na redoslijed boja. Kako bi se dobila tri kanala iz video zapisa, koristi se funkcija *bgr* koja prikaže izvorni video zapis te njegov plavi, zeleni i crveni kanal na ekranu. [6]

Slika 4.12. prikazuje kod kojim se prikazuju BGR kanali videa, a Slika 4.13. prikazuje rezultat toga.

[4] CIE (International Commission of Illumination) sustav boja koji predstavlja sve boje koje ljudsko oko može vidjeti

```
def bgr(mirror=False):
    cap = cv2.VideoCapture('video.mp4')
    cv2.namedWindow('RGB',cv2.WINDOW_NORMAL)
    zeros = None
    while True:
        ret_val, frame = cap.read()
        if ret_val == True:
            if mirror:
                frame = cv2.flip(frame, 1)
            height, width, layers = frame.shape
            zeroImgMatrix = np.zeros((height, width), dtype="uint8")
            (B, G, R) = cv2.split(frame)

            B = cv2.merge([B, zeroImgMatrix, zeroImgMatrix])
            G = cv2.merge([zeroImgMatrix, G, zeroImgMatrix])
            R = cv2.merge([zeroImgMatrix, zeroImgMatrix, R])

            final = np.zeros((height * 2, width * 2, 3), dtype="uint8")
            final[0:height, 0:width] = frame
            final[0:height, width:width * 2] = B
            final[height:height * 2, 0:width] = G
            final[height:height * 2, width:width * 2] = R
            cv2.imshow('RGB', final)
        else:
            break
        if cv2.waitKey(1) & 0xFF == ord('q'):
            break
    cap.release()
    cv2.destroyAllWindows()
def main():
    bgr(mirror=True)
if __name__ == '__main__':
    main()
```

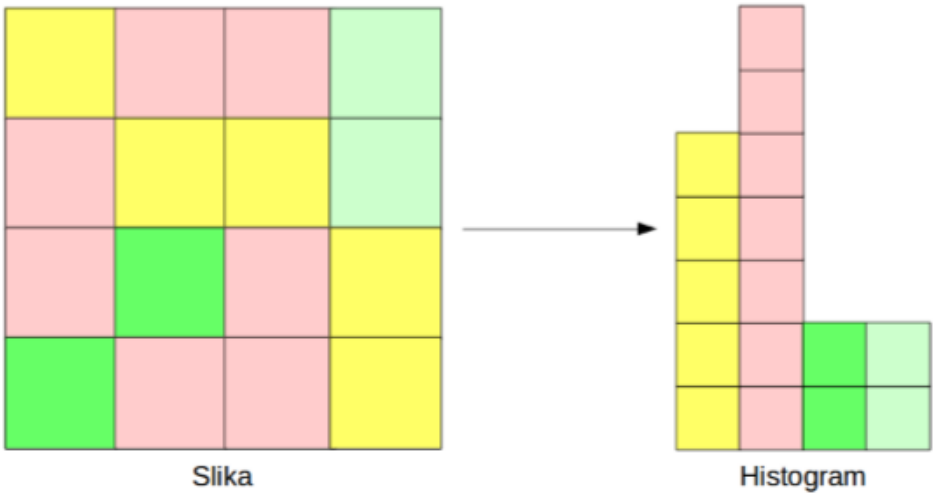
Slika 4.12. Kod za podjelu videa na BGR kanale



Slika 4.13. BGR podjela

4.6. Histogram

Histogram je vrsta grafa koja nam pruža vizualnu interpretaciju brojčanih podataka. Njihova uporaba najčešća je u statistici gdje grafički prikazuju brojeve i varijable kako bi prikazali vizualno jasne i poredane rezultate. [20] Kod digitalne slike histogram prikazuje broj piksela na slici po razinama intenziteta. Na primjer, ako se na slici nalazi 500 piksela koji imaju intenzitet 56, onda će pripadajući stupac histograma imati visinu 500. Slika 4.14 prikazuje shematski prikaz histograma. [19]



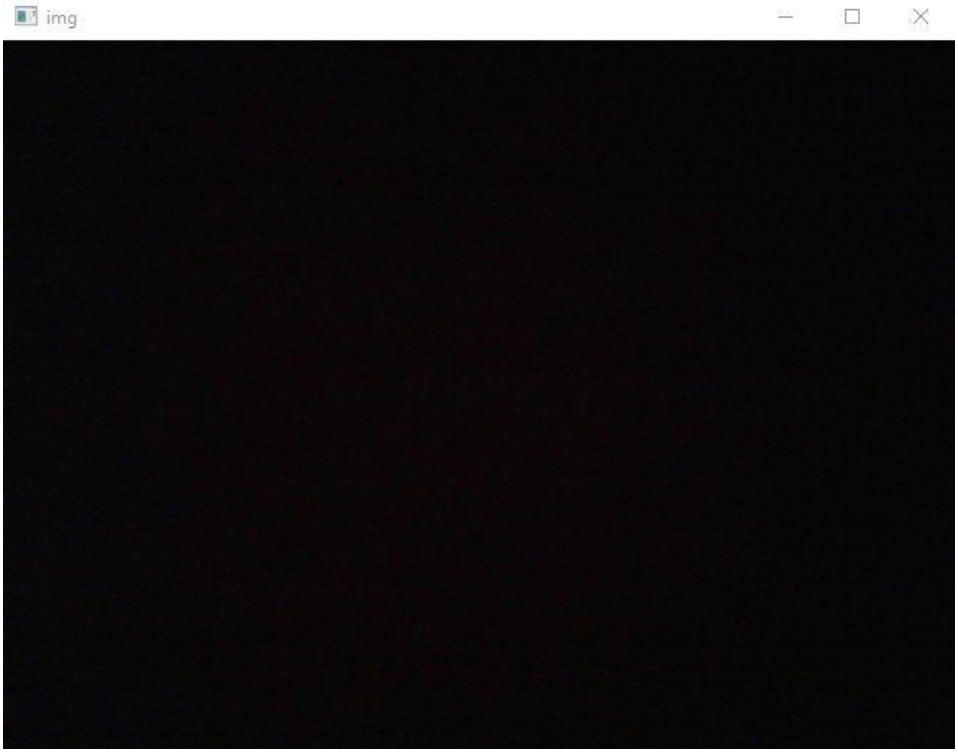
Slika 4.14. Shematski prikaz histograma

Histogram možemo napraviti i kod digitalnog video zapisa koristeći sljedeći kod.

