UDŽBENIK ELEKTROTEHNIČKOG FAKULTETA U BEOGRADU

Milica M. Janković

ANALIZA BIOMEDICINSKE SLIKE

Beograd, 2025.

dr Milica Janković, vanredni profesor
Univerzitet u Beogradu – Elektrotehnički fakultet

ANALIZA BIOMEDICINSKE SLIKE

elektronski udžbenik Elektrotehničkog fakulteta u Beogradu

Recenzenti:
dr Veljko Papić, vanredni profesor
dr Ana Gavrovska, vanredni profesor

Nastavno-naučnog veće Elektrotehničkog fakulteta Univerziteta u Beogradu odobrilo je objavljivanje ovog udžbenika odlukom broj 272/15 od 11.02.2025. godine.

Izdavač:
Univerzitet u Beogradu – Elektrotehnički fakultet
Bulevar kralja Aleksandra 73, 11120 Beograd, Srbija

Štampa:
Milica Janković, Beograd, 2025

Tiraž:
50 primeraka (CD)

ISBN: 978-86-7225-099-2

Mojoj porodici

SADRŽAJ

PREDGOVOR

„Analiza biomedicinske slike“ je udžbenik prevashodno namenjen studentima četvrte godine osnovnih akademskih studija Elektrotehničkog fakulteta Univerziteta u Beogradu koji prate izborni kurs pod istoimenim nazivom „Analiza biomedicinske slike“ (šifra predmeta 13E054ABS), ali i drugima koji žele da steknu osnovna znanja i praktična iskustva u oblasti analize biomedicinske slike. Očekuje se da čitaoci imaju predznanja u domenu osnovne obrade i predstavljanja digitalnih signala.

U uvodnom delu knjige je opisana priroda biomedicinskih slika sa tačke gledišta fizičkih procesa koji određuju način njihove akvizicije. Predstavljena je podela modaliteta biomedicinskog slikanja i pregled njihovih osnovnih primena.

Udžbenik sadrži dosta praktičnih primera napisanih i testiranih u programskom jeziku *Python*. U drugom poglavlju je opisano *Spyder* razvojno okruženja i osnovna pravila sintakse za *Python* programiranje kako bi se obezbedilo razumevanje programskih kôdova predstavljenih u nastavku udžbenika. Svi programski kôdovi su dostupni na oficijalnoj stranici predmeta „Analiza biomedicinske slike” u okviru veb-sajta Katedre za Signale i sisteme Elektrotehničkog fakulteta Univerziteta u Beogradu[1]. Slike koje se koriste za analizu većinom se nalaze u okviru javno dostupnih repozitorijuma čiji linkovi za preuzimanje su navedeni u tekstu, a preostale slike su dostupne zajedno sa programskim kôdovima.

Prvi korak svake analize biomedicinske slike je učitavanje podataka iz datoteka specijalizovanih formata. Stoga je treće poglavlje namenjeno pregledu različitih načina za učitavanje sadržaja datoteka, eksplorativnoj analizi sadržaja datoteke, osnovnom prikazu tog sadržaja, kao i upisu slika u datoteku.

Četvrto poglavlje je namenjeno poboljšanju biomedicinskih slika sa stanovišta prostora intenziteta. U ovom poglavlju je akcenat stavljen na različite vidove transformacije intenziteta, s posebnim osvrtom na efektnu i često korišćenu ekvalizaciju histograma.

Metode filtriranja biomedicinskih slika u frekvencijskom i prostornom domenu su predstavljene u petom poglavlju čime se završava celina koja se odnosi na predobradu biomedicinske slike.

