Univerzitet u Kragujevcu

Fakultet inženjerskih nauka

Naziv studijskog programa: Mašinsko inženjerstvo

Nivo studija: Osnovne akademske studije

Modul: Informatika u inženjerstvu

Predmet: Konačni elementi I

Broj indeksa: 144/2009

Andreja D. Radovanović

GENERISANJE 3D MODELA KOSTI NA OSNOVU 2D DICOM SNIMAKA

završni rad

Komisija za pregled i odbranu:

1. Dr Miroslav Živković - Mentor
2. Dr Radovan Slavković
3. Dr Vesna Ranković

Datum odbrane:06.07.2012.

Ocena:_____

U okviru ovog završnog rada, kandidat treba da izradi plugin za ImageJ. ImageJ predstavlja besplatan softver za obradu i analizu slika otvorenog koda. Plugin treba da omogući učitavanje DICOM medicinskih snimaka, nakon toga izvrši 3D rekonstrukciju i generisanje 3D modela kosti, i na kraju da omoguči snimanje dobijenog modela u formatu koji će biti kompatibilan sa ostalim poznatim softverima za 3D štampanje i obradu 3D modela. Plugin mora da sadrži tri načina prikazivanja 3D modela i to kao zapreminski prikaz, površinski i ortogonalni prikaz. Izradjeni proizvoljni interfejs bi trebao da sadrži izmedju ostalog osnovne opcije za manipulaciju prikaza 3D modela.

Od dobijenog 3D modela se očekuje da izgleda realno, i da se sa minimalnom obradom u nekom od softvera pošalje na 3D štampanje kako bi se kasnije mogao ugraditi u telo pacijenta.

Kragujevac, datum

mentor:

Dr Miroslav Živković

Rezime rada:

U ovom radu je opisan i predstavljen softver čiji je zadatak da se na osnovu 2D DICOM medicinskih snimaka, dobijenih CT skenerom, izvrši 3D rekonstrukcija, radi dobijanja 3D modela kostiju. Takodje su opisane sve metode i načini dobijanja 3D modela, kao i tehnologija za implementiranje dobijenog 3D modela u neki od 3D sofvera. Kao završna faza kompletnog procesa jeste 3D štampa i implementiranje odštampanog 3D modela kosti u telo obolelog pacijenta.

Ključne reči:

DICOM format, CT skener, ImageJ, 3D rekonstrukcija

Summary of work:

This work describes the software and presented with the task to be based on 2D DICOM medical images obtained by the CT scanner, to carry out 3D reconstruction, in order to obtain 3D models of bones. It also describes all the methods and procedures for the acquisition of 3D models, and technologies to implement the obtained 3D models of some of 3D software. The final phase is the 3D printing process and implementation of 3D printed models of the bones in the body of the sick patient.

Key words:

DICOM format, CT scanner, ImageJ, 3D reconstruction

Veliku zahvalnost dugujem svojoj porodici na svoj podršci prikom školovanja. Takodje zahvaljujem se svom mentoru Dr Miroslavu Živkoviću na podršci i savetima. Veliku zahvalnost dugujem Dr Nenadu Grujoviću koji je takodje ispratio ceo moj završni rad od početka i pružio mi izuzetno važne i korisne savete prikom izrade rada.

Andreja Radovanović

SADRŽAJ

1. UVOD

U vremenu drastičnih promena, budućnost nasleđuju oni koji uče. Oni koji misle da su sve naučili osposobljeni su da žive u svetu koji više ne postoji.

Eric Hoffer

Zahvaljujući veoma brzom napretku tehnologije, i razvoju računara, oblast njihove primene se znatno povećala. Nije moguće navesti oblast gde se sve informatičke tehnologije mogu primeniti, ali zasigurno, jedno od najbitnijih jeste primena računara u medicini.

Sam BIOINŽENJERING predstavlja primenu inženjerskih principa i metodologije dizajna za rešavanje medicinskih problema. Bioinženjering je podstakao razvoj novih tehnologija, proširio oblast primene unutar medicine, dao ideju za razvoj softvera, a ujedno pomogao pri rešavanju velikih problema današnjice, što je pre samo 10-ak godina nije bilo ni zamislivo.

Problemi sa kostima su veoma zastupljeni kod ljudske populacije, a za neke vidove bolesti, leka nema, pa jedino sto preostaje jeste da se kost zameni novom. Zamenjena kost, mora da odgovara pacijentu na više načina, počevši od oblika, dimenzija, težine itd.

Zaključeno je da se ne svodi sve na softver koji će generisati 3D model kostiju. Veliku ulogu pri izradi ima i oblast iz mašinstva(CAD/CAM), medicine i dr. Kao završna faza izrade jeste štampanje,i mplementiranje kosti u telo pacijenta. Veoma je bitno poznavanje i određivanje pravog materijala od kojeg će kost biti izradjena, pa kao odgovor na to rešenje nalazimo u biomaterijalima, koji se mogu definisati kao materijali koji su tako dizajnirani da nakon implementacije u ljudski organizam zamene žive delove organizma. Poznavanjem metoda, načina izrade i dr, moguće je pristupiti izradi softvera.

Zadatak softvera, koji će analizirati slike CT skenera pacijenata, i potom generisati 3D modele kostiju, se može pokazati kao izuzetno inovativan, a samim tim i veoma koristan. Softver je veoma zahtevan, jer uključuje veliki broj faktora, pa je veoma bitno da 3D model, dobijen na osnovu slika skenera, bude verno predstavljen, da bi se kasnije mogao odštampati, i implementirati u telo pacijenta.

Ovaj softver omogućava verni prikaz 3D modela kosti, mogućnost rekonstrukcije, raznih podešavanja itd. Snimak tela dobijen sa skenera, ubacujemo kao niz slika u softver, zatim sledi 3D analiza, snimanje fajla(izlazne fajlove je moguće naknadno doradjivati, ili implementirati u razne 3D softvere), i nakon toga štampanje na 3D štampaču.

2. JAVA PROGRAMSKI JEZIK

2. 1 Uvod u JAVA

Java (engl. Java, izgovor: java, ∗džava) je objektno-orjentisani programski jezik, koji je razvila kompanija **Sun Microsystems** početkom devedesetih godina.

U Javu su uvedeni paketi koji se oslanjaju na fajl sistem i koristi formalno koncept klasa iz objektno-orijentisane paradigme. Osim toga, jezik ima sintaksu sličnu jezicima C i C++, ali je mnogo strožiji pri prevođenju, dizajniran tako da bude nezavisan od platforme, i sa pojednostavljenim upravljanjem memorijom. Pretpostavlja se da je ovo urađeno zbog popularnosti jezika C, ali i zbog jednostavnosti nekih struktura. Prva verzija je zvanično objavljena 1995. godine.

Bilo je četiri osnovna cilja pred razvitak Java programskog jezika, koji su naravno do danas ispunjeni:

1. Da bude jednostavan i objektivno-orijentisan programski jezik
2. Da bude bezbedan
3. Da bude neutralan i portabilan kod svih operativnih sistema i platformi
4. Program da se izvršava sa izuzetno visokim perfomnasama

Jedna od karakteristika Java programskog jezika jeste prenosivost, što znači da računarski programi napisani u Javi, radi na isti način na bilo kom hardveru, operativnom sistemu, platformi. . .

Java je objektno-orjentisani programski jezik te kao takav poštuje pravilo da se jedna klasa nalazi u jednom fajlu (osim unutrašnjih klasa). Izvorni kod se čuva u fajlovima sa nastavkom . java. Programi napisani u programskom jeziku Java se ne kompajliraju u mašinski kod, već se prevode u bajt-kod, i tako prevedeni fajlovi imaju nastavak class. Da bi se izvršio program napisan u Javi, neophodno je imati Java Virtuelnu Mašinu, na kojoj se interpretira bajt-kod. Upravo se korišćenjem Java Virtuelne Mašine postiže nezavisnost od platforme, tako da se isti bajt-kod može jednako izvršavati na svakom operativnom sistemu na kome je instalirana Java Virtuelna Mašina.

2. 2. JAVA karakteristike

Verovatno najvažnija Javina karakteristika jeste činjenica da je dizajnirana tako da u startu bude mašinski nezavisna. Java programi mogu se neizmenjeni koristiti na svakoj platformi koja podržava ovaj programski jezik. Naravno, manji problemi su uvek moguci buduci da u krajnjoj liniji uvek zavisite od nacina na koji je Java implementirana na datom racunaru. Medutim, Java programi su, svakako, suštinski u mnogo vecoj meri prenosivi od programa pisanih u drugim programskim jezicima. Aplikacija napisana u Javi zahtevaće samo jedan skup iskaza izvornog koda, bez obzira na broj razlicitih platformi na kojima ce se koristiti. U svakom drugom programskom jeziku vrlo često je neophodno da se izvorni kôd aplikacije prilagodi konkretnom racunarskom okruženju, narocito ako ima zahtevan graficki korisnički interfejs.

Prema tome, Java omogućava značajne uštede i vremena i resursa kada su u pitanju razvoj, podrška i održavanje osnovnih aplikacija na vecem broju razlicitih softverskih i hardverskih platform.

Sledeću veoma važnu karakteristiku Jave predstavlja njena **objektna orijentisanost**. Objektno orijentisan pristup programiranju takode predstavlja suštinsku karakteristiku svih Java programa, tako da ćemo u nastavku ovog poglavlja razmatrati i šta to konkretno znači. Objektno orijentisani program u razumljiviji i za njihovo održavanje i proširivanje potrebno je manje vremena nego za programe koji su pisani bez korišcenja objekata.

2.3. Pisanje aplikacija u javi

Mnogo ljudi misli (čak i autor ovog teksta do prije par mjeseci) da *Javu* koristimo samo kao dodatak vlastitim *Web HTML* stranicama tako da rade interesantne efekte sa slikama ili računaju koliko ste dana ili minuta stari. Međutim, *Javu* možemo koristiti isto kao *C++* za pisanje *stand-alone* aplikacija. Postupak je sljedeći:

1. prevodimo Java izvorni kod u tzv. *bajtni kod*
2. izvršavamo *bajtni kod* interpretiranjem unutar *JVM* (*Javine Virtualne Mašine*)

Vidimo da je po strukturi implementacije programskog jezika *Java* negde na prelazu između dosad kristalno jasnih definicija pojmova *kompajler* i *interpreter*. Ona spada u obe klasifikacije dok uistinu nije potpuno niti jedna od njih.

Sâm *bajtni kod* je dosta manji od ekvivalentnog izvršnog koda, recimo, *C*-a, ali brzina njegovog interpretiranja daleko zaostaje naspram brzine izvršavanja ekvivalentnih programa pisanih u *C*-u. Alternativno rešenje je u korištenju tzv. *just-in-time* interpretiranja, gde *Javina Virtuelna Mašina* prebacuje *bajtni kod* u *native kod* prije samog izvršavanja. To za manje aplikacije pruža razumnu performansu uz zadržanu prenosivost izvršnog koda.

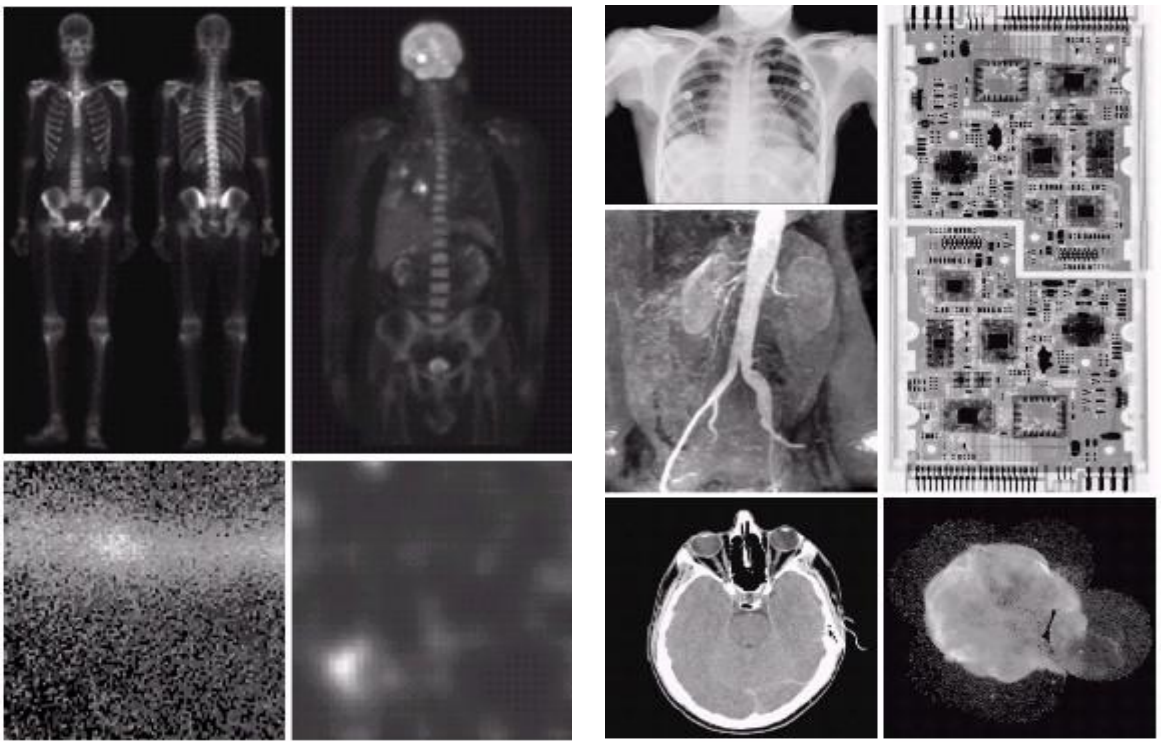
Međutim, čak i toliko naglašavana prenosivost možda ne počiva na čvrstim temeljima. Iole veće aplikacije će verovatno usled ograničenja nametnutih standardnim *Java* okruženjem biti prisiljene koristiti vlastita korisnička sučelja u vidu *native library* datoteka (*JNI*) čijim se funkcijama proširuju mogućnosti *Jave*. U tom slučaju mada je osnovna aplikacija prenosiva, moraće se prilagođavati od operativnog sistema do različitih arhitektura računara *library* datoteke pisane u *C*/*C++*-u bez kojih osnovna aplikacija neće moći raditi.

3. POJAM I OBRADA DIGITALNIH SLIKA

3.1. Pojam digitane slike

Digitalna slika je prikaz dvodimenzionalne slike sa konačnim skupom vrednosti koje se nazivaju pikseli.

Pikseli su pohranjeni u računarskoj memoriji ili hard disku kao rasterska slika ili rasterska mapa, odnosno dvodimenzionalni niz malih celina. Te vrednosti se uglavnom prenose u složenoj formi, kao što je JPEG format. Digitalne slike se mogu napraviti pomoću digitalnog fotoaparata, skenera, ultrazvuka itd [2].



Slika 1. *Primeri slika formiranih pomoću a)gama zraka(slika levo) i b) iks zraka (slika desno)*[2]

Površina slike je podeljena linijama po horizontali i vertikali u mrežu kvadratića – piksela. Memorija koju zauzima slika zavisi od broja piksela i broja boja koje su na raspolaganju. Broj piksela predstavlja rezoluciju i kvalitet slike je bolji što je rezolucija veća. Rezolucija se izražava brojem tačaka po inču.