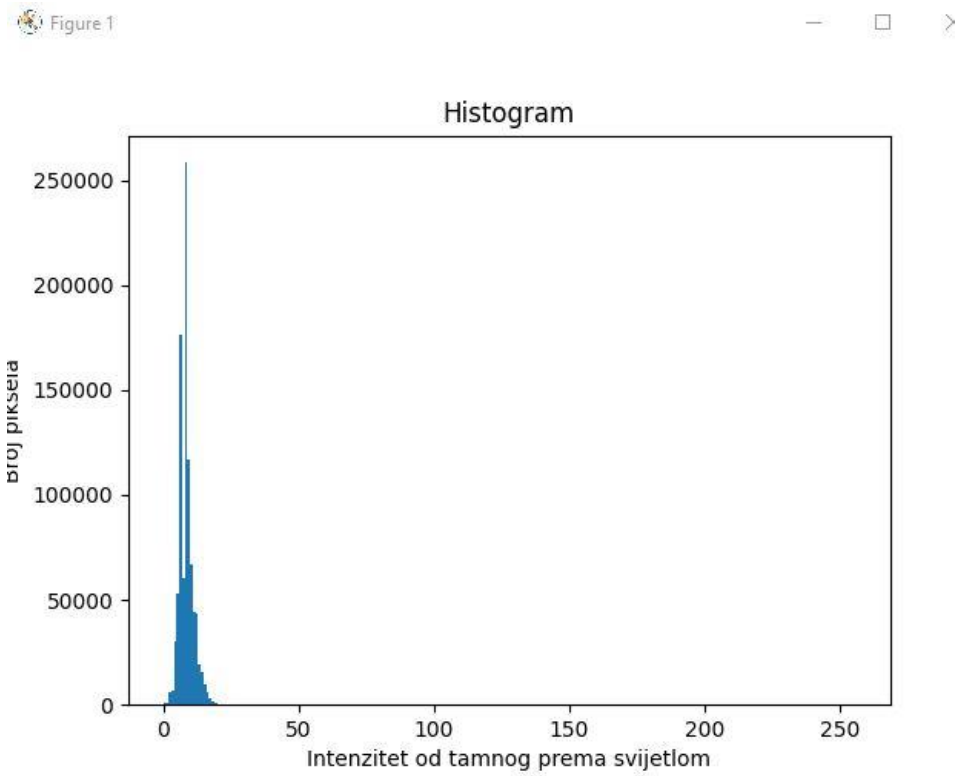
```
import cv2 as cv
from matplotlib import pyplot as plt

vid = cv.VideoCapture(0, cv.CAP_DSHOW)

while True:
    ret, img = vid.read()
    cv.imshow("img", img)
    plt.hist(img.ravel(), 256, [0, 256])
    plt.xlabel('Intenzitet od tamnog prema svijetlom')
    plt.ylabel('Broj piksela')
    plt.title('Histogram')
    plt.draw()
    plt.pause(0.1)
    plt.clf()

vid.release()
cv.destroyAllWindows()
```

Slika 4.15. Kod za histogram

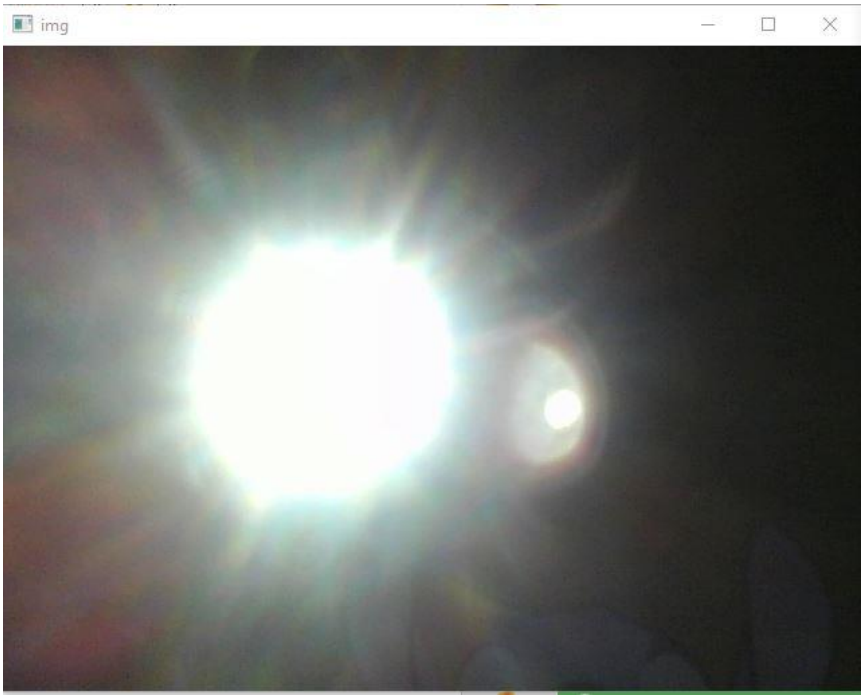
U primjeru se koristi *live* video sniman kamerom računala. Da bi mogli napraviti histogram potrebno je uključiti paket *matplotlib* koji se koristi za stvaranje statičkih, animiranih i interaktivnih vizualizacija u Pythonu. Slika 4.16. prikazuje prekrivenu kameru koja sadrži većinom crnu boju, pa se na histogramu može očitati da je broj piksela intenziteta 0-25 preko 250 000. S druge strane, kada na prema kameri uputimo svjetlo, odnosno kada je na slici dosta piksela svijetle boje, vidimo da na histogramu broj piksela intenziteta 250 raste preko 40 000. Slika 4.19. prikazuje histogram za takvu situaciju.