[1] https://automatika.etf.bg.ac.rs/

Savremena dostignuća u oblasti biomedicinskog slikanja se većinski odnose na oblasti segmentacije (izdvajanja regiona od interesa), registracije (poravnavanje slika prikupljenih sa različitih modaliteta slikanja za istog ili različite pacijente, prikupljenih sa istog modaliteta ali u različitim trenucima i sl.), vizuelizacije (primena računarske grafike za prikaz volumena) i primene metoda mašinskog učenja (tradicionalna i duboka radiomika). Teorijske osnove tehnika segmentacije su ilustrovane na primerima u šestom poglavlju. Odabrana statistička obeležja teksture koja su od značaja za tradicionalnu radiomiku su predstavljena u sedmom poglavlju. Osmo poglavlje je namenjeno opisu različitih metoda prostornih transformacija i registracije. Osnove vizuelizacije (primeri zapreminskog i površinskog renderovanja) su prikazane u devetom poglavlju. Poslednje, deseto poglavlje, posvećeno je uvodu u danas vrlo atraktivnu oblasti primene mašinskog učenja na biomedicinsko slikanje.

Niz kolega, lekara i studenta je posredno doprineo koncipiranju i stvaranju ovog nastavnog materijala. Autorka se zahvaljuje doc. Novici Jankoviću, dr Ivanu Vajsu, prof. Jeleni Ćertić, as. Mariji Novičić, prof. Milošu Vujisiću, Milici Badži Atanasijević, dr Vojislavu Antiću, dr Marku Barjaktaroviću, prof. Dragomiru El Mezeniju, dr Ani Koljević Marković, dr Tijani Radović, prof. dr Slobodanki Beatović, doc. dr Mariji Radulović, doc. dr Otašu Durutoviću, prof. dr Milovanu Matoviću, doc. dr Isidori Gvozdić, dr Mariji Jovanović, Dimitriju Šarcu, prof. dr Ani Starčević i dr Mariji Jeremić na otvorenosti i kritičnosti u razmatranju različitih praktičnih aspekata biomedicinskog slikanja.

Autorka se posebno zahvaljuje i recenzentima udžbenika, dr Veljku Papiću, vanrednom profesoru Elektrotehničkog fakulteta Univerziteta u Beogradu i dr Ani Gavrovskoj, vanrednom profesoru Elektrotehničkog fakulteta Univerziteta u Beogradu na pomoći i podršci u završnoj fazi pisanja ovog udžbenika.

Beograd, 2025. *Milica Janković*

1. PRIRODA BIOMEDICINSKIH SLIKA

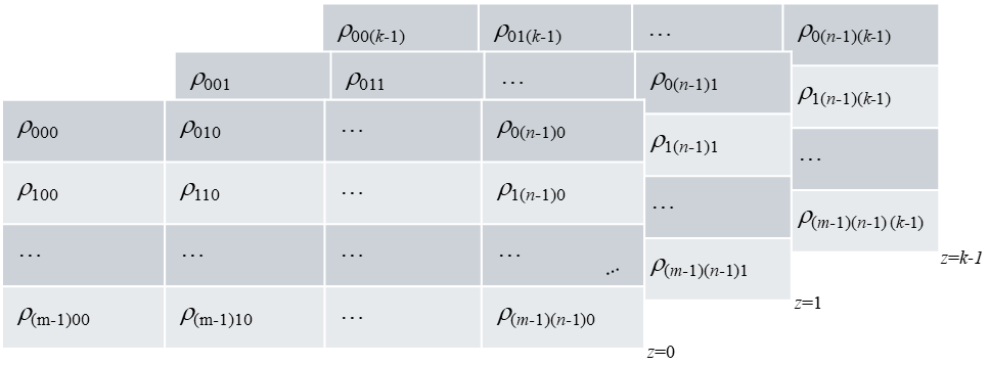
Biomedicinska slika je digitalna ili analogna vizuelizacija unutrašnjih struktura ili funkcionalnih procesa u ljudskom organizmu i drugim biološkim sistemima, koja se koristi u medicinskoj dijagnostici, istraživanjima i/ili terapiji. Danas se analogne biomedicinske slike koriste vrlo retko, jer su digitalne tehnologije skoro potpuno zamenile analogne metode u biomedicinskom slikanju. Stoga će se u ovoj knjizi termin biomedicinska slika odnositi isključivo na digitalne vizuelizacije morfologije i funkcije bioloških sistema. Ove slike su kreirane specifičnim tehnikama biomedicinskog slikanja koje za cilj imaju da detektuju i vizuelizuju raznovrsne efekte zračenja u interakciji sa biološkim sistemima ili pak zračenja koja biološki sistemi proizvode.