Svaki piksel se u memoriji čuva posebno i pridružuje mu se jedan, 2, 3 ili 4 bajta, u zavisnosti od toga sa koliko se boja radi.

Ako je pikselu pridružen:

1 bajt → 2^8 = 256 boja
2 bajt → 2^{16} = 65. 536 boja
3 bajt → 2^{24} = 16, 7 miliona boja
4 bajt → 2^{32} = 4, 3 milijarde boja (true color)

3.1.1. Formati digitalnih slika

RAW format zapisa fotografije je format u kojem fotoaparat beleži sve podatke koje je zabeležio i sam digitalni senzor fotoaparata (RAW znači sirov, neobrađen). Kada vam je potreban najveći mogući kvalitet koji može zabeležiti vaš fotoaparat, snimajte u RAW formatu. Nema kompresije pa ni redukcije kvaliteta slike.

BMP – Svaki piksel se čuva posebno sa odgovarajućim brojem bajtova. Nema kompresije, nema gubitka podataka.

GIF (Graphic Interchange Format) –niz istih piksela se čuva kao jedan simbol pomnožen sa brojem ponavljanja. Ovaj format je ograničen na paletu od 256 boja.

JPG ili JPEG (Joint Photographers Experts Group) – spada u grupu kompresija sa gubitkom. Zasniva se na osobini ljudskog oka da bolje uočava površine i oblike nego varijacije u boji i osvetljenju. Slika se čuva u crno beloj verziji i delom informacija o boji. Formiraju se blokovi od 8×8 piksela i izračunava se prosečna vrednost osvetljenja i boje za celu grupu i ova informacija se pamti.

MPEG (Motion Pictures Experts Group) – oblik kompresije dizajniran za komprimovanje pokretnog videa. Baziran je na JPEG-u, pri čemu se memoriše samorazlika između slika koje se ponavljaju.

PNG format je format bez gubitka, što znači da koji god podatak postoji u orginalnoj datoteci postajaće i kada se slika dekodira. Nije ograničen na 256 boja. Zauzima nekoliko puta više prostora od JPEG-a.

TIFF format Ranije je imao veliku primjenu. Primjenjuje isti algoritam kompresije bez gubitka kao i gif. Koristio se za čuvanje skeniranih slika. Nema gubitka prilikom kompresije.

3.1.2. Načini predstavljanja digitalnih slika

Kod **vektorskog načina** predstavljanja slike, u memoriji se čuvaju samo podaci o elementima crteža (prava, kriva…)a prilikom iscrtavanja crtaju se samo elemnti slike.

Vektorska grafika ili geometijsko oblikovanje (*eng. Vector graphics, geometric modeling*) je način prikazivanja slike pomoću geometrijskih oblika kao što su tačke, linije, krive i poligoni, a koji se temelje na matematičkim jednačinama.

U principu, vektorski oblici se mnogo lakše pamte nego zahtevne rasterske (bitmap) slike. Skoro svi današnji računarski grafički prikazi prevode vektorsku sliku u rasterski format. Rasterska slika je pohranjena u memoriju i sadrži podatke za svaki pojedinačni piksel neke slike. Pojam vektorska grafika je većinom korišćen u kontekstu dvo-dimenzionalne računarske grafike. Skoro svako 3D prikazivanje je izvršeno pomoću 2D vektorske tehnike (pomoću tačaka, linija i poligona) [16].

Kod **rasterskog načina** predstavljana slika se predstavlja približno,tako što je površina monitora (štampača, plotera) podeljena linijama paralelnim sa horizontalnom i vertikalnom osom u mrežu kvadratića- piksela. Svakom pikselu pridruženi su atributi koji ga opisuju-intezitet osvetljenja i boja. Broj podela po horizontali i vertikali izražava rezoluciju i kvalitet slike je bolji što je rezolucija veća. Zauzeće memorije za sliku zavisi samo od broja piksela na koji je podeljena površina i broja boja koje su na raspolaganju.

Rasterska grafika ili bitmap je podatak koji predstavlja pravougaonu mrežu piksela ili obojenih tačaka, na nekom grafičkom izlaznom uređaju kao što je monitor ili na papiru. Svaka boja pojedinog piksela je posebno definisana tako da (kao primer) RGB slike sadrže tri bajta po svakom pikselu, svaki bajt sadrži jednu posebno definiranu boju.

Kvalitet jedne rasterske slike određuje ukupan broj piksela (rezolucija) kao i broj vrijednosti za svaki pojedinačni piksel (dubina boje). Ako je dubina boje veća, više se nijansi može prikazati, to znači bolju sliku kao i verodostojniji prikaz. Slike zahtjevaju mnogo memorije, zbog toga se koriste razne vrste sažimanja. Bitmap (bmp) je nesažeta datoteka koja ne koristi ni jednu vrstu sažimanja, slike u tom formatu su veoma velike, za razliku od njega mnogo popularniji i češće korišteniji je Jpeg (jpg) format koji sažima sliku a da se ne primjeti gubitak na kvaliteti iako je to nemoguće izvesti, ali je blizu stvarnosti [17].

3.2. Analiza digitalnih slika

Analiza slike se sprovodi radi utvrdivanja kvantitativnih karakteristika objekata na slici i njihovih međusobnih odnosa. Jednostavni primeri analize slike su nadzor automatskih proizvodnih linija u industriji, automatska analiza slika u biomedicini, itd. Složeniji primeri analize uključujući i prepoznavanje oblika i koriste se u robotici i u širokoj oblasti poznatoj pod nazivom mašinski ili veštacki vid (engl. computer vision).

Analiza slike se sastoji iz više postupaka koji u izvesnoj meri oponašaju proces percepcije (opažanja) kod čoveka. U ovu grupu postupaka spadaju postupci za izdvajanje ivica, postupci segmentacije, postupci za identifikaciju objekata na sceni, itd.

Digitalne slike imaju široku primenu, npr: kontrola pakovanja pilula, kontrola punjenja boca, položaj implanta u oku, prisustvo mehurića u plastici, kontrola elektronskih ploča, skeniranje papirnih novčanica, otisaka prstiju, automatskog očitavanja registarskih tablica, mikroskopska analiza crvenih i belih krvnih zrnaca, mikroskopska analiza preseka mišićnog tkiva, mikroskopska slika minerala. . .

Iz ovoga se može zaključiti digitalizacija je doprinela razvoju medicine, industrije, hemije, elektronike itd.

Dakle, segmentacija slika je prvi i najkritičniji korak analize slika, i to iz najmanje dva razloga. Prvi razlog je očigledan uticaj tačnosti segmentacije, kao prvog koraka, na tačnost merenja karakteristika koje se odvija na osnovu informacija koje se dobijaju segmentacijom. Drugi razlog je to, što se i pored više od četrdeset godina istraživanja u oblasti segmentacije slika, nije došlo do uopštenog modela, koji će dati zadovoljavajuće rezultate za sve vrste digitalnih slika. Segmentacija slika se često opisuje kao proces koji treba da izdvoji određene delove slika i da izdvoji određene objekte na slici. Definicija kaže da segmentacija ima za cilj klasterizaciju, odnosno grupisanje piksela u smislene regione na slici, tj. regione koji odgovaraju pojedinim površinama na slici, objektima ili delovima prirodnih objekata na slici. Drugim rečima, možemo reći da segmentacija treba da prepozna semantički smislene celine na slikama. Navedena definicija je vrlo zahtevna sa aspekta realizacije, jer su algoritmi za segmentaciju razvijani tokom dužeg perioda u većoj ili manjoj meri uspevali da se na adekvatan način izbore sa šumovima na slici, ali ne da pritom sačuvaju semantičke ivice na zadovoljavajući način.

Algoritme za segmentaciju možemo da podelimo prema sadržaju na osnovu kojeg se vrši segmentacija, na osnovu automatizma segmentacije, na osnovu postojanja pridruženih modela ili nekon drugom kriterijumu. Prema sadržaju na osnovu kojeg se vrši segmentacija algoritmi mogu da budu zasnovani na pikselima, regionima, sadržaju, ivicama, objektima, semantici itd. Na primer, algoritmi zasnovani na sadržaju piksela koriste obeležja niskog nivoa, kao što su boja, tekstura, intezitet ili informaciju o razlikama između uzastopnih piksela. Algoritmi za segmentaciju zasnovani na regionu koriste informacije o ivicama, poligonskim granicama, oblicima i informacije o statistici teksture za pojedine objekte. Objektno orijentisani metodi koriste pridružene karakteristike za svaki objekat. Podaci o karakterističnim obeležjima mogu biti preuzeti, tj. ekstrahovani sa različitih nivoa, ali da bi se očuvala informacija o semantičkom smislu na slici, potrebno je očuvati informacije o semantičkim ivicama na slici prilikom obrade.

Algoritme za segmentaciju slika možemo da podelimo prema automatizmu, i to na automatske i poluautomatske. Poluautomatski algoritmi podrazumevaju vođenje procesa od strane korisnika preko postavljanja određenih početnih uslova. To su npr. izdvajanje ivica pomoću aktivnih kontura, gde je potrebno da korisnik zada neku inicijalnu tačku. Automatski algoritmi za segmentaciju se izvršavaju bez zadavanja početnih uslova, kakvi su npr. algoritmi zasnovani na teoriji grafova. Takođe, još jedan od načina da podelimo algoritme za segmentaciju je da li se koristi pridruženi model (npr. funkcije gustine verovatnoće) ili ne. Da bi se procenili nepoznati parametri mogu se koristiti metode zasnovane na minimizaciji matematičkog očekivanja ili Bajsovog učenja. Za segmentaciju složenih slika se često koristi stohastičo modelovanje, kao što je Markovljevo slučajno polje koje je ekvivalentno sa Gibsovom distribucijom koja rešava problem minimizacije energije. Ovakvi stohastički modeli zahtevaju estimaciju funkije gustine verovatnoće. Ponekad se umesto stohastičkog modelovanja može upotrebiti fizički model, npr. prilikom segmentacije na osnovu termalnih karakteristika kod infracrvenih slika, a takođe, koriste se i modeli refleksije svetlosti prilikom segmentacije na osnovu boje. Generalno gledano, prilikom izdvajanja semantičkih celina na slici, možemo da kažemo da su ivice, odnosno diskontinuiteti, od najvećeg značaja za idvajanje prirodnih objekata, dok su varijacije obeležja (u boji, intenzitetu, itd.) nepoželjne smetnje. Stoga, prilikom segmentacije, potrebno je ukloniti pomenute varijacije obeležja ili, drugim rečima, grupisati elemente slike (npr. piksele) u jednu grupu ako su varijacije (npr. intenziteta) tih piksela, unutar nekog opsega. Takvu vrstu segmentacije možemo primeniti u prostornom (eng. , spatial) domenu, u domenu određenog obeležja (kolor prostora 1), u vremenskom domenu ili u združenom domenu. Dalje, ako takva segmentacija slike izdvoji veliki broj segmenata, segmenti se mogu dalje spajati prema nekom određenom kriterijumu (u praksi uglavnom prema prostornim koordinatama).

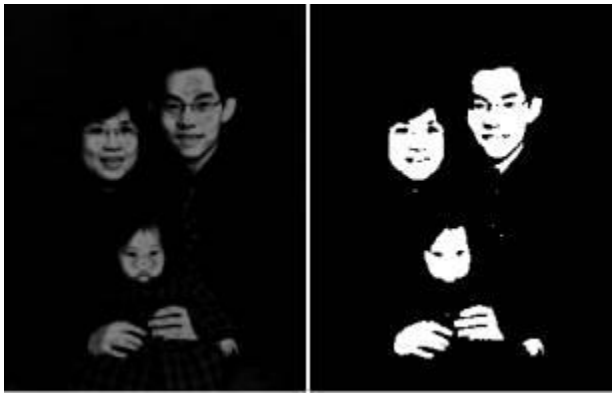
Prilikom izrade zadate aplikacije, korišćen je algoritam za prepoznavanje boje piksela. Posmatra se niz slika. Niz slika smešta sve slike jednu iznad druge. Slike se razlikuju u zavisnosti od preseka tela koji je slikan. Na kraju softver izdvaja samo svetlije piksele sa slika koji su označeni sa 1, i vrši spajanje takvih piksela(spajanje uzastupnih slika iz niza), te se nakon toga generiše 3D model, i na taj vrlo jednostavan način se dobija 3D prikaz koštanog sistema.

3.3. Thresholding

Thresholding je nelinearna operacija koja pretvara slike u boji, crno bele slike itd. u binarnu sliku, gde postoje 2 vrednosti koje se dodeljuju pikselima, ispod ili iznad odredjenog praga vrednosti.

Thresholding predstavlja metodu segmentacije, koja funkcioniše na jednostavan način. Svakom pikselu dodeljuje odredjenu visinu. Kao osnovna prednost ove metode jeste njena automatizovanost i brzina. Medjutim, thresholding, kao najveću manu ima veličinu izlaznog . obj fajla. Osim toga ova metoda odvlači dosta sistemske memorije, pa nije moguć rad sa većim brojem slika ukoliko računar nije dobro hardverski opremljen, ali ni fajlovi, tj slike generisane ovom metodom, ne predstavljaju nikakav problem za veće softvere za 3D obradu slika, pa se samim tim mogu implementirati u raznim 3D alatima.

Tokom procesa thresholdinga, pojedinačni pikseli na slici su označeni kao “object” pikseli, ako njihova vrednost nije veća od nekog praga vrednosti(uslov da objekat bude svetliji od pozadine). ”Object”piksel dobija vrednost “1”, dok pozadinski pikseli dobijaju vrednost “0”. Konačno binarna slika se kreira bojenjem svakog piksela u belu odnosno crnu boju.



Slika 2. *Thresholding*

Ključni parametar u procesu tresholdinga je izbor granične vrednosti kao što je malopre pomenuto.

Podešavanje thresholdinga je moguće ostvariti ručno, ili automatski korišćenjem raznih algoritama, poput:”Iso Data”, ”Maximum of entropy”, ”Otsu” i mnogih drugih. Što se tiče ručnog podešavanja vrednosti thresholdinga optimalan je izbor srednje vrednosti, tolike da se zadovolji uslov da objekat bude svetliji od pozadine, i da se na taj način izvrši razdvajanje pozadine od posmatranog objekta.

Automatsko izračunavanje thresholdinga je moguće na osnovu gore pomenutih metoda, ali nije preporučljivo koristiti takav proces, zbog toga što dolazi do raznih varijacija u Z pravcu, i takva procedura nije previše korisna za 3D segmentaciju.

3.4. CT skeneri

Kompjuterizovana tomografija (naziva se još i aksijalna tomografija) je savremena metoda ispitivanja, koja se zasniva na nekim starim metodama (upotreba X zraka, tehnika tomografskog snimanja) i nekim savremenim tehničkim dostignućima (upotreba računara).