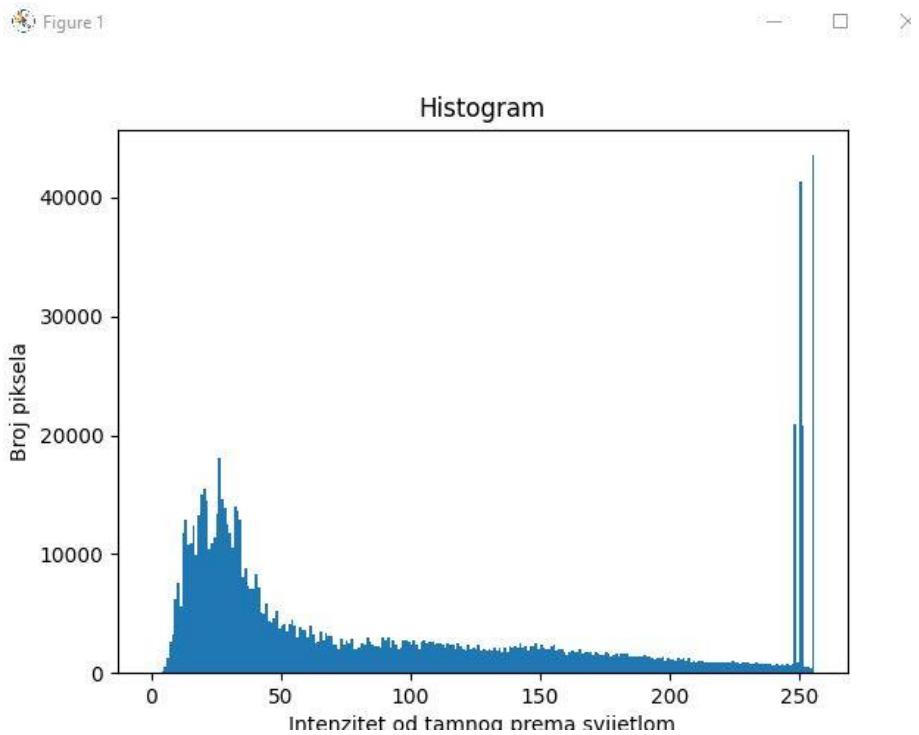
Slika 4.16. Prekrivena kamera



Slika 4.17. Histogram prekrivene kamere



Slika 4.18. Svjetlo na kameri

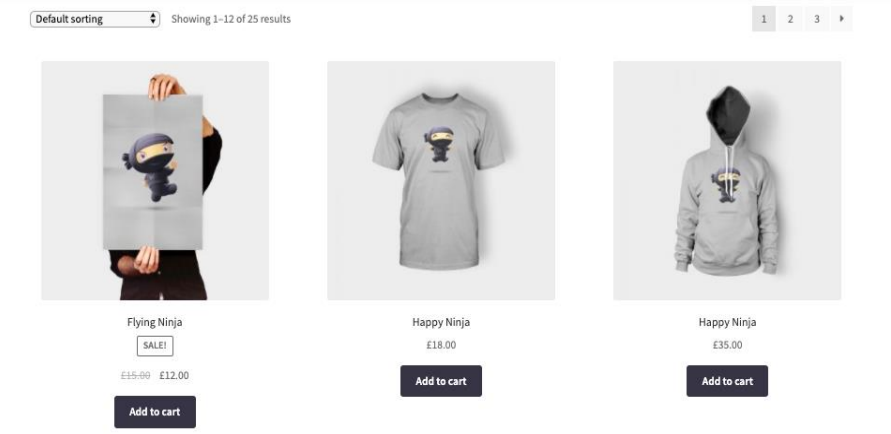


Slika 4.19. Histogram svjetla na kameri

4.7. Thumbnail

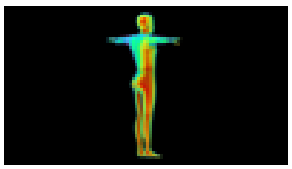
Thumbnail predstavlja inačicu slike ili video zapisa manje veličine. Koriste se za olakšavanje tijekom prepoznavanja i organiziranja velike količine slika ili video zapisa. *Thumbnail* omogućava smanjiti propusnost i vrijeme preuzimanja na web stranicama jer je on zapravo manja kopija izvorne slike. [20]

Osim što omogućava da posjetitelji web stranica odmah vide mnoštvo sadržaja bez povećanja vremena učitavanja stranice, omogućava i uštedu prostora, lakoću korištenja i interaktivnost. *Thumbnail* se pojavljuje svugdje i u nekim situacijama su od bitne važnosti, kao npr. na YouTube kanalima gdje su one jedan od načina kako privući gledatelja da pogleda naš video. Ista situacija je i kod internet trgovina gdje *thumbnail* predstavlja sliku nekog proizvoda i jako je bitno da *thumbnail* bude reprezentativan i oku privlačan kako bi privukli kupca da uđe u taj proizvod. Slika 4.20. prikazuje primjer *thumbnaila* u internet trgovini.



Slika 4.20. Primjer thumbnaila u internet trgovini

Slika 4.22. prikazuje kod za izdvajanje *thumbnaila* iz izvornog video zapisa. Potrebno je otvoriti video i podijeliti ga na okvire. Uzmemo jedan okvir kao *thumbnail*, u ovom primjeru okvir 5 te ga spremimo. Postavi se varijabla *MAX_SIZE* na određeno vrijednost (100,100) te se pomoću metode *.thumbnail()* pretvori okvir u *thumbnail*.



Slika 4.21. Thumbnail video zapisa

```
#Otvori video
video= cv2.VideoCapture('C:/vide.mp4')
i=0
frame=5
while(True):
    ret1,frame1 = video.read()
    if ret1:
        name1 = './thumbnail' + str(frame) + '.png'
        cv2.imwrite(name1,frame1)
    else:
        break

image = Image.open(r'C:/thumbnail' + str(frame) + '.png')
MAX_SIZE = (100, 100)
image.thumbnail(MAX_SIZE)

image.save('pythonthumb.png')
image.show()
```

Slika 4.22. Kod za thumbnail

5. VODENI ŽIGOVI

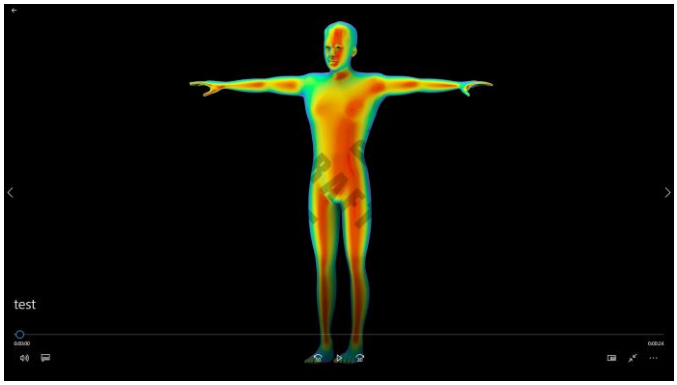
Vodeni žigovi (*engl. Watermarks*) služe za označavanje digitalne slike ili video zapisa, odnosno stavljanje pečata na njih. To se radi kako bi ih se zaštitilo od krađe i povrede autorskih prava. Koriste se za prikrivanje informacija tako da se kombiniraju dvije informacije od kojih se svaka može izdvojiti nezavisnim procesom. Vodeni žig može biti vidljiv ljudskom oko ili skriven.

Vodeni žigovi obuhvaćaju sljedeće procese:

- Stavljanje vodenog žiga u sliku/video
- Distribucija označenog dokumenta
- Izvlačenje vodenog žiga iz slike/videa
- Procjena valjanosti žiga

Vodeni žig može biti tekstualna poruka, manja slika, brojevi i slično. Nakon što se on sakrije u sliku/video i oni se distribuiraju prolaze kroz razne promjene kao što su kompresija, promjena veličine, kontrasta i sl. što može naštetiti valjanosti vodenog žiga. Poslije svih tih promjena potrebno je vodeni žig izvući iz slike, što u nekim situacijama zna biti teško.

Slika 5.1. Vodeni žig



Slika 5.2. Primjer vodenog žiga

Kako bi stavili vodeni žig preko videa potrebno je koristiti funkciju *CompositeVideoClip()* u kojoj spajamo video i vodeni žig. Slika 5.3. prikazuje kod za postavljanje vodenog žiga. Na isti način možemo staviti bilo koju drugu sliku kroz cijeli video ili samo pojedini isječak videa te na bilo koju poziciju videa.

```
video = mp.VideoFileClip("video.mp4")

watermark = (mp.ImageClip("watermark.png")
             .set_duration(video.duration)
             .set_pos(("center")))


final = mp.CompositeVideoClip([video, watermark])
final.write_videofile("test.mp4")
```

Slika 5.3. Kod za postavljanje vodenog žiga

Kako bi osigurali vodeni žig od uklanjanja sa slike ili videa potrebno je otežati taj proces napadaču. To možemo napraviti tako da vodeni žig ima sljedeće karakteristike:

- postavljen na mjesto koje je puno detalja,
- boja bude slična pozadini,
- zauzima što veću površinu slike
- složen s puno detalja
- proziran, odnosno neprimjetan

6. METAPODACI

Meta podaci (engl. *Metadata*) su svi dostupni podaci o digitalnom video zapisu, kao što su autor, datum stvaranja, mjesto snimanja, podaci o kameri i datumu prijenosa. Ovi podaci pomažu identificirati karakteristike datoteke te organizirati velike količine video zapisa. Tri su vrste meta podataka o video zapisu, a odnose se na opisne, strukturne i administrativne podatke.