Matematički gledano, svaka digitalna slika uključujući i biomedicinske slike, se može zapisati u obliku matrice numeričkih vrednosti I, dimenzije m x n, čiji svaki element $I(x,y)$ predstavlja piksel intenziteta ρ_{xy}, ($x \in \{0, 1, \ldots, m-1\}$, $y \in \{0, 1, \ldots, n-1\}$):

$$I = \begin{bmatrix} \rho_{00} & \rho_{01} & \cdots & \rho_{0(n-1)} \\ \rho_{10} & \rho_{11} & \cdots & \rho_{1(n-1)} \\ \cdots & \cdots & \cdots & \cdots \\ \rho_{(m-1)0} & \rho_{(m-1)1} & \cdots & \rho_{(m-1)(n-1)} \end{bmatrix}. \tag{1.1}$$

S obzirom na to da su digitalne slike u matematičkoj predstavi zapravo matrice, na njih se mogu primenjivati odgovarajuće algebarske operacije: sabiranje matrica, oduzimanje matrica, množenje matrica, množenje matrice skalarom, sabiranje matrice sa skalarom i sl., u zavisnosti od toga koji je cilj analize biomedicinske slike koja je primenjena (npr. poboljšanje kvaliteta slike, otklanjanje šuma, detektovanje segmenata od interesa, prostorna transformacija…).

U analizi biomedicinske slike, često se posmatra ne samo 2D prikaz tj. digitalna slika, već se posmatra diskretni trodimenzioni prostor (volumen) kroz 3D matričnu predstavu I dimenzije m x n x k, čiji svaki element $I(x,y,z)$ predstavlja voksel intenziteta ρ_{xyz}, ($x \in \{0, 1, \ldots, m-1\}$, $y \in \{0, 1, \ldots, n-1\}$, $z \in \{0, 1, \ldots, k-1\}$), Sl. 1.1. U praktičnom smislu, volumen se može sagledati kao niz digitalnih slika poređanih jedna iza druge.



Slika 1.1. Predstava volumena u matričnom obliku

Dodatno, moguće je posmatrati i vremensku dinamiku promena bilo na biomedicinskoj slici bilo na celom volumenu, tj. u matematičkoj predstavi se mogu posmatrati $I(x,y,t)$ i $I(x,y,z,t)$, respektivno, gde t predstavlja diskretnu vremensku komponentu.

Svaki piksel na biomedicinskoj slici ili voksel ukoliko se posmatra ceo volumen, je reprezent jednog od sledećih pet efekata elektromagnetnog zračenja (3 Hz-30 ZHz), ultrazvučnih talasa (2-15 MHz) ili elektronskih zraka (~10^{19} Hz), Sl. 1.2:

- efekat spoljašnjeg elektromagnetnog zračenja primenjenog na biološke sisteme,
- efekat spoljašnjih ultrazvučnih talasa primenjenih na biološki sistem,
- efekat elektromagnetnog zračenja unetog unutar biološkog sistema,
- efekat zračenja koje potiče od samog biološkog sistema,
- efekat zračenja od pobuđenih struktura u biološkom sistemu.

U slučaju kada je piksel/voksel reprezent efekta spoljašnjeg elektromagnetnog zračenja primenjenog na biološke sisteme (Sl. 1.2A), detektor efekata zračenja se postavlja van biološkog sistema za vizuelizaciju („transmisioni“ pristup). Tehnike biomedicinskog slikanja koje funkcionišu po „transmisionom“ pristupu obuhvataju sledeće osnovne kategorije spram vrste spoljašnjeg zračenja koje se primenjuje:

- tehnike biomedicinskog slikanja koje koriste X-zrake (rendgenske zrake) za ravansko (projekciono) snimanje: rendgenografija (detekcija X-zraka na rendgen filmu), kompjuterizovana radiografija (detekcija X-zraka na fotostimulabilnoj fosfornoj ploči), digitalna radiologija (detekcija X-zraka flat-panel detektorima koji direktno konvertuju X-zrake u digitalne signale).