Film je zamenjen sistemom detektora koji sa velikom preciznošću pretvaraju X-zrake u svetlosne ili električne impulse koji se dalje prenose u kompjuter. Kompjuter izračunava atenuaciju (slabljenje) X-zraka u delovima tkiva veoma male zapremine 0, 5* 0, 5*1, 5 mm (mogućnost razlikovanja malih promena-prostorna rezolucija). Promene se mogu pojačati ubrizgavanjem kontrasta. Definitivna slika se registruje na magnetnoj ploči i prikazuje na monitoru i snima na rendgenski film. Snima se u više nivoa. CT skener je u mogućnosti da pravi slike slojeva pojedinih organa od interesa. Samo par sekundi dovoljno je za dobijanje nekoliko slika preseka pojedinih organa u telu. CT je u mogućnosti da napravi jasno čistu sliku organa (npr. jetre) za razliku od običnog, klasičnog snimka. Gusta tkiva, kao što su kosti, prikazuju se na detektoru svetlim (belim nijansama). Manje gusta tkiva kao što su mozak i mišići u nijansama sivih boja, dok se crnom bojom prikazuju delovi tela koji imaju najmanju gustinu [11].

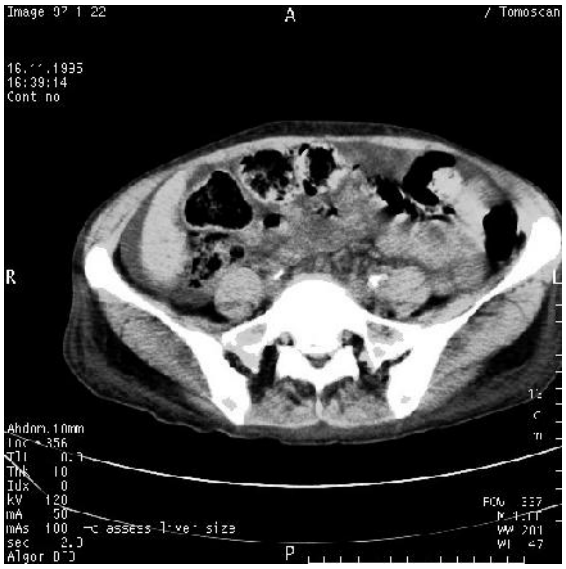
Slika 3. *CT skener novije generacije* [11]

Kod CT-a kao i kod ostalih digitalnih tehnika dobijena slika više nije posljedica direktnog dejstva X zraka na rentgenski film, kao kod klasičnih radioloških dijagnostičkih metoda. Kod digitalnih metoda ona je proizvod višestrukog detektovanja, mjerenja i izračunavanja digitalnih informacija. Sistem za prikupljanje podataka kod CT-a nalazi se u tzv. Gentriju koji predtavlja stativ CT aparata. U njemu su rentgenska cijev i detektori. Manje ili više oslabljeni ili apsorbovani snop X zraka koji su prošli kroz transvezalni presjek pacijenta izazivaju fluorescenciju detektora. Pod uticajem ove svetlosti stvara se odgovarajući električni potencijal odnosno analogni strujni impuls.

U analogno digitalnom konverteru ovi impulsi se iz analognog oblika pretvaraju u digitalne informacije odnosno nizove brojeva. Ove digitalne informacije se prenose u kompjuterski sistem, gdje se vrši njihova obrada. Procesor slike ima zadatak da u najkraćem mogućem vremenu procesira odnosno rekonstruiše CT sliku.

Slika se procesira i ponovo konvertuje u D/A konverteru u analogne signale koji formiraju CT sliku na monitoru. CT slika se sastoji od mnogobrojnih individualnih elemenata slike tzv. „pixela“. Oni imaju vlastite karakteristike atenuacije, odnosno CT brojeve, od čega zavisi nijansa zatamnjenja od bijelog i crnog svakog pojedinog piksela.

Prema tome CT slika je matematička, sa ogromnim brojem informacija jer su detektori u stanju da registruju neznatne razlike u atenuaciji X zraka. Dakle ona se razlikuje od klasične radiografske slike sa oskudnim brojem informacija jer je RTG film nedovoljno osjetljiv da registruje neznatne razlike u atenuaciji X zraka.



Slika 4. *DICOM fajl (slika dobijena sa CT skenera)* [11]

3.5. DICOM standard

3.5.1. Definicija DICOM standarda

DICOM (Digital Imaging and Communications in Medicine) standard je skup pravila, koji omogućava da se medicinski snimci i informacije razmenjuju između kompjutera i bolnica. Standard uspostavlja zajednički jezik, koji omogućava da se medicinski snimci i informacije, koji su načinjeni na jednoj vrsti opreme proizvođača, mogu da se koriste na digitalnim sistemima drugih proizvođača.

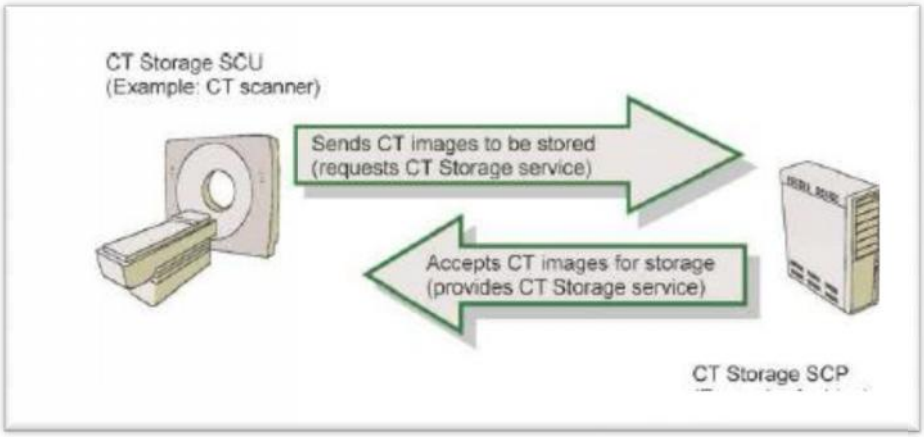
DICOM nije samo fajl format. DICOM omogućava prenos podataka, skladištenje i dizajniran je da obuhvati sve funkcionalne aspekte digitalne medicine u oblasti snimaka. Takođe, PACS informacioni sistemi se uvek pominju kada se govori o DICOM standardu. Oni su direktno povezani sa DICOM standardom.

PACS sistemi su medicinski informacioni sistemi, koji su dizajnirani da skladište, distribuiraju i prikazuju digitalne medicinske snimke. Dakle, oni omogućavaju pregled medicinskih snimaka sa CT skenera, PET skenera, ultrazvučnih aparata. . . Teško se može zamisliti savremena medicina bez DICOMa i PACSa [6].

3.5.2. Funkcionisanje DICOM-a

Sve podatke u realnom svetu: pacijente, studije, medicinsku opremu, DICOM vidi kao objekte sa odgovarajućim svojstvima, odnosno atributima. DICOM to naziva IOD (eng. Information Object Definition). Dakle, IOD predstavlja kolekciju atributa, koji opisuju neki objekat. IOD pacijenta može biti: ime i prezime, godina rođenja, težina itd. U širem smislu (pacijent kao i svaki drugi DICOM objekat) sastoji se od skupa atributa, međutim DICOM održava listu svih standardnih atributa (više od 2000), poznatijih kao DICOM rečnik podataka (eng. DICOM data dictionary), kako bi obezbedio doslednost u imenovanju atributa. Na primer, atributi pacijenta kao što su ime i prezime, godina rođenja, težina, su uključeni u DICOM rečnik podataka. Svi DICOM atributi su oblikovani u skladu sa 27 tipova podataka - vrednostima reprezentacije (eng. Value Representation -VR) što odgovara datumima, vremenima, imenima. .

Kada se akviziraju novi snimci, oni mogu da se prenose između različitih DICOM uređaja i aplikacija (aplikacionih entiteta, eng. Application Entity). DICOM ovaj proces “zahteva“ i “razmene“ podataka naziva service-rendering model, u kome DICOM aplikacije obezbedjuju servise (usluge) jedni drugima. DICOM ove asocijacije zove SOP (Service-Object Pairs) i grupiše ih u SOP klase. Na primer, čuvanje CT snimaka iz digitalnog CT skenera za digitalnu PACS arhivu, odgovara CT SOP-u za skladištenje (eng. CT Storage SOP) kao što je prikazano na slici 6 [6]. U ovom primeru, CT snimak predstavlja DICOM IOD (DICOM objekat podataka).



Slika 5. *DICOM servisi* [6]

CT skener poziva servis za skladištenje iz arhive, zatim arhiva obezbeđuje servis za skladištenje skeneru. Razlika između servisa za zahtev i servisa koji pružaju servise, DICOM naziva Service Class Users (SCU) i Service Class Providers (SCP). U konkretnom primeru, CT skener predstavlja korisnika - SCU, a digitalna arhiva predstavlja server - SCP.

Svaki mrežni prenos podataka, počinje sa uspostavljanjem asocijacije – DICOM sporazumevanje, kada dve aplikacije razmenjuju podatke. Mnoge DICOM uređaje, i aplikacije, proizvode različiti proizvođači, i zato je važno da svaki DICOM uređaj bude praćen svojim DICOM izveštajem o konformaciji od proizvođača. Ovaj izveštaj objašnjava koje SOP servise uređaj podržava, i u kojoj meri (da li podržava SCU ili SCP ili oba servisa). Dakle, DICOM izveštaj o konformaciji predstavlja najosnovnije uputstvo o svakom DICOM uređaju. Na primer, ako bolnica kupi digitalnu arhivu koja podržava samo servis za skladištenje (CT Storage SCU), a ne podržava SCP servis za skladištenje (CT Storage SCP), bolnica neće moći da u njoj skladišti CT snimke. Digitalna arhiva neće biti u stanju da obezbedi CT servis za skladištenje [6].

3.6. Softveri za obradu digitalnih slika

Na tržištu postoji izuzetno veliki broj softvera za manipulisanje sa DICOM snimcima. Osim toga u mnogima je moguće izvršiti generisanje 3D modela. Jedini problem u tim sofverima jeste cena i brzina rada. U tabeli 1. su priloženi neki od softvera i njihove cene, kao i osnovne karakteristike.

Naziv softvera	Cena	Opis
OsiriX MD	600. 00 $	DICOM viewer + 3D/4D navigacija + 3D rekonstrukcija
LEADTOOLS DICOM	900. 00$	DICOM viewer + 3D rekonstrukcija + 3D/4D/5D navigacija + orognalni prikaz
ORS Visual	2800. 00 $	PACS/DICOM viewer + 3D rekonstrukcija (zapreminski prikazr, ortogonalni) itd.

Tabela 1. *Poznati softveri za pregled DICOM snimaka i njihovu obradu u cilju dobijanja 3D modela*

Ono što je za pohvalu kod softvera koji je izradjen zadatkom završnog rada jeste sledeće:

1. Potpuno besplatan program za implementaciju izradjenog plugin-a

2. Potpuno besplatan plugin sa otvorenim kodom, koji je dostupan ostalim programerima za njegovo unapredjivanje

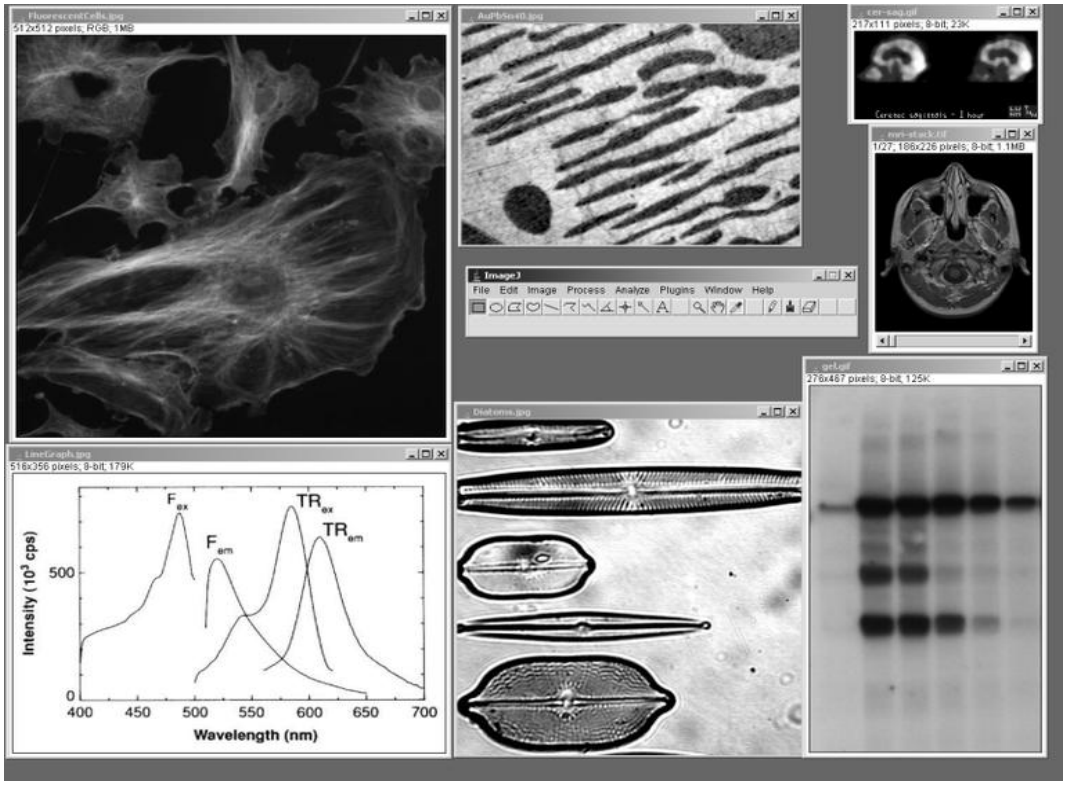
3. Sve biblioteke koje se koriste pri izradi plugin-a su potpuno besplatne

4. Izuzetno brz i koristan sofver (detalji o perfomansama plugin-a su dati u delu 4.4.)

3.6.1. ImageJ - besplatan softver za analizu i obradu digitalnih slika

ImageJ je besplatna aplikacija otvorenog koda (open source) koji se koristi za obradu slika razvijen u Nacionalnom institutu za zdravlje. Dizajniran je tako da zahvaljujući otvorenom kodu, koji je dostupan programerima, obezbedi razna proširenja preko Java pluginova. ImageJ u sebi ima ugradjen editor i Java prevodilac, te je moguće pisanje pluginova, direktno iz programa, bez korišćenja raznih softvera za izradu Java aplikacija. Dodaci napisani od mnogih korisnika i programera, rešavaju širok spektar i analizu problema iz oblasti medicine, mikrobiologije, fizike, hemije i slično.

Zahvaljujući mogućnosti velikog broja geometrijskih transformacija svaki posao obrade slike se čini izuzetno lakim, a samo vreme obrade slike je svedeno na minimum, sa uglavnom auotmatizovanim dodacima. Sam program podržava veliki broj prozora, ili fotografija, a jedino ograničenje jeste slobodna RAM memorija.