Opisni meta podaci sadrže informacije o videozapisu kao što su:

- Jedinstveni identifikatori
- Fizički atributi (dimenzija, vrsta datoteke,..)
- Dodani atributi (naslov, opis,..)

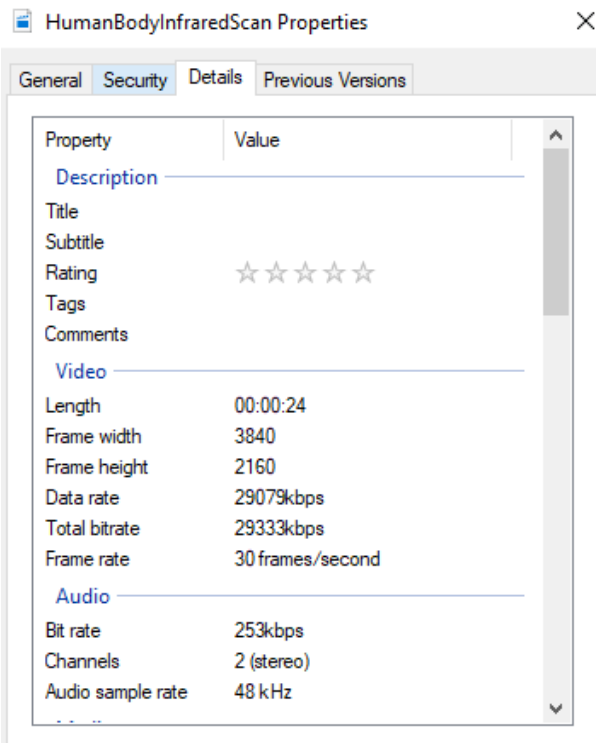
Strukturni meta podaci se odnose na organizaciju video zapisa, a uključuju odjeljke, video poglavlja, indekse i sadržaj.

Administrativni meta podaci sadrže prava i intelektualno vlasništvo, a dijele se u:

- Tehničke meta podatke (za dekodiranje i iscrtavanje datoteka)
- Meta podatke očuvanja (dugoročno upravljanje i arhiviranje)
- Meta podatke o pravima (intelektualno vlasništvo i prava korištenja)

Mijenjanjem digitalnog video zapisa mijenjaju se i njegovi meta podaci. Standard za meta podatke video zapisa je XMP (engl. *Extensible Metadata Platform*). On standardizira model podataka, format serije i osnovna svojstva za definiciju i obradu proširivih meta podataka. [21]

Slika 6.1. prikazuje dio meta podataka o video zapisu. Možemo vidjeti da je dužina video zapisa 24 sekunde, širina 3840 piksela, visina 2160 piksela, brzina mijenjanja okvira 30 fps-a i slične informacije.

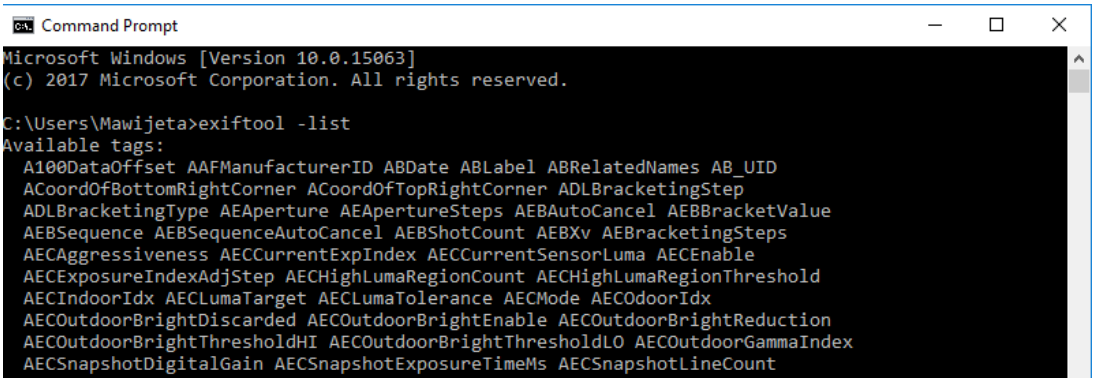


Slika 6.1. Primjer metapodataka

6.1. Pristup meta podacima pomoću ExifTool-a

Osim preko prethodno opisanog načina, meta podacima moguće je pristupiti i pomoću programa *ExifTool*. *ExifTool* je besplatni pomoćni program za čitanje, pisanje i rukovanje slikom, audio i video zapisom koji je objavljen 2003.godine. Njegov kreator je Phil Harvey, a koristi se kao alat za rad s meta podacima. Korisnicima daje veliki niz mogućnosti jer podržava široki raspon formata i vrsta datoteka meta podataka kao što su IPTC, XMP, JFIF, GeoTIFF, ICC profil i slično. [22]

ExifTool podržava skoro 23.000 oznaka, a kako bi pronašli sve oznake koristi se parametar *list.* Slika 6.2. prikazuje taj postupak, gdje vidimo cijeli niz oznaka.



Slika 6.2. Lista svih oznaka programa ExifTool

Da bi prikazali meta podatke iz određenog videa potrebno je samo napisati exiftool i naziv videa u *Command Promptu.* Slika 6.3. predstavlja taj postupak u kojem možemo vidjeti naziv videa, veličinu datoteke, datum i vrijeme kreiranja i slične podatke.

```
Microsoft Windows [Version 10.0.15063]
(c) 2017 Microsoft Corporation. All rights reserved.

C:\Users\Mawijeta\Videos\Movavi Library>exiftool video.mp4
ExifTool Version Number         : 12.30
File Name                       : video.mp4
Directory                       : .
File Size                       : 5.0 MiB
File Modification Date/Time     : 2021:08:30 22:10:33+02:00
File Access Date/Time           : 2021:08:30 22:10:42+02:00
File Creation Date/Time         : 2021:08:30 22:10:42+02:00
File Permissions                : -rw-rw-rw-
File Type                       : MP4
File Type Extension             : mp4
MIME Type                       : video/mp4
Major Brand                     : MP4 v2 [ISO 14496-14]
Minor Version                   : 0.0.0
Compatible Brands               : isom, mp42
Media Data Size                 : 5226574
```

Slika 6.3. Meta podaci videa

Osim pristupa svim podacima, moguće je pristupiti i samo nekim podacima. Ako želimo pristupiti samo jednoj oznaci, onda to radimo pomoću crtice (-) i naziva oznake. Na primjer, da bi ispisali samo vrstu formata, odnosno oznaku *FileType,* u *Command Prompt* ćemo napisati *exiftool -FileType* i naziv datoteke kojoj pristupamo. Slika 6.4. prikazuje takav primjer.

```
C:\Users\Mawijeta\Videos\Movavi Library>exiftool -FileType video.mp4
File Type                       : MP4
```