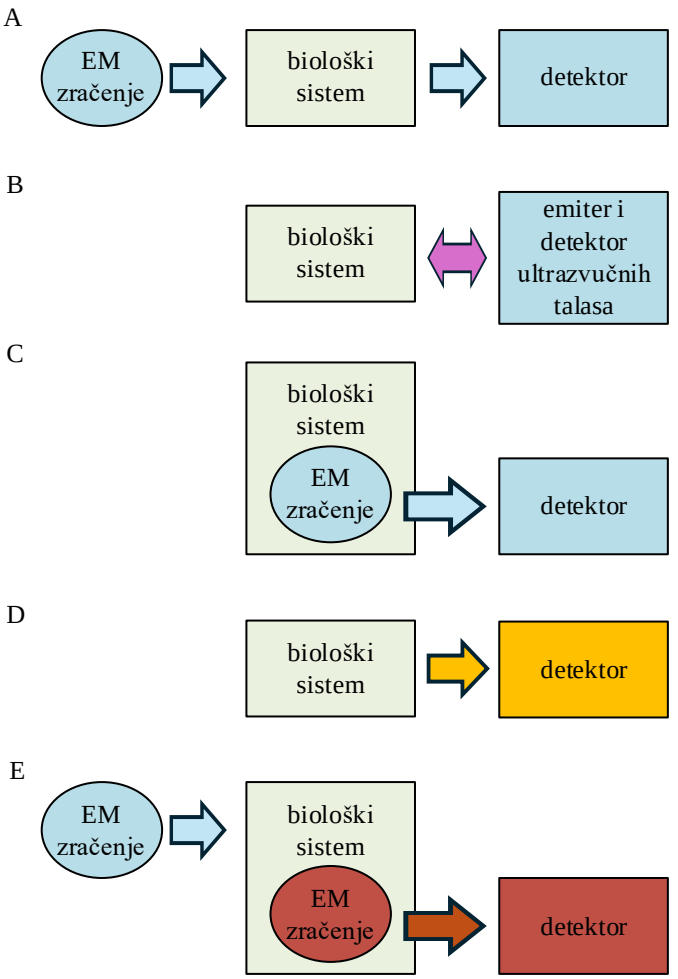
- tehnika biomedicinskog slikanja koja koristi X-zrake za tomografsko[2] snimanje: kompjuterizovana tomografija (eng. *Computerized Tomography, CT*)
- tehnike biomedicinskog slikanja koje koriste vidljivu svetlost kao izvor zračenja: dermatološka fotografija (slikanje kože), endoskopija (osvetljavanje i snimanje unutrašnjosti biološkog sistema) i optička mikroskopija (detekcija svetlosti koja prolazi kroz biološki uzorak ili se odbija od njega)
- tehnika biomedicinskog slikanja koja koristi elektronski snop kao izvor zračenja: elektronska mikroskopija (detekcija interakcije elektrona sa biološkim uzorkom)
- tehnike biomedicinskog slikanja koje koriste zračenje blisko infracrvenom zračenju: spektroskopija u delu spektra bliskom infracrvenom zračenju (eng. *Near-Infrared Spectroscopy, NIRS* – detekcija apsorpcije svetlosti u bliskom infracrvenom spektru) i optička koherentna tomografija (eng. *Optical Coherence Tomography, OCT* – detekcija reflektovane svetlosti iz različitih slojeva tkiva)
- tehnika biomedicinskog slikanja koja koristi mikrotalasno zračenje: mikrotalasna tomografija (detektuje se permitivnost i provodljivost u biološkom sistemu kao i apsorpcija mikrotalasa)

U slučaju kada je piksel/voksel reprezent efekta spoljašnjih ultrazvučnih talasa (Sl. 1.2B), spolja se na biološke sisteme putem sonde emituju ultrazvučni talasi koji se reflektuju od unutrašnjih struktura bioloških sistema, a potom sonda detektuje reflektovane talase na osnovu čega se vrši vizuelizacija unutrašnjosti biološkog sistema. Ultrasonografija je primer tehnike biomedicinskog slikanja koja koristi ultrazvučne talase.