Slika 6. *ImageJ*

3.6.1.1. Osobine ImageJ-a

- ImageJ može da posluži za prikazivanje, uredjivanje, analizu, procesiranje, čuvanje i štampanje 8-bitnih, 16-bitnih i 32-bitnih slika bilo crno-belih bilo u boji.
- Podržava veliki broj formata slika poput: TIFF, PNG, GIF, JPEG, BMP, DICOM. . . .
- ImageJ podržava slikovne stacks serije slika koje dele jedan prozor
- Može izvršiti razna merenja, od površine, preko histograma, do razdaljine i uglova
- Podržava veliki broj logičkih i aritmetičkih operacija nad slikama, kontrast manipulacije, Furijeova analiza, izoštrenje slike, detekcija ivica. . . .

3.6.1.2. Pisanje plugin-a u Javi i implementiranje u ImageJ

U ovom delu će biti kompletno objašnjen postupak kreiranja jednostavnog plugin-a(pretvarnje bele u crnu boju, i obratno), tj Inverter, koji će biti implementiran u ImageJ softver.

Pre nego što počnemo sa pisanjem plugina, neophodno je da se preuzme poslednja verzija **source koda** ImageJ-a, sa sledećeg sajta: http://rsbweb. nih. gov/ij/download/src/. Potrebno je da se preuzeti fajl raspakuje, i time će biti kreiran **source** direktorijum.

Ovaj primer izrade plugina je prikazan u radnom okruženju NetBeans iz razloga što je potpuno jednostavan za korišćenje, a ujedno i besplatan program, mada se postupak ne razlikuje mnogo ni u ostalim programima za razvoj aplikacija.

U nastavku sledi objašnjenje kreiranja jednostavnog plugin-a za imageJ.

1. Pokrenuti NetBeans, zatim FILE-NEW PROJECT-JAVA APPLICATION. Uneti naziv aplikacije. Nakon toga treba odčekirati opciju „create main class“, iz razloga što mi kreiramo novu klasu koja mora sadržati donju crtu na kraju reči. Npr:“Inverter_“. Klasu kreirati desnim klikom na projekat, NEW-NEW JAVA CLASS, i dati joj gore naveden naziv.

2. Potrebno je dodati imageJ biblioteke koje će biti korišćene pri izradi aplikacije. To se postiže na sledeći način: desni klik na Inverter_-PROPERTIES-LIBRARIES-ADD JAR/FOLDER. U zavisnosti od namene plugin-a, i biblioteka koje će biti korišćene pri izradi, možemo dodavati beskonačno mnogo biblioteka, ali za potrebe ovog plugin-a neophodno je dodati osnovnu biblioteku, koja se koristi kod svih pluginova za ImageJ, a to je **ij. jar**(ova biblioteka je takodje dostupna u source folderu), i **tools. jar.**

3. Neophodno je da se promeni direktorijum preko koga će se aplikacija izvršavati, i takodje moramo promeniti Main klasu. Kliknuti na RUN(takodje se nalazi unutar kartice PROPERTIES). Za „Working directory“ izabrati folder „ij“, koji se nalazi unutar source foldera koji smo na početku raspakovali. Unutar polja „Main class“ uneti „ij. ImageJ“.

4. Unutar Inverter_. java, uneti sledeći kod:

```
import ij. *;
import ij. plugin. filter. PlugInFilter;
import ij. process. *;
import java. awt. *;

public class Inverter_ implements PlugInFilter {
```

Sledeće što je potrebno je metod za podešavanje plugin-a. Za slučaj da dobijamo "about" kao argument, pozivamo metod "showAbout", koji će otvoriti dijalog. U tom slučaju vraćamo DONE, zato što ne želimo da se metod **Run** pozove. Ovaj plugin radi sa 8-bitnim slikama.

```
public int setup(String arg, ImagePlus imp) {
if (arg. equals("about")) {
showAbout();
return DONE;
}
return DOES_8G+DOES_STACKS+SUPPORTS_MASKING;
    }
```

Metoda "Run" sprovodi stvarnu funkciju plugina. Mi smo dobili stvarnu sliku, zatim sliku kao niz piksela kao što je 8-bitna slika (256 mogućih vrednosti). Moramo imati na umu da je niz piksela jednodimenzionalan, sadrži jednu liniju nakon skeniranja druge, zatim čitamo širinu slike itd.

```
public void run(ImageProcessor ip) {
byte[] pixels = (byte[])ip. getPixels();
int width = ip. getWidth();
    Rectangle r = ip. getRoi();
```

Sada deklarišemo 2 promenljive. Posmatramo trenutnu vrednost promenljive **i,** i dodeljujemo suprotnu vrednost piksela, oduzimanjem od 255.

```
int offset, i;
for (int y=r. y; y<(r. y+r. height); y++) {
offset = y*width;
for (int x=r. x; x<(r. x+r. width); x++) {
i = offset + x;
pixels[i] = (byte)(255-pixels[i]);
}
}
    }
```

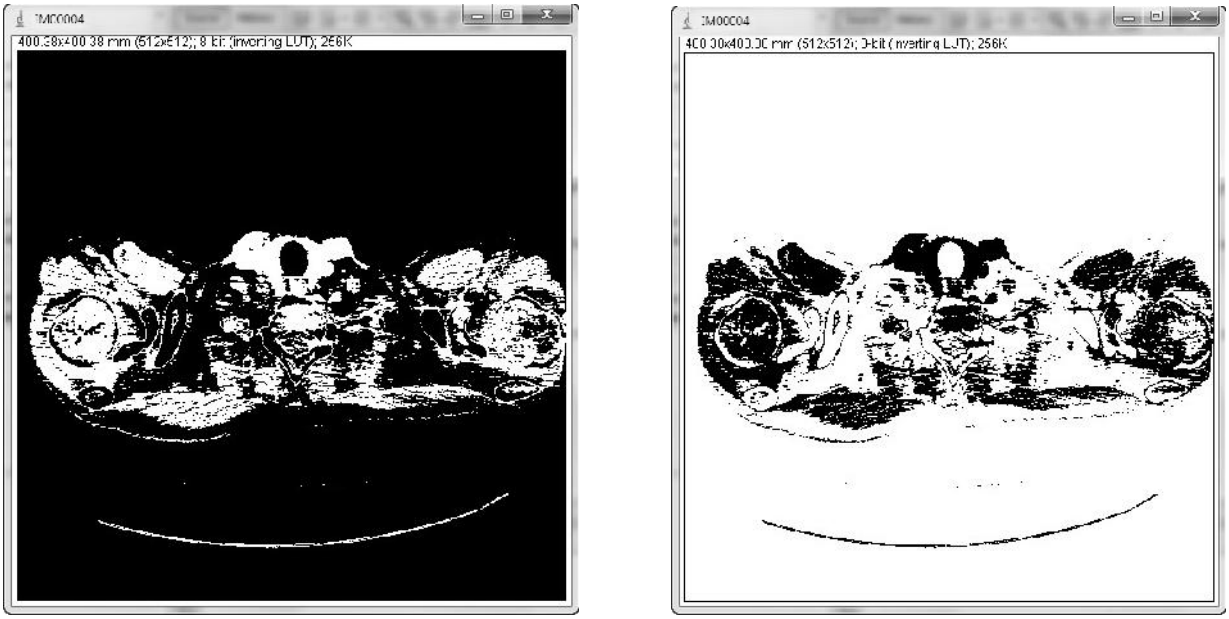
ShowAbout koristi metodu s**howMessage** iz klase **IJ,** za prikazivanje teksta. Prvi parametar je naslov, a drugi tekst poruke.

```
void showAbout() {
IJ. showMessage("About Inverter_. . . ",
"This sample plugin filter inverts 8-bit images. Look\n" +
"at the 'Inverter_. java' source file to see how easy it is\n" +
"in ImageJ to process non-rectangular ROIs, to process\n" +
"all the slices in a stack, and to display an About box. "
);
}
    }
```

Nakon što smo ispratili ove korake, pokrenuti aplikaciju iz NetBeans-a, klikom na Run Main Class. Sada primećujemo da se pokreće ImageJ aplikacija, jer smo kao „working directory“ postavili „ij“. Plugin pokrećemo na sledeći način.

PLUGINS-COMPAILE AND RUN i nalazimo src folder koji se nalazi na mestu gde je snimljen projekat, i zatim pokrećemo u našem slučaju „Inverter_. java”. Prikazaće se obaveštenje da nema otvorenih slika, pa je potrebno dodati 8-bitnu sliku, I pokrenuti plugin. Nakon toga plugin će raditi.

Ukoliko slika nije binarna, potrebno je prevesti u 8-bitnu. Klikom na PROCESS-BINARY-MAKE BINARY.



Slika 7. *Primer: a)pre invertovanja , b) nakon pokretanja plugin-a, tj nakon invertovanja*

4. KREIRANJE ELEKTRONSKOG MODELA KOSTI I IZGLED GOTOVOG PLUGIN-A ZA ANALIZU SLIKA

U ovom poglavlju je objašnjen komplet postupak kreiranja plugin-a u Javi i njegova implementacija u ImageJ. Takodje priložen je i opis osnovnih klasa u Javi, koje su neophodne za pravilno funkcionisanje aplikacije, i na kraju i sam izgled plugin-a sa objašnjenjem načina izrade 3D modela, komandi itd.

Kompletan plugin je radjen u Java programskom jeziku kao što je pomenuto. Biblioteke koje su korišćene (ij. jar, tools. jar, Jama. jar) su potpuno besplatne, dok sve ostale biblioteke su sastavni deo Java programskog jezika, koje su takodje besplatne.

Segmentacija DICOM snimaka i generisanje . obj fajlova je omogućena pomoću metode thresholding-a. Ta metoda je objašnjena u delu 3. 4, a njenu primenu ćete moći da vidite u nastavku.

4.1. Izgled aplikacije

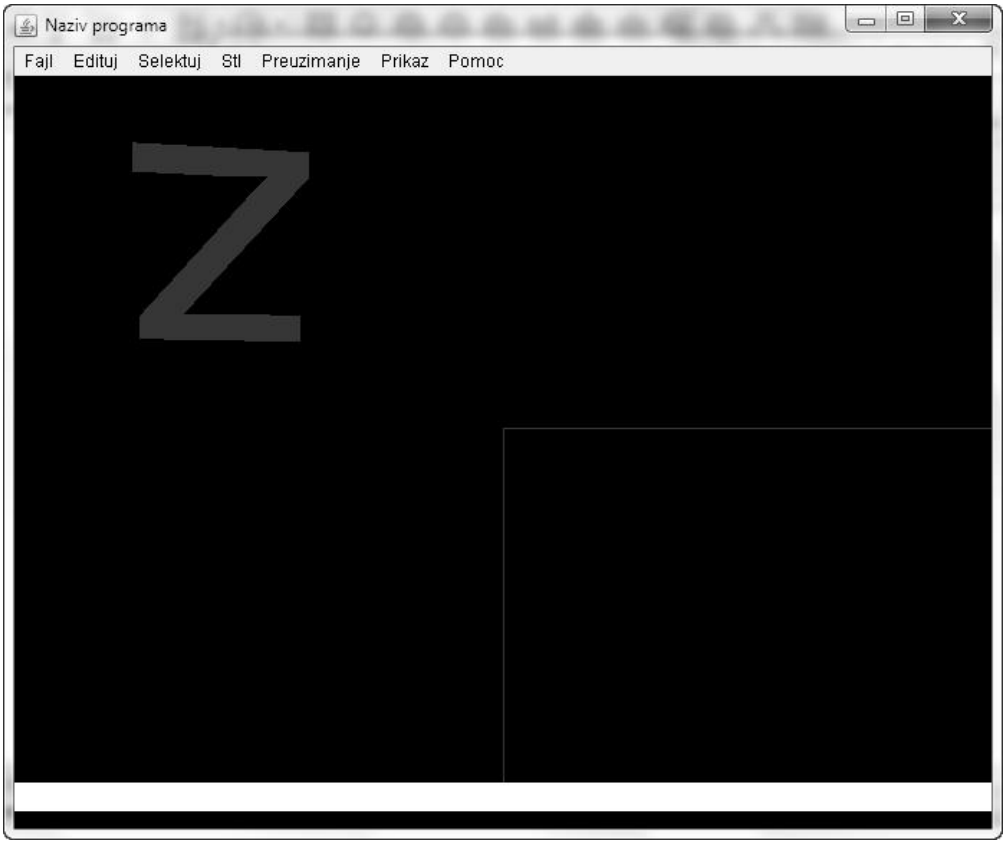
Sam izgled plugin-a, kao i broj komandi koje su neophodne za izradu gotovog 3D modela su znatno pojednostavljeni, pa je samim tim i vreme koje je potrebno od učitavanja niza slika, do dobijanja konačog rezultata, svedeno na minimum.

Postoje 2 načina obrade slika u ovom plugin-u:

1. **Dobijanje zapremine tela** - realni prikaz pune zapremine tela pacijenta sa spoljašnjom površinom. Ovakav način prikaza je izuzetno efikasan pri detaljnom pregledu 3D modela, a samim tim je takodje dostupan i skeletni sistem, koji se jasno ocrtava na 3D modelu. Detaljno objašnjenje postupka je dato u delu 5. 1. 1.

2. **Prikaz 3D površine** – koji se odnosi isključivo na kosti. Dakle, ovom metodom se prikazuju kosti, koji su dobijeni segmentacijom slika (thresholding). Vrlo povoljna metoda jer prikazuje veran(realan), grafički prikaz kosti, bez tela pacijenta, što je izuzetno povoljno za štampu 3D modela kosti, bez dodatne obrade modela. Detaljno objašnjenje postupka je dato u delu 5. 1. 2.

Na slici 8. je prikazan radni prozor aplikacije, nakon pokretanja plugin-a.



Slika 8. *Radni prozor aplikacije*

1. Fajl

Iz padajućeg menija FAJL, omogućena je manipulacija fajlovima.

Slika 9. *Fajl padajući meni*

Opcije koje su dostupne iz padajućeg menija su:

Dodaj - koristi se za odabir metode (prikaz kao površina ili kao zapremina). Takodje su dostpuna podešavanja koja se odnose na boju, naziv fajla, threshold-a, faktora itd. Ova opcija se može koristiti tek nakon učitavanja niza slika, i taj postupak će biti objašnjen u nastavku.

Slika 10. *Izgled opcije "Dodaj"*

Ucitaj . obj – služi za učitavanje već odradjenog 3D modela sa ekstenzijom . obj.

Izbrisi – koristi se za brisanje trenutnog prikaza 3D modela, i opcija je dostupna nakon selektovanja modela kojeg želimo da obrišemo.

Snimi kao . obj fajl – ova opcija snima trenutni 3D prikaz na željeno mesto na hard disku kao . obj fajl.

Izadji – izlaz iz programa

1. **Edituj**

Prikazi kao – Ova opcija omogućava manipulaciju 3D modelom, u realnom vremenu. Npr. ukoliko želimo da trenutni prikaz koji je kao zapremina, prebacimo u površinski prikaz, dovoljno je da selektujemo dati model, i klikom na željenu opciju predjemo iz jednog režima u drugi.