Slika 6.4. Ispis oznake FileType

Osim pristupa određenoj oznaci, moguće ju je i ukloniti iz liste oznaka o video zapisu. Ovaj način je sličan prethodnom, samo se doda još jedna crtica(-) uz prethodnu, odnosno potrebno je upisati *exiftool - -FileType* i naziv datoteke. Slika 6.5. predstavlja primjer gdje vidimo da u popisu meta podataka nema oznake *FileType* kao što je to bilo u prethodnim primjerima.

```
C:\Users\Mawijeta\Videos\Movavi Library>exiftool --FileType video.mp4
ExifTool Version Number         : 12.30
File Name                       : video.mp4
Directory                       : .
File Size                       : 5.0 MiB
File Modification Date/Time     : 2021:08:30 22:10:33+02:00
File Access Date/Time           : 2021:08:30 22:10:42+02:00
File Creation Date/Time         : 2021:08:30 22:10:42+02:00
File Permissions                : -rw-rw-rw-
File Type Extension             : mp4
MIME Type                       : video/mp4
Major Brand                     : MP4 v2 [ISO 14496-14]
Minor Version                   : 0.0.0
Compatible Brands               : isom, mp42
Media Data Size                 : 5226574
Media Data Offset               : 40
```

Slika 6.5. Uklanjanje oznake iz liste meta podataka

Osim pristupa meta podacima, pomoću programa ExifTool moguće je i promijeniti meta podatke. Slika 6.6. prikazuje jednostavni postupak, gdje pomoću naredbe *exiftool -FileName=NoviNazivVidea.mp4 video.mp4* mijenjamo naziv datoteke. Prikazano je i da se datoteci više ne može pristupiti sa starim nazivom.

```
C:\Users\Mawijeta\Videos\Movavi Library>exiftool -FileName=NoviNazivVidea.mp4 video.mp4
    1 image files updated

C:\Users\Mawijeta\Videos\Movavi Library>exiftool video.mp4
Error: File not found - video.mp4

C:\Users\Mawijeta\Videos\Movavi Library>exiftool NoviNazivVidea.mp4
ExifTool Version Number         : 12.30
File Name                       : NoviNazivVidea.mp4
Directory                       : .
File Size                       : 5.0 MiB
File Modification Date/Time     : 2021:08:30 22:10:33+02:00
File Access Date/Time           : 2021:08:30 22:10:42+02:00
File Creation Date/Time         : 2021:08:30 22:10:42+02:00
File Permissions                : -rw-rw-rw-
File Type                       : MP4
```

Slika 6.6. Promjena naziva videa pomoću programa ExifTool

S obzirom da je moguće mijenjati meta podatke, najzanimljivija je promjena GPS lokacije kako bi lažirali mjesto snimanja samog video zapisa. Na taj način možemo postaviti GPS koordinate tako da izgleda kao da je video zapis snimljen u gradu Splitu čije su koordinate 43.5116383 i 16.4399659. Slika 6.7. pokazuje taj postupak, gdje su primijenjene GPS koordinate i izlistani svi GPS podaci o video zapisu.

```
C:\Users\Mawijeta\Videos\Movavi Library>exiftool -GPSLatitude=43.5116383 -GPSLongitude=16.4399659
 NoviNazivVidea.mp4
    1 image files updated

C:\Users\Mawijeta\Videos\Movavi Library>exiftool -s -*GPS* NoviNazivVidea.mp4
GPSLatitude                     : 43 deg 30' 41.90" N
GPSLongitude                    : 16 deg 26' 23.88" E
GPSLatitudeRef                  : North
GPSLongitudeRef                 : East
GPSPosition                     : 43 deg 30' 41.90" N, 16 deg 26' 23.88" E
```

Slika 6.7. Promjena i prikaz GPS lokacije video zapisa

S druge strane, moguće je vidjeti je li video modificiran na neki način. Ako pristupimo videu koji nije mijenjan od trenutka kreiranja video zapisa, onda će njegove oznake *CreateDate* i *ModifyDate* biti jednake. U protivnom, ako je video modificiran na bilo koji način, onda će te oznake biti različite. Slika 6.8. prikazuje taj primjer, gdje se izvorni video skratio za jednu stotinku, što je rezultiralo promjenom oznake *ModifyDate*.

```
C:\Users\Mawijeta\Videos\Movavi Library>exiftool -CreateDate -ModifyDate IzvorniVideo.mp4
Create Date                     : 2021:08:30 16:53:25
Modify Date                     : 2021:08:30 16:53:25

C:\Users\Mawijeta\Videos\Movavi Library>exiftool -CreateDate -ModifyDate MijenjaniVideo.mp4
Create Date                     : 2021:08:30 16:53:25
Modify Date                     : 2021:08:30 20:09:11
```

Slika 6.8. Prikaz meta podataka videa koji je modificiran

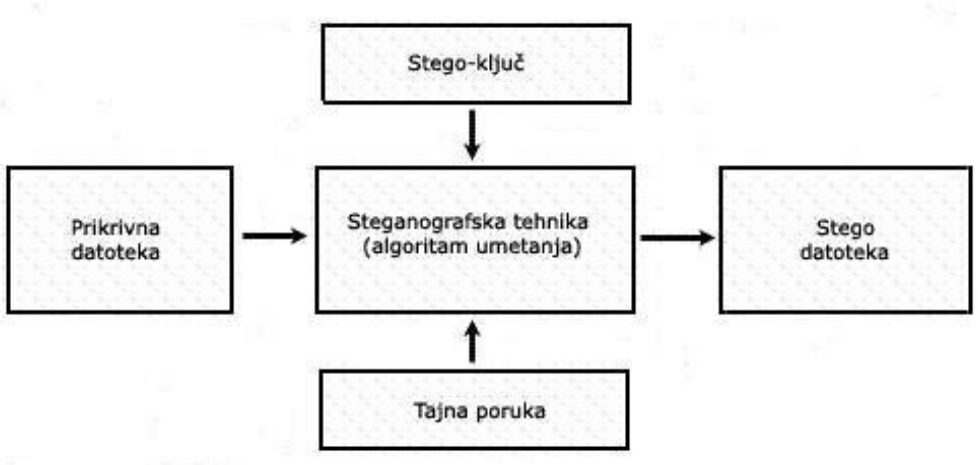
Osim navedenih mogućnosti, postoji još mnoštvo raznih naredbi pomoću kojih je moguće čitati, mijenjati i rukovati raznim meta podacima.