Za kategoriju biomedicinskog slikanja kod koje je piksel/voksel reprezent efekta elektromagnetnog zračenja unetog unutar biološkog sistema (Sl. 1.2C), u unutrašnjost biološkog sistema se unose jedinjenja koja emituju gama zračenje ili koja u interakciji sa biološkim sistemom proizvode gama zračenje, a detektor efekata zračenja se postavlja izvan biološkog sistema („emisioni“ pristup). Tehnike biomedicinskog slikanja koje funkcionišu po „emisionom“ pristupu obuhvataju sledeće osnovne kategorije spram vrste unetog zračenja koje se primenjuje:

[2] Tomografsko snimanje je slikanje iz više uglova kako bi se metodama rekonstrukcije (npr. Radonovom transformacijom, metodom povratne projekcije (eng. *Filtered Back Projection*), iterativnim metodama rekonstrukcije itd.) dobili snimci slojeva tj. slajseva (eng. *slice*), odnosno trodimenzionalan prikaz volumena.

- tehnike biomedicinskog slikanja koje koriste gama zračenje za ravansko (projekciono) snimanje: scintigrafija
- tehnike biomedicinskog slikanja koje koriste gama zračenje za tomografsko snimanje: jednofotonska emisiona kompjuterska tomografija (eng. *Single-photon emission computed tomography, SPECT*) i pozitronska emisiona tomografija (eng. *Positron Emission Tomography, PET*).



Slika 1.2. Biomedicinsko slikanje predstavlja vizuelizaciju efekata elektromagnetnog zračenja (EM), ultrazvučnih talasa ili elektronskih zraka: A) spoljašnje elektromagnetno zračenje deluje na biološki sistem i detektuje se zračenje koje „prođe“ kroz biološki sistem – „transmisioni pristup“, B) sonda emituje ultrazvučne talase i detektuje reflektovane talase od unutrašnjosti biološkog sistema, C) zračenje koje je uneto u biološki sistemi „izlazi“ iz njega i detektuje se – „emisioni pristup“, D) sam biološki sistem emituje zračenje koje se detektuje, E) detektuje se zračenje struktura biološkog uzorka pobuđenih spoljašnjim zračenjem

U slučaju kada je piksel/voksel reprezent efekta zračenja koje potiče od samog biološkog sistema (Sl. 1.2D), pomoću odgovarajućeg detektora se snima i vizuelizuje ovakvo zračenje koje se emituje od strane biološkog sistema. Termografija je primer tehnike biomedicinskog slikanja koje koristi infracrveno zračenje koje se emituje sa površine kože za vizuelizaciju toplote koju emituje telo.

Za kategoriju biomedicinskog slikanja kod koje je piksel/voksel reprezent efekta zračenja od pobuđenih struktura u biološkom sistemu (Sl. 1.2E), detektor akvizicionog sistema snima i vizuelizuje ovakvo zračenje. Primeri ovakvih metoda slikanja su: slikanje magnetnom rezonancom (eng. *Magnetic Resonance Imaging*, *MRI*) i fluorescentna mikroskopija. Kod metode *MRI*-a magnetno polje i radio talasi pobuđuju jezgra atoma što izaziva emitovanje radio talasa koji učestvuju u stvaranju biomedicinske slike. Fluorescentna mikroskopija koristi ultraljubičasto zračenje za pobuđivanje fluorescentnih molekula u biološkom uzorku koji potom emituju zračenje na većoj talasnoj dužini.

Tabela 1.1. Frekvencije, talasne dužine i energije fotona u spektru elektromagnetnog zračenja koje se primenjuje u biomedicinskom slikanju (1 p=10^{-12}, 1 f=10^{-15}, 1 T=10^{12}, 1 P=10^{15}, 1 E=10^{18}, 1 Z=10^{21})