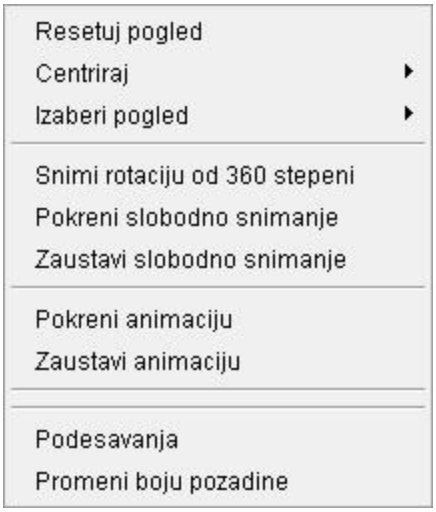
Prikazi/sakrij

2. **Selektuj**

Klikom na ovaj padajući meni, dobijamo ponudu 3D modela koji se trenutno koriste. Ova opcija je korisna zbog manipulacije odredjenim delovima.

3. **Prikaz**

Ovaj padajući meni sadrži podešavanja prikaza, počevši od pogleda na 3D model, preko animacija, boje pozadine do ostalih podešavanja.

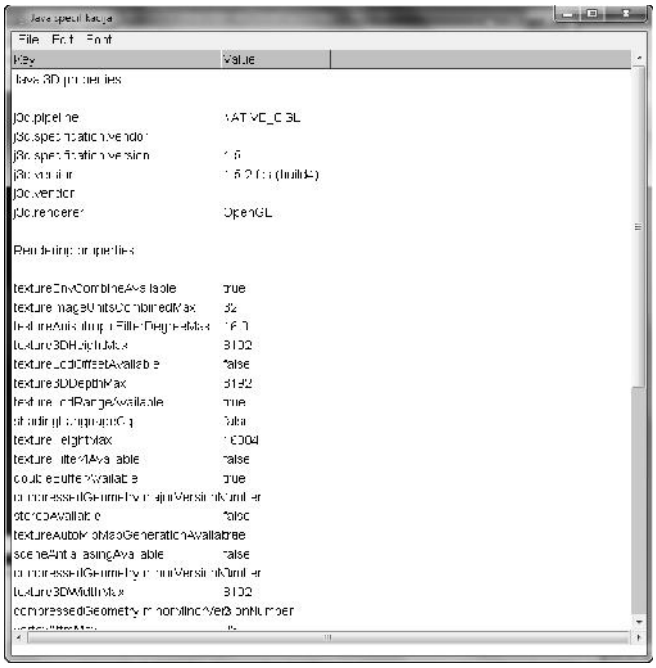


Slika 11. *Igled menija "Prikaz"*

Ono što je bitno izdvojiti iz ovog menija jeste mogućnost snimanja animacije. Snimanje animacije je moguće ostvariti automatski, slobodnim rotiranjem od 360 stepeni, ili ručno po želji korisnika.

4. Pomoć

U ovom meniju su dostupne osnovne informacije i programu, kao i o Java3D specifikaciji(sistemska obaveštenja, raspoloživa memorija, maksimalni učinak grafike…).



Slika 12. *Izgled prozora "Java 3D Specifikacija"*

5. Stl

U okviru ovog menija imamo 2 opcije:

a)**MeshLab**-Konverter . obj fajla u neki drugi popularni 3D format.

b)**STL viewer** – Pregled STL fajlova, kao i njihova modifikacija.

Objašnjenje ovih softvera će biti dato u nastavku.

6. Preuzimanje

Iz ovog menija možemo direktno pristupiti lokaciji za preuzimanje gore navedenih sofvera

4.2. Opis koda

Zbog opširnosti koda programa, neće biti opisan celokupni kod, već će biti obraćena pažnja na bitne delove koda, kao i delove koda koji su izmenjeni u odnosu na originalni plugin.

Ovaj program se sastoji iz 14 paketa, odnosno preko 120 klasa. Postoji veliki broj metoda u svakoj od klasa. U nastavku će biti objašnjeni osnovni delovi koda.

1. Paket "Default package" sadrži:

checkJava3D() metodu koja služi za proveru Java3D.

```
String version = ij3d. Install_J3D. getJava3Dversion();
```

Verzija programa je ona koja se dobija pozivanjem getJava3Dversion() metode koja se nalazi u ij3D paketu.

```
System. out. println("version = " + version);
```

Štampa se verzija java 3D.

```
if(version != null && Float. parseFloat(version) >= 1. 5)
    return true;
```

Ukoliko je verzija jednaka ili veća od 1. 5 vraća se vrednost true.

```
boolean inst;
if(version != null) {
    inst = IJ. showMessageWithCancel("Detektovana je starija verzija JAVA
3D",
        "Java 3D verzija je " + version +
        "ali neophodna verzija je >= 1. 5 \n" +
        "Da li zelite automatsku instalaciju Java 3D");
} else {

    inst = IJ. showMessageWithCancel(
        "Java 3D nije instalirana\n",
        "Java 3d nije instalirana\n" +
        "Auto-instalacija?");
```

```
            }
```

Ukoliko je verzija manja od zahtevane otvara se prozor kojim je moguće pokrenuti autoinstalaciju.

```
            if(inst) {
                  if(ij3d. Install_J3D. autoInstall())
                        IJ. showMessage("Restartujte ImageJ sada!");
```

Ukoliko korisnik da potvrdan odgovor, poziva se metoda autoInstall() iz ij3D paketa, odnosno Install_J3D klase.

```
            }
            return false;
      }
```

Instalacija se zasniva na pokretanju određene URL adrese u zavisnosti od sistemskih informacija.

U ovom paketu takodje se nalazi klasa Softver_domaci koji se implementira pomoću PlugIn-a.

```
public class Softver_Domaci implements PlugIn {

if(checkJava3D())

                  new ImageJ3Dviewer(). run(args);

}
```

U koliko je proverena pokreće se ImageJ3Dviewer klasa iz ij3D paketa.

Klasa "Menu" je zadužena za korisnički interfejs plugin-a.

U opisu programa su priložene i objašnjene neke od opcija programa. Svaka opcija je ostvarena u ovoj klasi, a kasnije se po potrebi poziva iz Executer klase i šalje na dalju "obradu".

```
public class Meni extends MenuBar implements ActionListener,
                                              ItemListener,
                                              UniverseListener {
private Menu viewMenu;
private Menu fileMenu;
private Menu helpMenu;
private Menu stl;
private Menu download;
private MenuItem oProgramu;
private MenuItem stlViewer;
private MenuItem meshLab;
private MenuItem downloadStl;
private MenuItem downloadMesh;
```

U kodu iznad su definisani neki od padajućih menijia, kao i delovi tih menija odnosno MenuItem.

U nastavku je priložen kod pomoću kojeg se kreiraju padajući meniji. Public Meni, kao argument prosledjuje univ, koji je ustvari Image3Duniverse.

```
public Meni(Image3Duniverse univ) {
        super();
        this. univ = univ;
        this. executer = univ. getExecuter(
        univ. addUniverseListener(this);
```

fileMenu = createFileMenu();

fileMenu poziva metodu createFileMenu koja je kreirana u nastavku.

this. add(fileMenu);

Pomoću ovog koda se kreira padajući meni.

public Menu createFileMenu() {

Menu file = new Menu(“Fajl”);

Pomoću ovog dela koda se dodaje naziv padajućem meniju

add = new MenuItem(“Dodaj”);
add. addActionListener(this);
file. add(add);

Dodaje se prvi MenuItem u okviru File padajućeg menija, sa nazivom “Dodaj”. I takodje se dodaje ActionListener, koji govori kada da se se “Dodaj aktivira”, a to se događa kada se klikne na tu opciju.

Priloženo je još linija koda koje potpuno isto funkcionišu kao i prethodno naveden primer dodavanja MenuItem-a.

importObj = new MenuItem(“Ucitaj. obj”);
importObj. addActionListener(this);
file. add(importObj);

delete = new MenuItem(“Izbrisi”);
delete. setEnabled(false);
delete. addActionListener(this);
file. add(delete);

file. addSeparator();

……. .

return file;

}

Na isti način se kreiraju i ostale metode za kreiranje menija, i kod unutar njih je potpuno identičan, samo se menja naziv MenuItem-a.

Da bi program znao kada da je koja opcija odabrana, kreira se sledeća klasa koja nasledjuje ActionListener:

```
private class SelectionListener implements ActionListener {
        public void actionPerformed(ActionEvent e) {
```

AtionPerformed uzima kao argumente ActionEvent, i dalje se prosledjuje klasi executer.

```
            executer. select(e. getActionCommand());
        }
    }


    public void actionPerformed(ActionEvent e) {
        Object src = e. getSource();
```

"e" je kreiran i pokazuje na ActionEvent. e. getSource uzima kod u zavisnosti od toga koja je opcija izabrana i koji je taster pritisnut. Ukoliko se taj source kod poklapa sa nekim od navedenih opcija, onda će se pozvati klasa executer odnosno metoda te klase.

```
        if (src == bgColor)
            executer. changeBackgroundColor();
        else if(src == add)
            executer. addContent(null, -1);
        else if(src == resetView)
            executer. resetView();
```

Dole je uveden sistem TRY-CATCH, jer nije sigurno da li će program uspeti da odradi korisnikovu komandu. Ukoliko korisnik želi da pokrene STL viewer, a nema ga instaliranog na računaru, zahvaljujući izuzetku CATCH, dobija se izlazna poruka sa greškom.

```
   else if (src==stlViewer)
 try {
  executer. stl();
 } catch (IOException ex) {
  JoptionPane. showMessageDialog(null, "Greska! Proverite da li imate instaliran STL VIEWER.
\n Ukoliko nemate, mozete direktno iz padajuceg menija 'preuzimanje' to i uciniti. ");
 }
```

Klasa “Executer” predstavlja operativnu klasu programa. Pomoću nje se vrše glavne operacije u programu, i iz nje se pozivaju metode za kreiranje površinskog prikaza 3D modela.

“Fajl” menu:

```
public void addContent(final ImagePlus image, final int type) {
        new Thread() {
            @Override
            public void run() {
                addC(image, type);
            }
        }. start();
    }
```

Pomoću gore navedenog koda, moguće je kreirati 3D model. Ono što ova metoda zahteva jeste ImagePlus, i tip. Sadrži takodje metodu “run” pomoću koje se kreće sa izvršavanjem programa, i tu se poziva metoda addC(image, type).

```
private Content addC(ImagePlus image, int type) {

        int img_count = WindowManager. getImageCount();
        Vector windows = new Vector();
        String[] images;
        for(int i=1; i<=img_count; i++) {
```

“img_count” predstavlja broj slika sa kojima se radi.

```
            int id = WindowManager. getNthImageID(i);
            ImagePlus imp = WindowManager. getImage(id);
            if(imp != null && !imp. getTitle(). equals(“3d”)){
                windows. add(imp. getTitle());
            }
        }
```

```
if(windows. size() == 0) {
        IJ. error("Nije otvorena ni jedna slika");
        return null;
}
```

Ukoliko je broj slika jednak nuli, izbacuje se greška da nije otvorena ni jedna slika, i sistemu se vraća **null.**

```
images = (String[])windows. toArray(new String[]{});
String name = image == null ? images[0] : image. getTitle();
String[] types = new String[] {
        "Orto prikaz", "Zapreminski prikaz", "Povrsina kao 3D"};
type = type < 0 ? 0 : type;
int threshold = type == Content. SURFACE ? 50 : 0;
int resf = 2;
```

Postoje tri vrednosti sa kojima se radi:Orto prikaz, Zapreminski prikaz i Površina kao 3D. To je niz od 3 vrednosti, i taj niz predstavlja tip.

U nastavku se kreira dialog nakon klika na "Dodaj" iz padajućeg menija "Fajl". U prvom delu koda su data osnovna podešavanja korisniku, poput threshold, boje, faktora itd.

```
GenericDialog gd = new GenericDialog(
                        "Dodaj…", univ. getWindow());
gd. addChoice("Slika", images, name);
gd. addStringField("Ime", name, 10);
gd. addChoice("Prikazi kao", types, types[type]);
gd. addChoice("Boja", ColorTable. colorNames,
                                ColorTable. colorNames[0]);
gd. addNumericField("Threshold", threshold, 0);
gd. addNumericField("Faktor", resf, 0);
gd. addMessage("Boje:");
gd. addCheckboxGroup(1, 3,
                new String[] {"crvena", "zelena", "plava"},
                new boolean[]{true, true, true});
```

U nastavku koda se zahtevaju željene vrednosti korisnika. Odnosno odabir ponudjenih opcija i podešavanja za 3D rekonstrukiju modela.

```
final TextField th = (TextField)gd. getNumericFields(). get(0);
final Choice di = (Choice)gd. getChoices(). get(1);
di. addItemListener(new ItemListener() {
        public void itemStateChanged(ItemEvent e) {
                if(di. getSelectedIndex() == Content. SURFACE)
                        th. setText(Integer. toString(50));
                else
                        th. setText(Integer. toString(0));
        }
});
final Choice im = (Choice)gd. getChoices(). get(0);
final TextField na = (TextField)gd. getStringFields(). get(0);
im. addItemListener(new ItemListener() {
        public void itemStateChanged(ItemEvent e) {
                na. setText(im. getSelectedItem());
        }
});
```

Dalji kod iz ove metode je označen sa tri tačke, ali njegova jasnoća je na visokom nivou, jer se sve svodi na odabir korisnika, i kupljenja tih vrednosti.

```
…
```

Nakon što se snimi dati 3D model u . obj fajl, moguće ga je očitati i ponovo manipulisati njime unutar programa. Otvaranje . obj fajla se vrši na sledeći način, i taj postupak je objašnjen u nastavku.

```
public void importWaveFront() {
        OpenDialog od = new OpenDialog("Selektuj . obj fajl", OpenDialog.
getDefaultDirectory(), null);
```

Otvara se dijalog za selektovanje . obj fajla.

```
        String filename = od. getFileName();
        if (null == filename) return;
        if (!filename. toLowerCase(). endsWith(". obj")) {
                IJ. showMessage("Morate izabrati iskljucivo . obj fajl");
                return;
```

4.2.1. Proces 3D rekonstrukcije 2D snimaka DICOM formata

Deo koda koji je najbitniji pri gore navedenom procesu jeste kreiranje tzv. virtuelnog univerzuma i prikazivanje niza slika, odnosno "image stack", sam proces se ostvaruje u nekoliko koraka.

Prva stvar koja se zahteva od korisnika jeste da otvori taj niz slika, koji u stvari predstavlja više od 400 slika, koje su uzete od CT skenera, koje je napravio prilikom prolaska kroz 400 uzastupnih preseka tela.

```
String path = "/home/student/primer. tif";
ImagePlus imp = IJ. openImage(path);
new StackConverter(imp). convertToGray8();
```

Ukoliko se proverom utvrdi da slike nisu 8-bitne, to se vrši pozivanjem konvertera.