7. STEGANOGRAFIJA

Naziv steganografija dolazi od grčke riječi *steganos,* što u prijevodu znači skriveno i korijena riječi *graph,* što znači pisati. Ova tehnika se odnosi na skrivanje tajnih podataka unutar neke datoteke ili poruke kako se podaci ne bi otkrili. Nakon što se podaci prenesu, oni se iščitavaju iz datoteke u kojoj su skriveni. Moguće je koristiti ovu tehniku u kombinaciji s enkripcijom podataka kako bi se povećala zaštita podataka. [23]

Steganografija, kao princip skrivanja informacija, postoji već stoljećima. Jedan od najstarijih primjera dolazi iz 500.-te godine prije Krista kada je jedan vladar tetovirao poruku na glavi svoga roba i pustio njegovu kosu da raste. Nakon toga je rob otišao do zeta svog vladara koji je obrijao glavu i pročitao skrivenu poruku. Kroz godine oblici steganografije su se mijenjali, a danas su najzastupljeniji u digitalnom svijetu. [24]

Steganografija se koristi za prikrivanje digitalnog sadržaja kao što je tekst, slika, video ili audio zapis. Najvažnije je skriti informaciju na način da on ne izaziva sumnju. Osnovni princip je da pošiljatelj skrivene poruke odabere datoteku, sliku, zvuk ili tekst koju šalje primatelju. Nakon toga u odabranu datoteku stavlja skrivenu poruku jednom steganografksom metodom čime kreira stego datoteku. Slika 7.1. prikazuje sami princip steganografskog procesa. [25]



Slika 7.1. Princip steganografskog procesa

Najpopularnija tehnika je supstitucija bita najmanje važnosti (engl. *LSB substitution; least significant bit substitution*). Ova metoda temelji se na skrivanju podataka u najmanje značajnim bitovima datoteke. Na ovaj način moguće je sakriti veliku količinu podataka bez velikog

utjecaja na datoteku koja prenosi informaciju. S obzirom da se postupak provodi korištenjem binarnih zapisa, potrebno je pretvoriti vrijednosti piksela okvira u binarni zapis.

Tablica 7-1 Primjer pretvaranja vrijednosti piksela okvira u binarni zapis

0	202	45	12	63
8	2	14	26	28
140	5	155	26	13
85	69	21	36	255
255	15	75	255	30

00000000	11001010	00101101	00001100	00111111
00001000	00000010	00001110	00011010	00011100
10001100	00000101	10011011	00011010	00001101
01010101	01000101	00010101	00100100	11111111
11111111	00001111	01001011	11111111	00011110

Kao što je rečeno, postupak steganografije provodi se zamjenom najmanje značajnih bitova okvira u koji skrivamo s najznačajnijim bitovima okvira koji skrivamo. Tablica 7-2 prikazuje binarni zapis matrice piksela okvira u koji skrivamo drugi okvir gdje je crvenom bojom označen najmanje značajan bit svakog piksela. S druge strane, Tablica 7-3 predstavlja binarni zapis matrice piksela okvira koji se skriva u drugi okvir, a zelenom bojom označeni su najznačajniji bitovi piksela. Nakon što se zamjene bitovi označeni crvenom bojom s bitovima označenim zelenom bojom, dobije se matrica koju predstavlja Tablica 7-4, odnosno binarni zapis matrice piksela nakon provedbe postupka steganografije.

Tablica 7-2 Binarni zapis matrice piksela okvira u koji se skriva drugi okvir

00000000	11001010	00101101	00001100	00111111
00001000	00000010	00001110	00011010	00011100
10001100	00000101	10011011	00011010	00001101
01010101	01000101	00010101	00100100	11111111
11111111	00001111	01001011	11111111	00011110

Tablica 7-3 Binarni zapis matrice piksela okvira koji se skriva u drugi okvir

11111111	01001001	01001011	11110111	00011110
00000010	00000010	00001110	00000010	00011100
00110010	00000101	10011011	00011010	11001100
01010101	01000101	01001111	00100100	11111111
10001100	00000101	10011011	00011010	00001101

Tablica 7-4 Binarni zapis matrice piksela nakon provedbe postupka steganografije

00000011	11001001	00101101	00001111	00111100
00001000	00000000	00001100	00011000	00011100
10001100	00000100	10011010	00011000	00001111
01010101	01000101	00010101	00100100	11111111
11111110	00001100	01001010	11111100	00011100

Postupak kodiranja, odnosno skrivanja video zapisa u drugi video zapis prikazan je sljedećim kodovima. Slika 7.2. prikazuje kod za čitanje video zapisa koji skrivamo i video zapisa u koji skrivamo taj zapis. Također, potrebno je napraviti, u slučaju da ne postoje, datoteke *data1* i *data2* u koje se spremaju okviri iz tih video zapisa te datoteku *data3* u koju se unose okviri nakon provedbe postupka steganografije.

```
import cv2
import os
from PIL import Image
import numpy as np
import glob

cam1 = cv2.VideoCapture("video_u_koji_skrivamo.mp4")
cam2 = cv2.VideoCapture("video_koji_skrivamo.mp4")
try:
    if not os.path.exists('data1'):
        os.makedirs('data1')
    if not os.path.exists('data2'):
        os.makedirs('data2')
    if not os.path.exists('data3'):
        os.makedirs('data3')

except OSError:
    print ('Error: Creating directory of data')
```

Slika 7.2. Kod za čitanje video zapisa na kojima se izvodi postupak steganografije i stvaranje datoteka u koju se spremaju okviri tih videa

Slika 7.3. prikazuje petlju koja sprema okvire video zapisa u datoteke *data1* i *data2*. Okviri se spremaju pod nazivom trenutnog okvira u petlji. Nakon spremanja prvih okvira, brojač se povećava za 1 i prolazi se kroz petlju dok se ne prođu svi okviri video zapisa.

```
currentframe = 0
while(True):

    ret1,frame1 = cam1.read()
    ret2,frame2 = cam2.read()

    if ret1 & ret2:
        name1 = './data1/frame' + str(currentframe) + '.png'
        cv2.imwrite(name1,frame1)

        name2 = './data2/frame' + str(currentframe) + '.png'
        cv2.imwrite(name2,frame2)
        currentframe += 1
    else:
        break
```

Slika 7.3. Kod za spremanje okvira u datoteke

Slika 7.4. prikazuje dio koda za *for* petlju gdje se učitavaju svi okviri koji su prethodno spremljeni. Prethodno je potrebno definirati ukupan broj okvira naredbom *cv2.CAP_PROP_FRAME_COUNT* . U varijable *matrica_sa_pikselima_1* i *matrica_sa_pikselima_2* se spremaju matrice okvira koje će se kasnije koristiti u postupku steganografije.

```
total = int(cam1.get(cv2.CAP_PROP_FRAME_COUNT))
print(total)
img=[]

for x in range (0,total):

        slika_1 = Image.open('./data1/frame' + str(x) + '.png')

        slika_2 = Image.open('./data2/frame' + str(x) + '.png')

        stupci, retci = slika_1.size

        matrica_sa_pikselima_1 = slika_1.load()
        matrica_sa_pikselima_2 = slika_2.load()
```