ZRAČENJE	TALASNA DUŽINA	FREKVENCIJA	ENERGIJA FOTONA	PRIMER PRIMENE U BIOMEDICINSKOM SLIKANJU
Radio talasi	0.1 m – 100000 km	3 Hz – 3 GHz	12.4 feV – 12.4 μeV	*MRI*
Mikrotalasi	1 mm – 0.3 m	1 GHz – 300 GHz	4.14 μeV– 1.24 meV	mikrotalasna tomografija
Infracrveno zračenje	750 nm – 1 mm	300 GHz – 400 THz	1.24 meV – 1.654 eV	termografija
Zračenje blisko infracrvenom	750 nm – 2.5 μm	120 THz – 400 THz	496 meV – 1.654 eV	*NIRS*, *OCT*
Vidljiva svetlost	400 nm – 700 nm	430 THz – 750 THz	1.78 eV – 3.1 eV	dermatološka fotografija, endoskopija, optička mikroskopija
Ultraljubičasto zračenje	10 nm – 400 nm	750 THz – 30 PHz	3.1 eV – 124 eV	fluorescentna mikroskopija
X-zraci (rendgenski zraci)	10 pm – 1 nm	300 PHz – 30 EHz	1.24 keV – 124 keV	rendgenografija, kompjuterizovana radiografija, digitalna radiologija, *CT*
Gama zraci	10 fm – 100 pm	3 EHz – 30 ZHz	12.4 keV – 124 MeV	scintigrafija *SPECT* *PET*

U Tabeli 1.1 je dat sistematizovan pregled zračenja u elektromagnetnom spektru koja se koriste za biomedicinsko slikanje sa uporednim prikazom odgovarajućih talasnih dužina, frekvencija i energija fotona, a takođe su dati i primeri gde se primenjuju u gore pomenutim tehnikama biomedicinskog slikanja.

Postoje i hibridni pristupi biomedicinskog slikanja koji svojim multimodalnim pristupom omogućavaju kombinovanje (registrovanje) biomedicinskih slika dobijenih različitim tehnikama sa ciljem dobijanja potpunije informacije o strukturi i/ili funkciji posmatranog biološkog sistema. Primeri kliničkih uređaja koji su multimodalni i kombinuju različite tehnike formiranja medicinske slike su *PET/CT* i *PET/MRI* uređaji. Radi ispitivanja funkcije, pojedini modaliteti omogućavaju praćenje promena na biomedicinskoj slici u vremenu (funkcionalni *MRI – fMRI*, dinamska scintigrafija, dinamski *SPECT*, dinamski *PET*, *NIRS*, *OCT*, fluorescentna mikroskopija, ultrasonografija).

U Tabeli 1.2, za osnovne tehnike biomedicinskog slikanja je navedena priroda intenziteta piksela spram detekcije fizičkih fenomena, kao i dominantna praktična primena.

Tabela 1.2. Priroda biomedicinskog slikanja i primeri primene

TEHNIKA	PRIRODA INTENZITETA PIKSELA	DOMINANTNA PRIMENA
Digitalna radiologija	Srazmeran količini X-zračenja koje je biološki sistem apsorbovao (gusti materijali kao npr. kosti apsorbuju više zračenja i pojavljuju se kao svetliji pikseli na slici)	Dijagnostika preloma i povreda, praćenje bolesti pluća
CT		Dijagnostika i praćenje tumora, procena unutrašnjih povreda i krvarenja
Dermatološka fotografija	Srazmeran osvetljenju i refleksiji svetlosti sa kože	Praćenje promena na koži (mladeži, pigmentacija i sl.), dijagnostika kožnih bolesti
Endoskopija	Srazmeran osvetljenju i refleksiji svetlosti sa unutrašnjih površina tela	Dijagnostika i procena unutrašnjih organa, minimalno invazivna hirurgija
Optička mikroskopija	Srazmeran količini svetlosti koja prolazi kroz uzorak i stiže do detektora (delovi biološkog uzorka koji propuštaju više svetlosti, npr. tanak ili proziran deo, se pojavljuju kao svetliji pikseli na slici)	Proučavanje ćelija, tkiva, mikroorganizama i drugih bioloških struktura