Nakon što je izvšeno kreiranje virtuelnog univerzuma, on se prikazuje na prozoru aplikacije.

```
Image3Duniverse univ = new   Image3Duniverse();
univ. show();
```

3D objekat je krieran i dodaje se od strane univerzuma. To se odnosi na sledeću liniju koda:

```
Content c = univ. addVoltex(imp);
```

Funkcionisanje ove aplikacije je jednostavno. Prikazivanje 3D modela na osnovu 2D DICOM snimaka, funkcioniše tako što se niz slika prikazuju jedna iznad druge poredjane po odgovarajućem redosledu. Svetliji pikseli se odvajaju od tamnijih, i na taj način se vrši njihovo povezivanje. Tamnijim pikselima se dodeljuje vrednost 0, a svetlijim 1. Vrši se povezivanje 2 uzastupna piksela sa vrednosti 1, i na taj način korak po korak se dobija zapreminski odnosno površinski model.

Za potrebe ovog seminarskog se kreira površinski model:

```
c = univ. addSurfacePlot(imp);
```

Dodavanje nove slike, i prikazivanje 3D modela kao površine:

```
public Content addMesh(ImagePlus img) {
        return addContent(img, Content. SURFACE);
    }

public Content addMesh(ImagePlus image, Color3f color, String name,
        int threshold, boolean[] channels, int resamplingF) {

        return addContent(image, color, name, threshold, channels,
                resamplingF, Content. SURFACE);
    }
```

Content addMesh vraća sadržaj koji je dodat, a nulu ukoliko je došlo do neke greške prilikom dodavanja.

4.2.2. Izmena koda

Kardinalne izmene koje su načinjene u programu se odnose pre svega na korisnički interfejs. Sve opcije u programu su prevedene na srpski jezik.

Opcije koje su izbačene iz osnovne verzije plugina su: prikaz kao zapremina, ortognalni prikaz, snimanje 3D rotacije, razna podešavanja, snimanje kao DXF format, JAVA 3D properties, razne vrste prikaza modela itd. Celokupne izmene su izvšene radi lakšeg manipulisanja modelom, i jednostavnijeg dobijanja gotovog 3D modela. Same opcije su isključene iz programa, prepoznavanjem tog dela koda programa, i njegovim brisanjem, ili postavljanjem koda pod komentar.

Program je sada prilagodjen isključivo 3D rekonstrukiciji 2D snimaka DICOM formata, baš kao što je i zahtevano seminarskim radom.

Opcije koje su dodate u novu verziju programa se odnose na konvertovanje . obj fajla u stl. Korisniku se omogućuje direktno iz programa otvaranje 2 programa:

1. **MeshLab** koji služi za prikaz 3D modela dobijenog iz plugin-a, prikaz mreže trouglova, itd. Kao najbitnije svojstvo ovog programa jeste to što omogućava konverziju . obj fajla u . stl. Time je ostvarena osnovna komunikacija plugin-a i MeshLab softvera. Nakon snimanja u . stl format, moguće je dalje manipulisanje 3D modelom u nekim od 3D softvera za njegovo 3D štampanje.

2. **STL viewer** program pomoću kog je mogući pregled novo dobijenog formata fajla . stl.

Takodje je dodat padajući meni „Preuzimanje“, iz koga se samo jednim klikom mogu preuzeti MeshLab i STL viewer.

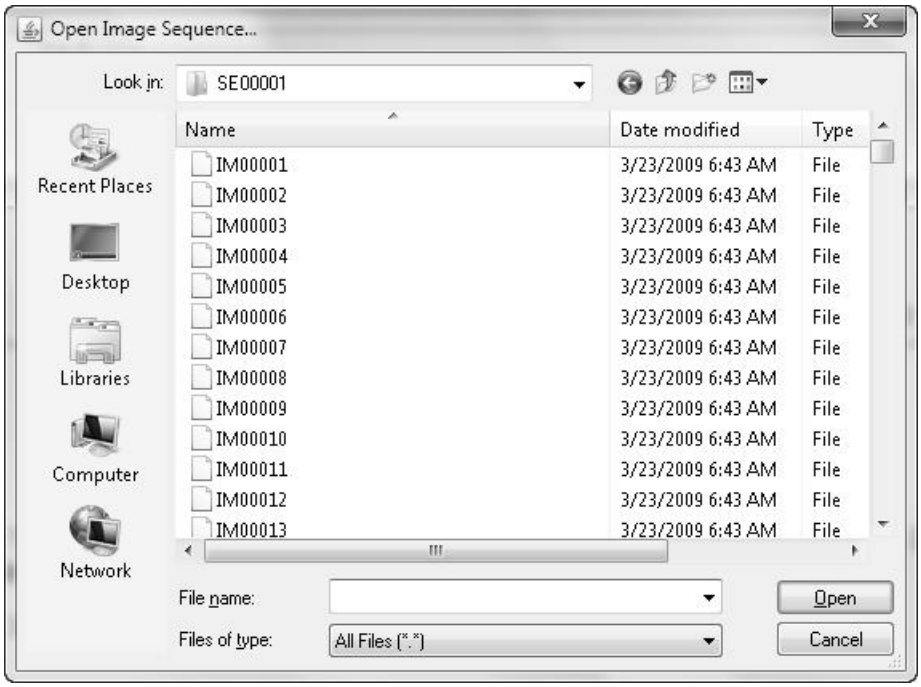
Osnovna zamisao prilikom prepravke originalnog programa, jeste kao što je pomenuto svođenje jednog kompleksnog i izuzetnog kvalitetnog programa, na minimalno komplikovan i takodje veoma koristan program.

Opis izmenjenog koda nije posebno priložen, već je prilikom objašnjenja osnovnog koda, ujedno i predstavljen izmenjeni kod programa.

4.3. Proces pretvaranja DICOM snimaka u elektronski 3D model, metode i načini

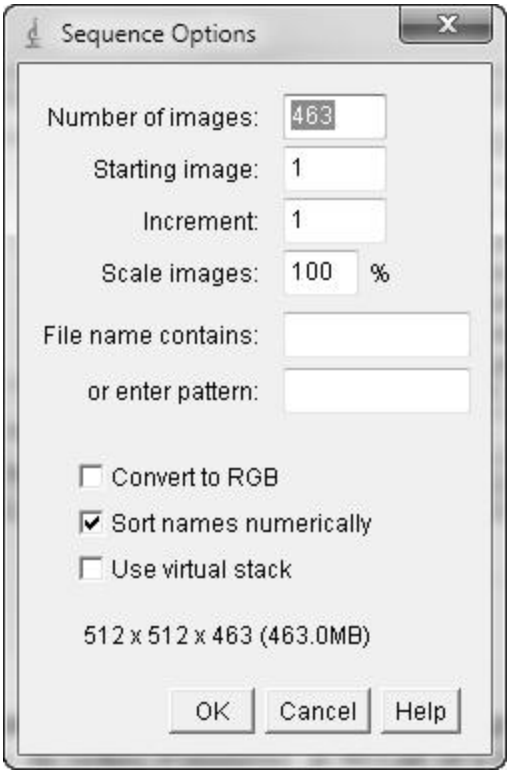
U ovom delu će biti dato objašnjenje postupka koji se odnosi na učitavanje niza slika iz ImageJ-a, njegova implementacija u plugin-u, i na kraju dobijanje 3D modela.

Nakon pokretanja ImageJ-a, potrebno je učitati niz slika: File-Import-Image Sequence…Nakon toga se otvara prozor kao na slici 13, i tu je potrebo da odaberemo početnu sliku formata DICOM.



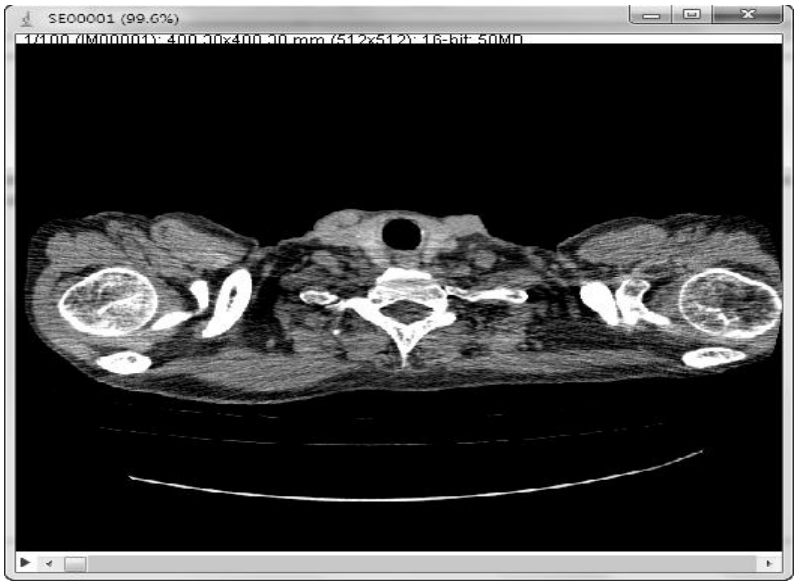
Slika 13. *Open Image Sequence…*

Otvara se novi prozor pod nazivom “Sequence Options”(slika 17), gde je potrebno da podesimo broj slika iz foldera, sa kojima želimo da radimo. Ponudjeno je 463, jer se u folderu SE00001, nalazi 463 Dicom snimaka. Takodje možemo podesiti startnu sliku, jer su sve slike sortirane po nazivu.



Slika 14. *Sequence Options*

Kada smo izvršili odgovarajuća podešavanja potvrdićemo ih klikom na dugme OK. Potvrdom se otvara novi prozor, gde je u smešteno svih 100 slika koje smo odabrali.

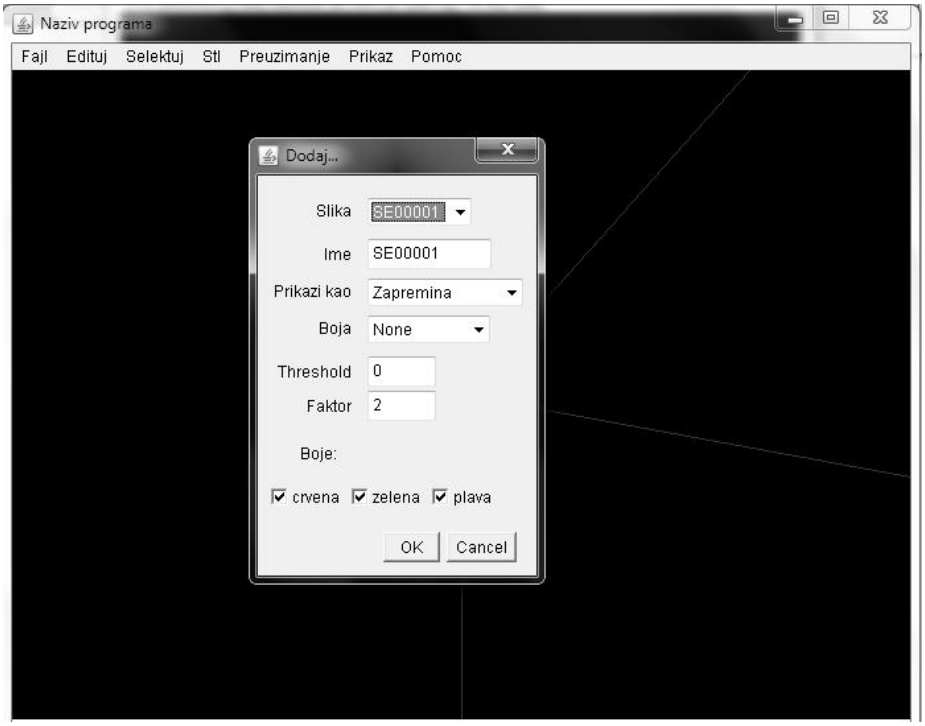


Slika 15. *Izgled radnog prozora nakon otvaranja DICOM snimaka*

Na donjem delu prozora primećujemo slider , koji pokazuje da je prvi "slice" ukoliko se nalazi krajnje levo, odnosno poslednji slice ukoliko se nalazi krajnje desno. Takodje postoji dugme play, koje kada kliknemo na njega počinje prelaz kroz svih 100 slika, I same promene se ažuriraju na prozoru.

Pošto smo učitali 100 DICOM slika, sada možemo pokrenuti aplikaciju iz plugins padajućeg menija ImageJ-a.

Nakon pokretanja plugin-a, početni prozor će izgledati kao na slici 19.



Slika 16. *Izgled početnog prozora plugin-a, nakon učitavanja niza slika*

4.4. Načini prikaza obradjenih DICOM snimaka

Ovaj deo je nastavak prethodnog, kao objašnjenje mogućih načina prikaza i analiza DICOM snimaka. Ovaj plugin sadrži tri osnovna prikaza i to kao:

1. Zapreminski prikaz
2. Ortogonalni prikaz
3. Površinski prikaz

Svaki od ovih prikaza biće naknadno objašnjen, i svaki sadrži prednosi i mane u odnosu na ostale prikaze, ali najčešće primenjivan u svrhu izrade fizičkog modela kosti je „Površinski prikaz“.

4.4.1. Zapreminski prikaz

Sam plugin nudi dobro dokumentovan API za izuzetno visok kvalitet 3D vizuelizacije. Zapreminski prikaz ostvaruje 3D efekat, tako što “pakuje” slike jedne na drugu, koje su razdvojene na odredjenoj udaljenosti

. Za svaki piksel u svakoj slici, vrednost za transparentnost je dodeljena, što naravno zavisi od osvetljenosti odredjenog piksela. Bela boja odaje veću prozirnost za razliku od crne. Ovaj način omogućava složeniju i interaktivniju vizuelizaciju. .

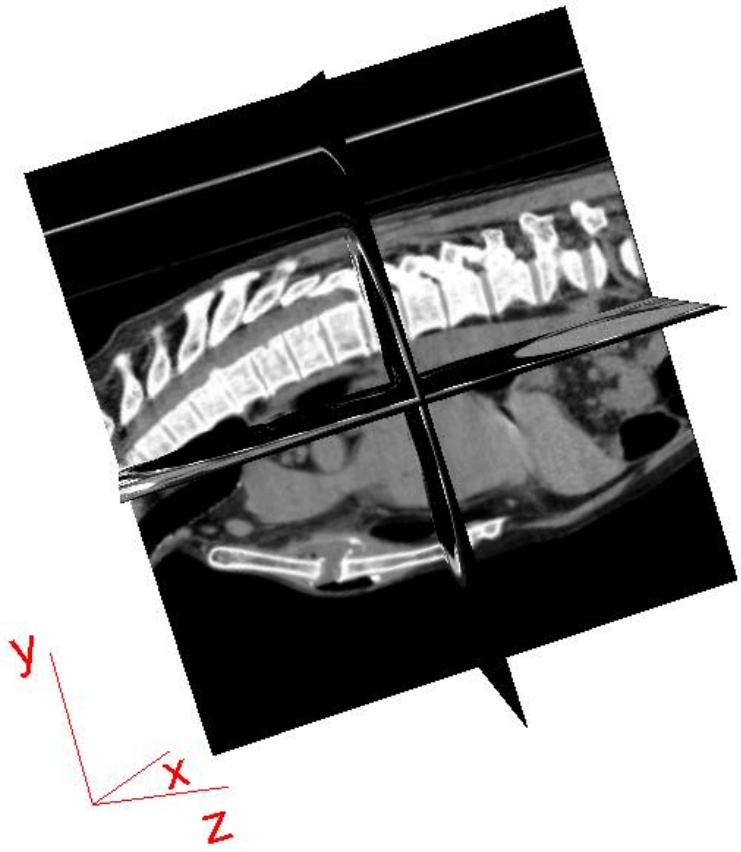
Ukoliko želimo da prikažemo kompletnu zapreminu, nije potrebno podešavanje thresholda, i faktora, dovoljno je samo da potvrdimo dati prozor, opcijom OK.