Slika 7.4. Kod za učitavanje okvira koji su prethodno spremljeni

```
for i in range (0, stupci):
    for j in range (0, retci):
            r = matrica_sa_pikselima_1[i,j][0]
            g = matrica_sa_pikselima_1[i,j][1]
            b = matrica_sa_pikselima_1[i,j][2]

            r_binary = format(r, '#010b').replace("0b", "")
            g_binary = format(g, '#010b').replace("0b", "")
            b_binary = format(b, '#010b').replace("0b", "")

            r2 = matrica_sa_pikselima_2[i,j][0]
            g2 = matrica_sa_pikselima_2[i,j][1]
            b2 = matrica_sa_pikselima_2[i,j][2]

            r_binary2 = format(r2, '#010b').replace("0b", "")
            g_binary2 = format(g2, '#010b').replace("0b", "")
            b_binary2 = format(b2, '#010b').replace("0b", "")
```

Slika 7.5. Kod za pretvaranje matrice s pikselima u binarni oblik

Kao što je ranije navedeno, potrebno je pretvoriti piksele u binarni oblik. Za svaki okvir se pojedinačno traži matrica s pikselima te se te vrijednosti pretvaraju u binarni oblik kako bi se kasnije u postupku steganografije zamijenili najmanje značajni bitovi okvira u koji skrivamo s najznačajnijim bitovima okvira je potrebno skriti.

```
novi_r_binary = [0, 0, 0, 0, 0, 0, 0, 0]
novi_r_binary[0] = int(r_binary[0])
novi_r_binary[1] = int(r_binary[1])
novi_r_binary[2] = int(r_binary[2])
novi_r_binary[3] = int(r_binary[3])
novi_r_binary[4] = int(r_binary2[0])
novi_r_binary[5] = int(r_binary2[1])
novi_r_binary[6] = int(r_binary2[2])
novi_r_binary[7] = int(r_binary2[3])

novi_r_int = int("".join(map(str, novi_r_binary)),2)

novi_g_binary = [0, 0, 0, 0, 0, 0, 0, 0]
novi_g_binary[0] = int(g_binary[0])
novi_g_binary[1] = int(g_binary[1])
novi_g_binary[2] = int(g_binary[2])
novi_g_binary[3] = int(g_binary[3])
novi_g_binary[4] = int(g_binary2[0])
novi_g_binary[5] = int(g_binary2[1])
novi_g_binary[6] = int(g_binary2[2])
novi_g_binary[7] = int(g_binary2[3])

novi_g_int = int("".join(map(str, novi_g_binary)),2)
```

```
novi_b_binary = [0, 0, 0, 0, 0, 0, 0, 0]
novi_b_binary[0] = int(b_binary[0])
novi_b_binary[1] = int(b_binary[1])
novi_b_binary[2] = int(b_binary[2])
novi_b_binary[3] = int(b_binary[3])
novi_b_binary[4] = int(b_binary2[0])
novi_b_binary[5] = int(b_binary2[1])
novi_b_binary[6] = int(b_binary2[2])
novi_b_binary[7] = int(b_binary2[3])

novi_b_int = int("".join(map(str, novi_b_binary)),2)
matrica_sa_pikselima_1[i,j] = (novi_r_int, novi_g_int, novi_b_int)
```

Slika 7.6. Kod za steganografiju

Nakon steganografije potrebno je spremiti okvire u datoteku kako bi se mogli ponovo pretvoriti u video zapis. U ovom postupku važno je spremiti okvire jer se okviri u protivnom neće sortirati prema pravom redoslijedu. Razlog tome je jer je naziv okvira tipa *string*, što znači da će se sortiranje provesti po principu 1, 10 ,100, 2, 20 i tako dalje.

```
if (x<=9):
    slika_1.save('./data3/frame' + '00'+str(x) + '.png')
elif(x>=10 & x<=99):
    slika_1.save('./data3/frame' + '0'+str(x) + '.png')

img.append(slika_1)
```

Slika 7.7. Spremanje okvira nakon provedbe steganografije

Nakon što se napravio cijeli proces potrebno je samo još spremljene okvire ponovo pretvoriti u video zapis. Slika 7.8. prikazuje kod gdje je definirana funkcija za pretvaranje okvira u video zapis.

Postupak dekodiranja, odnosno dobivanja video zapisa nakon postupka steganografije je obrnut prikazanom postupku.

```
from os.path import isfile, join
def convert_frames_to_video(pathIn,pathOut,fps):
    frame_array = []
    files = [f for f in os.listdir(pathIn) if isfile(join(pathIn, f))]
    files.sort(key = lambda x: x[5])
    files.sort()

    for i in range(len(files)):
        filename=pathIn + files[i]
        img = cv2.imread(filename)
        height, width, layers = img.shape
        size = (width,height)
        print(filename)
        frame_array.append(img)
    out = cv2.VideoWriter(pathOut,cv2.VideoWriter_fourcc(*'DIVX'), fps, size)
    for i in range(len(frame_array)):
        out.write(frame_array[i])
    out.release()

pathIn= './data3/'
pathOut = 'video1.avi'
fps = 30.0
convert_frames_to_video(pathIn, pathOut, fps)
```

Slika 7.8. Kod za pretvaranje okvira u video zapis

8. ZAKLJUČAK

S obzirom da se danas tehnologija razvija velikom brzinom, tako se i mogućnosti snimanja, editiranja i analiziranja video zapisa povećavaju. Razni mobilni uređaji i osobna računala sadržavaju mnoštvo podataka, odnosno dokaza o digitalnim video zapisima. Upravo pomoću njih mogu se saznati razne informacije vezane za multimedijske zapise.

Većina ljudi upoznata je sa snimanjem i editiranjem video zapisa, ali s druge strane postoji široka pozadina u kojoj su prikazani razni podaci o tim datotekama. Forenzičkom analizom tih podataka, u mogućnosti smo spremati i obrađivati video zapise te mijenjati njihove meta podatke. Osim izravnog pristupa njima, postoje i razne aplikacije i alati koje nam olakšavaju taj pristup da bi na jednostavan način mogli baratati s velikim brojem video zapisa.

Može se zaključiti da je video zapis kompleksan oblik multimedije na kojem je moguće provesti razne metode kako bi ga spremili, sortirali, editirali ili čak koristili kao kanal za tajno komuniciranje. Daje mnoštvo informacija korisniku i današnja digitalna tehnologija bila bi znatno siromašnija bez njegove uporabe. Zahvaljujući velikom razvoju i tehnološkom napretku, moguće je bez problema krivotvoriti, odnosno lažirati podatke pa tako i video zapise. Zbog svih tih mogućnosti, forenzika je uvelike uključena u analiziranje video zapisa te je primorana stalno pratiti trendove razvoja mobilnih uređaja i osobnih računala kako bi onemogućila plasiranje lažnih informacija u javnost.

LITERATURA

[1] T. HICKMORE, s Interneta https://www.nicemedia.co.uk/history-video-first-things-first/, 14.07.2021.

[2] M. Pavlak, s Interneta, http://pvprm.zesoi.fer.hr/1999-00-web/1999-00-seminari/mpavlek/seminar/analog_video.htm., 14.07.2021.

[3] »Wikipedija,« , s Interneta, https://hr2.wiki/wiki/Video#Analog_video, 14.07.2021.

[4] »technopedia.com,«, s Interneta, https://www.techopedia.com/definition/5505/digital-video-dv, 15.07.2021.