Tabela 1.2. (nastavak) Priroda biomedicinskog slikanja i primeri primene

TEHNIKA	PRIRODA INTENZITETA PIKSELA	DOMINANTNA PRIMENA
Elektronska mikroskopija	Srazmeran interakciji elektronskog snopa sa uzorkom (delovi uzorka koji generišu više sekundarnih elektrona ili reflektovanih elektrona se pojavljuju kao svetliji pikseli na slici)	Analiza nanostruktura
NIRS	Srazmeran apsorpciji talasa iz spektra bliskog infracrvenom spektru u biološkim sistemima (delovi sa većom refleksijom tj. manjom apsorpcijom svetlosti, usled niže koncentracije apsorbujućih molekula, npr. deoksigenisanog hemoglobina, odgovaraju svetlijim pikselima slike)	Praćenje cerebralne oksigenacije u realnom vremenu (novorođenčad, operaciona sala), funkcionalno mapiranje mozga
OCT	Srazmeran stepenu refleksije ili raspršenja svetlosti unutar biološkog sistema (delovi sa visokom refleksijom ili raspršenjem, npr. granice između različitih slojeva tkiva, odgovaraju svetlijim pikselima slike)	Dijagnostika i praćenje očnih bolesti kao što su glaukom, dijabetička retinopatija
Mikrotalasna tomografija	Srazmeran dielektričnim svojstvima tkiva (permitivnost i provodljivost) koje utiču na apsorpciju i raspršenje mikrotalasnog zračenja	Detekcija tumora dojke
Ultrasonografija	Srazmeran reflektovanim ultrazvučnim talasima od granica između različitih bioloških struktura (delovima gde postoji jaka refleksija ultrazvučnih talasa odgovaraju svetliji pikseli slike, npr. na granici kost-meko tkivo koji imaju različita akustična svojstva)	Praćenje trudnoće i razvoja fetusa, procena funkcije srca (srčane komore, srčani zalisci, protok krvi), abdominalna dijagnostika
Scintigrafija	Srazmeran koncentraciji unesenog izvora zračenja (u obliku radiofarmaceutika) u biološkim sistemima (visokoj koncentraciji radiofarmaceutika odgovaraju svetliji pikseli slike, npr. na mestima povećane funkcionalne aktivnosti ili abnormalnosti, kao što su tumori ili upale)	Detekciju i praćenje tumora, miokardijalna perfuzija, procena moždane funkcije
SPECT		
PET	Srazmeran broju gama fotona koji nastaju kada pozitronski emitovani radiofarmaceutik intereaguje sa elektronima u biološkom sistemu. (visokoj koncentraciji radiofarmaceutika odgovaraju svetliji pikseli slike, npr. na mestima povećane funkcionalne aktivnosti ili abnormalnosti, kao što su tumori ili upale)	

Tabela 1.2. (nastavak) Priroda biomedicinskog slikanja i primeri primene

TEHNIKA	PRIRODA INTENZITETA PIKSELA	DOMINANTNA PRIMENA
Termografija	Srazmeran temperaturi bioloških struktura (svetliji pikseli odgovaraju delovima sa višom temperaturom, npr. deo na kome je upala, povećana cirkulacija i sl.)	Detekcija i praćenje upalnih procesa, problema u cirkulaciji, detekcija tumora
MRI	Srazmeran relativnoj količini vodonikovih atoma u tkivu i njihovom odgovoru na radiofrekventne impulse u prisustvu jakog magnetnog polja (svetliji pikseli odgovaraju delovima sa velikom količinom vodonikovih atoma, npr. cerebrospinalna tečnost ili upaljena tkiva)	Detekcija tumora mozga, dijagnostika neuroloških poremećaja, dijagnostika povreda i degenerativnih bolesti kostiju i zglobova, procena srčane funkcije
Fluorescentna mikroskopija	Srazmeran intenzitetu fluorescencije koju emituje uzorak kada je izložen svetlu određene talasne dužine (svetliji pikseli odgovaraju delovima sa visokom koncentracijom fluorescentnog markera koji se koristi za označavanje specifičnih biomolekula ili struktura unutar uzorka)	Dijagnostika i praćenje raka ili infekcija, farmaceutska istraživanja

U okviru ovog poglavlja su predstavljeni osnovni koncepti načina formiranja biomedicinskih slika, neophodnih čitaocima kako bi ispratili naredna poglavlja koja imaju fokus na analizu formiranih biomedicinskih slika. Za detaljnije informisanje o fizici biomedicinskog slikanja preporučena literatura je [Webb 2022] i [Bushberg et al. 2020].