Program će sam izvršiti generisanje datog seta podataka, i 3D model će imati izgled kao na slici 16. Za zapreminski prikaz moguće je učitati i svih 463 slika, jer ovaj metod zahteva manje RAM memorije, i manje je zahtevan. Na slici se jasno vidi 3D model grudnog koša pacijenta, sa jasnim konturama.

Slika 17. *Zapreminski prikaz*

4.4.2. Ortogonalni prikaz

Ortogonalni predstavlja 3 ortogonalne ravni, koje prolaze kroz 3 dimenzije. Početno su postavljene u centru posmatranog objekta. Pokretanjem željene ravni omogućeno je pomeranje iste kroz preseke zapremine tela.



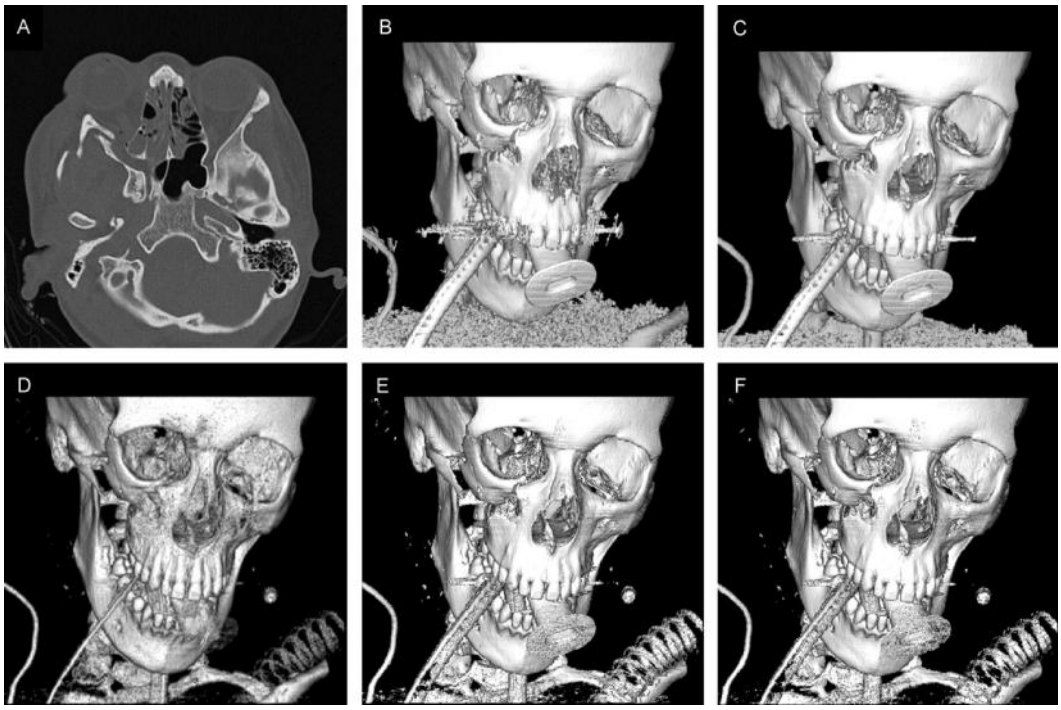
Slika 18. *Orthoslice izgled grudnog koša*

Pomeranje odgovarajuće ravni se vrši pritiskom na taster CTRL+X za pomeranje u X pravcu, CTRL+Y za pomeranje u Y pravcu odnosno CTRL+Z za pomeranje u Z pravcu, držeći pritom i strelicu u zavisnosti u kom smeru želimo da pomerimo ravan.

Ovom metodom se lako prolazi kroz slojeve tela, gde se veoma lako mogu uočiti deformiteti, prelomi, i druga oštećenja kostiju kod pacijenta.

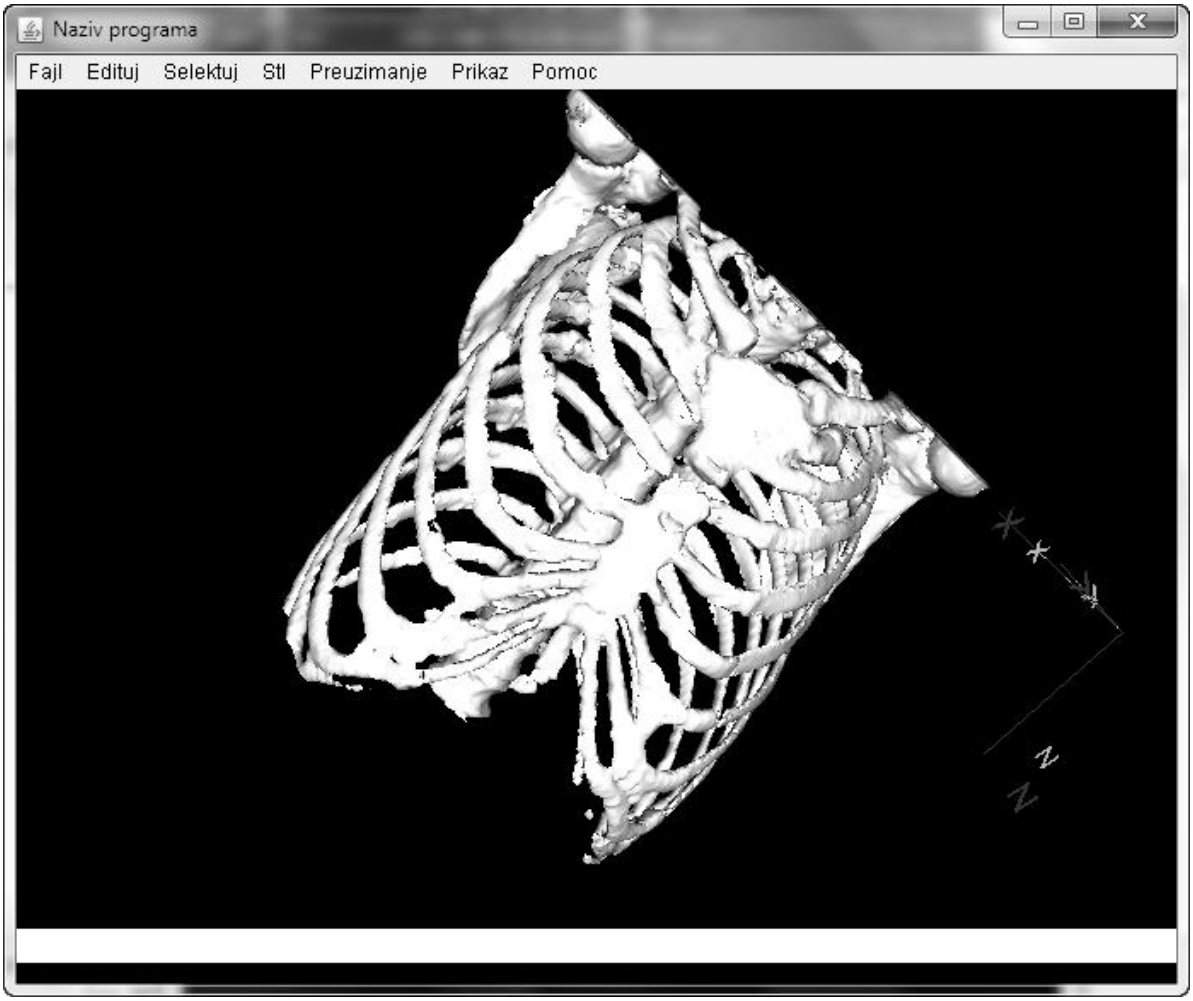
4.4.3. Površinski prikaz

Ukoliko bi za prikaz želeli isključivo kosti, bez zapremine tela, odlučujemo se za opciju "Površina kao 3D". Idealna vrednost za podešavanje thresholda je od 150-165, u zavisnosti od količine bele boje na snimku, ali sa par pokušaja možemo pronaći tu tačnu vrednost. Threshold utiče na kvalitet dobijenog modela, pa je I neophodno da se podesi što tačnije. Za naš primer postavićemo njegovu vrednost na 160.



Slika 19. *Primer korišćenja "Surface rendering" u medicini*

Na slici 24. vidimo 3D model koštanog sistema, dobijenog od niza od 100 slika. Dobijeni snimak je izuzetno visokog kvaliteta, ne samo grafički već i medicinski. Sama kost je prikazana veoma blizu stvarnog izgleda, što je i bio zadatak, da se napravi aplikacija koja će izdvojiti kost sa snimka, i da se što bolje obradi pred 3D štampu.



Slika 20. *Površinski prikaz kosti dobijen korišćenjem aplikacije*

U zavisnosti od memorije, procesora i ostalih računarskih komponenti moguće je učitati mnogo veći broj snimaka, pa samim tim će se i dobiti kompletnija struktura grudnog koša. Ovaj način obrade će biti objašnjen u sledećem poglavlju.

Iz ovog poglavlja možemo zaključiti da je kompletan postupak automatizovan, što je izuzetno bitno kod ovakvih projekata, i uopšteno u medicine zbog same brzine izrade, i pristupa problemu.

4.4. Perfomanse softvera

Ono što je izuetno bitno kod softvera jeste njegova brzina. Brzina pristupa softveru je izuetno velika, dok brzina obrade slike zavisi od broja DICOM snimaka sa kojima se radi. U tabeli 1 su date srednje vrednosti brzine izrade modela u zavisnosti od broja snimaka. Broj snimaka koji može aplikacija da očita je ograničena samo memorijom računara.

Količina DICOM snimaka	Thresholding	Faktor	Brzina izrade 3D modela	Veličina . obj fajla
100	160	4	2, 79 sec	1, 67 mb
200	160	4	4, 19 sec	4, 15 mb
300	160	4	4, 45 sec	4, 22 mb
400	160	4	4, 85 sec	4, 85 mb

Tabela 2. *Brzina generisanja 3D modela*

Veliku ulogu pri brzini plugin-a ima radna memorija računara, odnosno RAM memorija, kao i procesor.

OS	Windows 7
Radni program	ImageJ 1. 45s
RAM	4GB
Procesor	Intel Core i3 CPU M370 2, 40 ghz

Tabela 3. *Konfiguracija PC računara na kom su vršena ispitivanja*

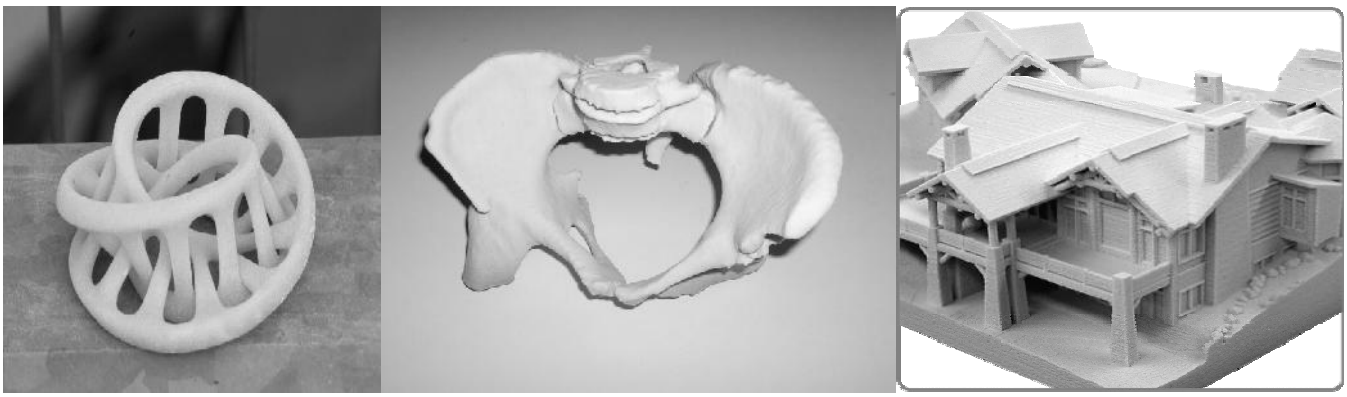
Takođe je bitno napomenuti da brzina zavisi i od veličine ulaznih DICOM snimaka (standardna veličina jednog DICOM snimka je 1 mb), thresholdinga (što je manji thresholding program sporije očitava vrednosti, jer se koristi više bele boje, pa je samim tim potrebno više vremena da se snimak obradi), i faktora (standardna vrednost podešavanja faktora je 4)

5. KREIRANJE FIZIČKOG MODELA KOSTI

U ovom delu je objašnjen postupak kreiranja fizičkog modela kosti na osnovu izradjenog elektronskog modela u plugin-u. Ovaj proces kreiranja modela se sastoji od kreiranja fajla (to je . obj fajl iz Plugin-a), štampanje 3D modela i naknadne obrade. Sam proces se obogaćuje primenom dodatnih tehnika, posebnih vrsta materijala za medicinske svrhe, kao i niz dodatnih softvera za finalnu elektronsku obradu i pripremu za štampu.

5.1. Brza izrada prototipova - "rapid prototyping"

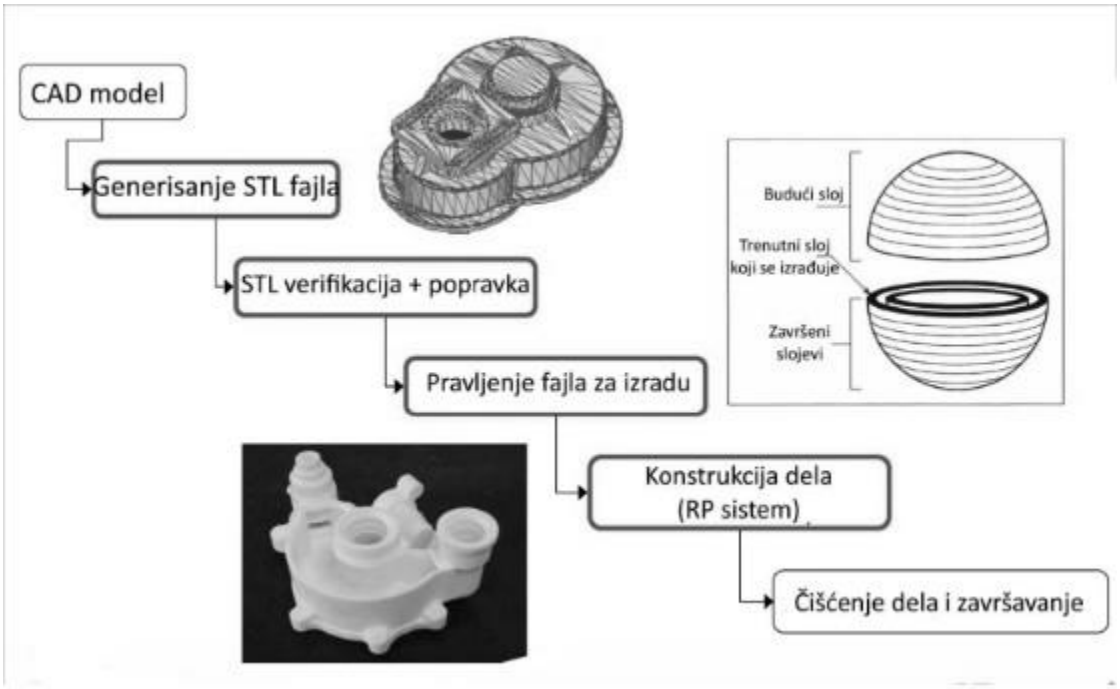
Današnji tempo razvoja novih proizvoda kao i zahtevi globalnog tržišta od poduzetnika zahtevaju brzo prilagodjavanje novim uslovima i neprestano usvajanje novih tehnologija. U poslednjih nekoliko godina tehnologija brze izrade prototipova uz pomoć računara potpomognuto konstruiranjem (CAD) postala je neizostavan inženjerski alat. Smanjenje vremena razvoja proizvoda, poticano stalnim inovacijama i tržišnom borbom, veća složenost proizvoda, potreba za stvarnim modelima u cilju bolje procene oblika, dimenzija, kontrole funkcionalnosti, ergonomije, sve su to komponente koje su uticale na brzi razvoj ove tehnologije i njenu implementaciju u brojna područja inženjerske prakse.