[5] Zovko-Cihlar, B.; Grgić, S.; Grgić, M.; "STANDARDNI POSTUPCI ZA KODIRANJE VIDEOSIGNALA", Sveučilište u Zagrebu, Fakultet elektrotehnike i računarstva, Zagreb

[6] Dujmić, H., "Multimedijski sustavi", Fakultet elektrotehinke, strojarstva i brodogradnje u Splitu, Split

[7] »Wikipedia,«, s Interneta, https://en.wikipedia.org/wiki/H.263, 21.07.2021.

[8] »Wikipedia,« s Interneta, https://en.wikipedia.org/wiki/Moving_Picture_Experts_Group#:~:text=The%20Moving%20Picture%20Experts%20Group,file%20formats%20for%20various%20applications.&text=MPEG%20formats%20are%20used%20in%20various%20multimedia%20systems., 22.07.2021.

[9] »Wikipedia,« s Interneta, https://en.wikipedia.org/wiki/MPEG-1, 22.07.2021.

[10] »Wikipedia,«s Interneta, https://en.wikipedia.org/wiki/MPEG-2, 30.07.2021.

[11] »Wikipedia,«s Interneta, https://en.wikipedia.org/wiki/MPEG-4, 30.07.2021.

[12] Wikipedia. s Interneta, https://en.wikipedia.org/wiki/Advanced_Video_Coding#Features, 25.07.2021.

[13] »Samsung.com,« s Interneta, https://www.samsung.com/in/support/tv-audio-video/what-is-avi-format/,01.08.2021.

[14] »Wikipedia,« s Interneta, https://en.wikipedia.org/wiki/Audio_Video_Interleave#Metadata, 01.08.2021.

[15] »fileinfo,«s Interneta, https://fileinfo.com/extension/mov, 01.08.2021.

[16] Wikipedia.s Interneta, https://en.wikipedia.org/wiki/MPEG-4_Part_14, 30.08.2021.

[17] »Techopedia,«s Interneta, https://www.techopedia.com/definition/10713/mpeg-4-part-14-mp4-multimedia-format, 03.08.2021.

[18] B. Doug, »Techsmith,«s Interneta, https://www.techsmith.com/blog/frame-rate-beginners-guide, 02.08.2021.

[19] Braović, M.; Krstinić, D., "FORENZICKA ANALIZA DIGITALNE SLIKE", Sveučilište u Splitu, Fakultet elektrotehnike, računarstva i brodogradnje, Split

[20] s Interneta, https://hr.awordmerchant.com/histograma, 02.08.2021.

[21] »Wikipedia,«s Interneta, https://en.wikipedia.org/wiki/Thumbnail, 10.08.2021.

[22] »vidispine,«s Interneta, https://www.vidispine.com/video-metadata-management. 12.08.2021.

[23] B. Chris, s Interneta, https://adamtheautomator.com/exiftool/#What_is_exiftool, 30.08.2021.

[24] s Interneta, https://searchsecurity.techtarget.com/definition/steganography, 15.08.2021.

[25] D. Ben, s Interneta, https://portswigger.net/daily-swig/what-is-steganography-a-complete-guide-to-the-ancient-art-of-concealing-messages, 15.08.2021.

[26] Z. Sanja, s Interneta, http://e.math.hr/old/stegano/index.html, 15.08.2021.

POPIS SLIKA

POPIS TABLICA

POPIS KRATICA

AVC	Advanced Video Coding
AVI	Audio Video Interleave
CIE	International Commission on Illumination
CIF	Common Intermediate Format
DCT	Discrete cosine transform
DVD	Digital Video Disc
engl	engleski
FPS	Frames per second
GOB	Group of blocks
IDCT	Inverse discrete cosine transform
IPTC	International Press Telecommunications Council
ISDN	Integrated Services Digital Network
ISO	International Organization for Stardardization
ITU	Iternational Telecommunication Union
JFIF	JPEG File Interchange Format
JPEG	Joint Photographic Experts Group
LSB	Least significant bit
MB	Macro Block
MPEG	Moving Picture Experts Group
NTSC	National Television Standards Committee
PAL	Phase Alternating Line
QCIF	Quarter Common Intermediate Format
QTFF	QuicskTime File Format
RGB	Red, green, blue
RIFF	Resource Interchange File Format

SECAM	Sequential Color and Memory
VLC	Variable Length Coding
VO	Video object
VOP	Video object plane
XMP	Extensible Metadata Platform

SAŽETAK

Digitalni video zapis se snima i pohranjuje u digitalnom formatu kao jedinice i nule, za razliku od analognog koji koristi niz fotografija. Tehnološkim napretkom razvili su se razni standardi i formati video zapisa. Razvoj standarda kompresije video zapisa i tehnika koje su se koristile pri standardizaciji dolaze od dviju velikih međunarodnih organizacija ITU-T i MPEG. Video zapis je najkompleksniji oblik multimedije, a osim slike, zvuka, grafike i ostalih sadržaja, u sebi sadrži i meta podatke koji ga opisuju. Ti meta podaci mogu se na razne načine ekstrahirati iz video zapisa i na temelju njih možemo saznati mnoštvo informacija, od naziva datoteke, vremena snimanja, formata, do GPS oznaka gdje je video snimljen. Kako bi onemogućili zloupotrebu video zapisa, potrebno ga je zaštiti vodenim žigom. Na tržištu postoje razne aplikacije i alati pomoću kojih je moguće analizirati meta podatke. Forenzičkom analizom tih podataka, u mogućnosti smo spremati i obrađivati video zapise te mijenjati njihove meta podatke. Osim toga, moguće je koristiti video zapis kao kanal za tajnu komunikaciju na način da se neki oblik medija skrije u video zapis. Ovaj postupak naziva se steganografija, a koristi za prikrivanje digitalnog sadržaja kao što je tekst, slika, video ili audio zapis. Najvažnije je skriti informaciju na način da on ne izaziva sumnju.

KLJUČNE RIJEČI: digitalni video zapis; forenzička analiza; meta podaci; ekstrakcija podataka; steganografija; vodeni žig

SUMMARY

Digital video is recorded and stored in digital format as ones and zeros, unlike analog video which uses a series of photos. Technological advances have developed various standards and video formats. The development of video compression standards and the techniques used in standardization come from two major international organizations, ITU-T and MPEG. Video is the most complex form of multimedia and in addition to image, sound, graphics and other content, it also contains metadata that describes it. This metadata can be extracted from videos in a variety of ways and can be used to learn a lot of information, from file name, recording time, format, to GPS tags where the video was recorded. In order to prevent the misuse of the video, it is necessary to protect it with a watermark. There are various applications and tools on the market that can be used to analyze metadata. By forensic analysis of this data, we are able to store and process videos and change their metadata. In addition, it is possible to use video as a channel for secret communication by hiding some form of media in the video. This procedure is called steganography, and is used to disguise digital content such as text, images, video, or audio. The most important thing is to hide the information in such a way that it does not cause suspicion.

KEY WORDS: digital video; forensic analysis; metadata; data extraction; steganography; watermark