Slika 21. *Primeri primene "Rapid prototyping"* [13]

Ova tehnologija primjenjuje se u raznim aspektima mnogih delatnosti poput arhitekture,dizajna,edukaciji,izradi kalupa,medicini (vizuelizacija CT-a,hemijskih i molekulskih spojeva…)

Pod pojmom 'rapid prototyping' podrazumeva se čitav niz tehnika pomoću kojih se izrađuju trodimenzionalni modeli složenih oblika pomoću CAD sustava i bez uporabe alata.



Slika 22. *RP proces* [13]

Rapid prototyping (brza izrada prototipova): Izrada modela na temelju 3D CAD podataka, računarske tomografije ili na temelju podataka dobijenih digitalizovanjem. Kao materijali mogu se koristiti drvo, polimeri, metali, keramika i kompoziti u raznim oblicima (tečni, prahovi, tanke ploče itd.). Modeli se najčešće izrađuju u svrhu: procene oblika i dimenzija, određivanja funkcionalnosti, ergonomskih studija, izrade uzoraka za kupca, fotografisanje proizvoda u marketinške svrhe, testiranja u zračnim tunelima itd.

Metodologija svih procesa brze izrade prototipova jednaka je i sastoji se od nekoliko faza:

Stvaranje STL (stereolithography) fajla iz 3D CAD podataka: Model se oblikuje pomoću jednog od brojnih CAD softverskih paketa kao što su npr. I-DEAS, Catia, Solidworks, Pro/Engineer itd. Nakon toga sledi konvertovanje u STL format. STL je standardni format koji se koristi za brzu izradu prototipova. On predstavlja trodimenzionalnu površinu koja nastaje kao sklop planarnih trouglova. Upravo zbog planarnosti trouglastih elemenata nemoguće je dobiti tačno zakrivljenu površinu.

Bolja aproksimacija postiže se povećanjem broja trouglova čime se povećava veličina STL fajl, a time se povećava i vrijeme potrebno za preradu. **Stvaranje slojeva**: Zavisno od postupka, laserska ili ink-jet glava počine stvaranje slojeva, od najdonjeg prema najgornjem. Proces skrućivanja materijala ponavlja se za svaki sloj dok se ne dobije gotovi izradak. **Naknadna obrada prototipa**: Nakon stvaranja izratka, s njega se uklanja zaostali, neželjeni materijali, te se brusi ili polira kako bi se dobilo željeno stanje površine

Vreme izrade prototipa zavisi od vrste postupka i najčešće se sastoji od tri dela: priprema filea (od 0, 5 do 2 sata), stvaranje dela (od 0, 5 do 15 sati) i naknadna obrada (od 0, 25 do 6 sati). Sve to znači kako vreme od narudžbine do gotovog prototipa iznosi od 1 do 4 dana.

5.2. RP Tehnologije

U zavisnosti od potrebe, materijala, ekonomskih uslova i dr. odlučujemo se za vrstu RP tehnologije koju ćemo primeniti:

- Stereolitografija
- Modeliranje deponovanjem topljenog materijala (FDM)
- Selektivno lasersko sinterovanje (SLS)
- 3D štampanje
- ThermoJet, LENS, LOM
- Naglo zamrzavanje vode
- TheriForm

Sve ove tehnologije imaju zajednički princip rada: uzimaju geometriju modela iz CAD datoteka, model softverski predočavaju nizom slojeva, šalju informacije za ispis, a štampač umjesto tintom ispisuje slojeve materijala, potpore i veziva- sloj po sloj, sve do finalnog modela. Način rada samih 3D štampača, kao i izbor materijala, razlikuje se od tehnologije do tehnologije.

U nastavku će biti objašnjene najzastupljenije RP tehnologije, poput 3D Štampanja i Stereolitografije [14].

5.2.1. 3D štampanje

"Ink-jet 3D printer" pojam je koji se odnosi na čitavu klasu mašina koji za svoj rad koriste ink-jet tehnologiju konvencionalnih štampača. Prvi takav štampač bio je 3DP (3D Printing) razvijen na američkom univerzitetu MIT, a ZCorpovi 3D štampači jedni su od popularnijih ovakve vrste. Princip rada ove tehnologije vrlo je jednostavan: ink-jet glava selektivno polaže (odnosno štampa) vezivnu tečnost na sloj praha. Na mestima gde tečnost pada na prah, čestice praha se lepe zajedno u krut deo, dok se nepovezani prah oko "polivenih" područja zadržava kao privremena potpora. Platforma s modelom se zatim spušta, dodaje se novi sloj praha i proces se ponavlja.

Slika 23. *Primer monohromatske i kolor 3D štampe* [12]

Slika 24. *3D Štampači (Z Corporation ZPrinter 650- cena oko 40000eur)* [12]

Kada se model dovrši, sav suvišan prah se "izduva" pomoću vazduha pod pritiskom, a model može ići na dodatnu obradu voskom, lepkom i sličnim materijalima koje povećavaju trajnost. Tipična debljina sloja je oko 1 mm, što površinu ovakvih modela čini zrnastom i neuglednom, a ni preciznost dobijenih modela nije previše velika. Ipak, dobra strana 3D štampača jeste brzina modeliranja i relativno mali trošak rada. Međutim, postoje i neke varijacije na ovu temu koje nude veću preciznost, poput Thermo-Jet ili Multi-Jet štampača koji koriste linearnu mrežu ispisnih glava. Vrlo precizni su i Model Maker ink-jet štampači koji koriste dva dela i to jedan za polaganje termoplastike kao gradevnog materijala, a drugi za polaganje voska kao potpornog materijala. Svaki sloj se izravnava pomoću alata za rezanje pre ponavljanja procesa za sljedeći sloj. Model Maker štampači su zbog ove dodatne dimenzije obrade i preciznosti pogodni i za korišćenje npr. u industriji plastičnog nakita i drugo [12].

5.2.2. Stereolitografija (SLA)

Stereolitografija je prva metoda brze izrade prototipova koja je razvijena 1986. godine u softveru 3D Systems iz SAD-a. Prototip se dobija na temelju CAD modela koji je podijeljen na slojeve. Kao materijali se rkoriste različite vrste fotopolimera u tečnom stanju koji su osetljivi na ultraljubičasto zračenje. Do skrućivanja slojeva dolazi pod uticajem laserskog snopa. Deo koji se izrađuje nalazi se na platformi. Nakon što je pod utjecajem laserskih zraka došlo do skrućivanja sloja, platforma se pomiče za 0, 1 mm prema dole nakon čega sledi skrućivanje drugog sloja. Nakon što su svi slojevi skrutnuti, prototip je gotov. Nakon toga se platforma izdiže iznad tekućeg fotopolimera i čeka se ceđenje preostalog tekućeg materijala.

Proizvod se zatim čisti i, ako je potrebno, naknadno obrađuje kako bi se dobila površina odgovarajućeg stanja. Iz samog opisa procesa je vidljivo kako će gotovi proizvod, zbog slojevite strukture, imati stepeničasti oblik. Što je veća visina pomicanja platforme, taj je oblik vidljiviji. On se može smanjiti različitom orijentacijom dijela tijekom proizvodnje. Ako se dulja os modela orijentira vertikalno, dulje je vrijeme izrade, ali su i manje izražene stepenice [13].

Slika 25. *Mašina za stereolitografiju* [13]

Prednosti stereolitografije jesu:

- visoka rezolucija,
- moguća izrada dvobojnih prototipova,
- nema geometrijskih ograničenja oblika,
- potpuna automatiziranost procesa.

Nedostaci jesu:

- ograničen broj upotrebljivih materijala (samo fotopolimeri),
- prototip slabijih mehaničkih svojstava,
- naknadna obrada prototipa UV očvršćivanjem u peći,
- pri izradi su potrebni potporinji prototipa koje treba ukloniti,
- stezanje fotopolimera pri očvršćivanju može uzrokovati pojavu naprezanja i deformacija,
- fotopolimer je otrovan u tekućem stanju.

Opisani postupci nalaze svoje mesto primene i u medicini. RP medicinski modeli su našli primenu u planiranju kompleksnih hirurških intervencija, simulaciji, projektovanju i izradi implanata, medicinskih uredaja, hirurških pomocnih alata, kao i u inženjeringu tkiva [13].

5.3. STL fajl

Da bi se master model mogao izraditi na 3D štampacu, potrebno je generisati STL fajl na osnovu CAD modela.

Različiti CAD paketi koriste brojne različite algoritme kojima se predstavljaju čvrsti predmeti. Radi doslednosti, STL format je usvojen kao standard u industriji brze izrade prototpipova. Drugi korak je konvertovanje CAD fajla u STL format. Ovaj format predstavlja 3D površinu kao sklop dvodimenzionalnih trouglova"nalik brušenih površinama isečenog dragog kamena. U fajlu su sadržane koordinate najviših tačaka i smer spoljne normale svakog trougla. Zbog toga što STL fajlovi koriste dvodimenzionalne elemente, oni ne mogu potpuno tačno da predstave zakrivljene površine. Povećanje broja trouglova povećava tačnost, ali je zbog toa fajl veći. Većim, komplikovanim fajlovima potrebno je više vremena za rad na pred-obradi i izradi, tako da dizajner treba da uskladi zadovoljavajuću tačnost sa fajlom kojim se može lako raditi, kako bi napravio koristan STL fajl. Budući da je .stl format univerzalan , ovaj proces je identičan za sve RP tehnike izrade.

Program za pred-obradu priprema STL fajl koji će se raditi. Na raspolaganju je nekoliko programa i većina njih korisniku omogućava da podesi veličinu, poziciju i orijentaciju modela. Orijentacija izrade je važna iz nekoliko razloga. Prvo, svojstva brze izrade prototipova se može razlikovati u zavisnosti od smera koordinate. Na primer, prototipovi su obično slabiji i manje precizni na z(vertikalnom) smeru, nego u x-y ravni. Pored toga, orijentacija dela delimično uslovljava vremenski period koji će biti potreban za izradu modela. Postavljanje najkraće dimenzije u z smer, smanjuje broj slojeva, što rezultira skraćenjem vremena izrade. Softver za ppred-obradu seće STL model na više slojeva debljine izmedju 0, 01mm i 0, 7mm, u zavisnosti od tehnike izrade. Podrške su korisne za izuzetno osetljiva svojstva, kao što su kontrakonusi, unutrašnje šupljine i odeljci sa tankim zidovima. Svaki proizvođač RP mašine obezbeđuje sopstveni softver za pred-obradu.

6. Zaključak

Sam razvoj softvera za generisanje 3D modela na osnovu 2D snimaka DICOM formata predstavlja veliki izazov. Metode i tehnike su znatno napredovale u odnosu na prethodnu deceniju, ali sam razvoj ovih tehnologija i procesa i danas traje. Hardver u današnje vreme konstantno napreduje, te se od softvera takodje iziskuje izuzetno velika snaga, a programeri se trude da iskoriste maksimalnu snagu hardvera radi implementacije što jačih, pouzdanijih, korisnijih programa.

Među softverskim paketima slične namene aplikacije prerađene u ovom seminarskom se izdvajaju Mimics (komercijalni softver), ImageJ3DViewer(plugin za ImageJ), Visualization Toolkit(softverska biblioteka) koju koriste ostali komercijalni ili razvojni softveri poput ParaView, VisIt, Slicer itd.

Veliki plus aplikacije izradjene u seminarskom radu se može navesti izuzetna tačnost(vernost predstavljanja) izlaznog 3D modela. Softver omogućava pristup besplatnim aplikacijama za konvertovanje . obj u neki od drugih popularnih fajl formata, poput . stl, catPart itd.

Kao osnovna mana se može navesti u zavisnosti od broja slika koje se koriste, veličina izlaznog . obj fajla. Međutim ta veličina se može i zanemariti imajući u vidu koliko je napredovala hardverska tehnologija računara u odnosu na 2009. godinu, a samim tim i napredovanje i poboljšanje ostalih 3D softvera gde se dati 3D model i može implementirati i kasnije obrađivati u zavisnosti od potrebe modela.

Kao moguća nadogradnja softvera u budućnosti može biti ugradnja konvertera . obj fajla u neki drugi format direktno u aplikaciju, zatim generisanje više od 1000 slika itd.

Bez obzira na izlazni kvalitet 3D modela(koji je inače na visokom nivou), vrlo lako se taj model može obraditi, poboljšati u drugom alatu za 3D obradu modela.

Za pohvalu ovog softvera se može i navesti to da se izlazni fajl može direktno slati na 3D štampanje, i time recimo spasiti nekome život (primer izrade modela kosti i impelemntacija u telo pacijenta).

7. Literatura

[1] http://bs. wikipedia. org/wiki/Digitalna_slika

[2] http://automatika. etf. bg. ac. rs/files/predmeti/os4dos/lekcija_1_-_osnove_digitalne_slike. pdf

[3] http://elib. mi. sanu. ac. rs/files/journals/ncd/2/d003download. pdf

[4] http://bs. wikipedia. org/wiki/Kompjuterizirana_tomografija

[5] http://dicom. rubyforge. org/content/tutorial2-1. jpg

[6] http://www. telemed. co. rs/dicom-standard

[7] http://medical. nema. org/

[8] http://www. tera. hr/index. php?option=com_content&view=article&id=5&Itemid=6

[9] http://www. neurofly. de/

[10] http://rsbweb. nih. gov/ij/

[11] http://www. stetoskop. info/Kompjuterizovana-tomografija-CT-Skener-507-c13-content. htm

[12] http://www. acmi. be/images/3Dprint

[13] http://www. gradimo. hr/clanak/brza-izrada-prototipova/15509

[14] http://www. ctc. kg. ac. rs

[15] Writing ImageJ Plugins- A Tutorial, Werner Bailer, New Jersey, 2005

[16] http://sh. wikipedia. org/wiki/Vektorska_grafika

[17] http://sh. wikipedia. org/wiki/Rasterska_grafika