1.1. Rezime poglavlja

- Svaka digitalna slika uključujući i biomedicinske slike, se može zapisati u obliku matrice numeričkih vrednosti čiji svaki element predstavlja tzv. piksel slike. Takođe, i volumen (trodimenzioni prostor) može da se prikaže kroz 3D matričnu predstavu numeričkih vrednosti čiji svaki element predstavlja tzv. voksel volumena.
- Svaki piksel na biomedicinskoj slici ili voksel ukoliko se posmatra ceo volumen, je reprezent jednog od sledećih pet efekata elektromagnetnog zračenja, ultrazvučnih talasa ili elektronskih zraka: 1) efekta spoljašnjeg elektromagnetnog zračenja primenjenog na biološke sisteme, 2) efekta spoljašnjih ultrazvučnih talasa, 3) efekta elektromagnetnog zračenja unetog unutar biološkog sistema, 4) efekta zračenja koje potiče od samog biološkog sistema i 5) efekta zračenja od pobuđenih struktura u biološkom sistemu.

- U Tabeli 1.2 je dat pregled osnovnih tehnika biomedicinskog slikanja sa naznačenom prirodom piksela u fizičkom smislu, kao i sa dominantnom primenom svake od tehnika.
- Pojedini modaliteti omogućavaju praćenje promena u biološkom sistemu kroz posmatranje biomedicinskih slika tokom vremena.
- Hibridni pristupi biomedicinskog slikanja kombinuju (registruju) biomedicinske slike dobijene različitim tehnikama biomedicinskog slikanja sa ciljem dobijanja potpunije informacije o morfologiji i/ili funkciji posmatranog biološkog sistema.

2. *PYTHON* OKRUŽENJE

Python programski jezik je 1990-tih osmislio *Guido van Rossum* iz *Centrum Wiskunde & Informatica* (CWI) u Holandiji, a prva njegova javno dostupna implementacija se pojavila februara 1991. godine. Interesantno je pomenuti da je naziv dobio prema popularnoj seriji „*Monty Python's Flying Circus"*[3]. Sve implementacije *Python*-a su besplatno[4] dostupne, a zajednica korisnika *Python* programskog jezika je vrlo široka.

Python je moderan programski jezik visokog nivoa čije biblioteke omogućavaju izuzetno široku primenu, počev od numeričkih proračuna, analize podataka i primene metoda mašinskog učenja pa sve do razvoja *web* aplikacija, programiranja baza podataka i razvoja računarskih igara.

Da bi kôdovi napisani u programskim jezicima visokog nivoa mogli da se izvršavaju na računaru, neophodno je da se uradi kompajliranje (prevođenje u mašinski jezik) ili interpretiranje programskog kôda (prevođenje linije po linije kôda u izvršne instrukcije u realnom vremenu, bez prethodnog prevodjenja u mašinski kôd) ili njihova kombinacija. *Python* programski jezik spada u interpretatorske programske jezike visokog nivoa, tj. *Python* interpreter prevodi kôd u izvršne instrukcije u realnom vremenu, što omogućava izvršavanje kôda na različitim platformama bez izmena kôda (*Windows*, *Mac*, *Linux*, *Unix*). Jezici visokog nivoa kao što je *Python* omogućavaju komforno programiranje, kôd je čitljiviji i lakše se pronalazi greška (debaguje) u odnosu na kôdove pisane u programskim jezicima niskog nivoa, tako da su ovakvi programski jezici pristupačniji početnicima u programiranju, a značajno olakšavaju programiranje i iskusnim programerima. Medjutim, ovaj komfor se postiže na račun smanjene brzine izvršavanja u odnosu na jezike nižeg nivoa. Ovaj nedostatak se može nadoknaditi tako što se najveći deo kôda napiše u programskom jeziku visokog nivoa, a vremenski zahtevni delovi kôda se napišu u jeziku nižeg nivoa. Primeri programskih jezika niskog nivoa su: asembler, mašinski jezik, *C*, *Fortran*, *Ada*, a osim *Python*-a u jezike višeg nivoa spadaju i *C#*, *C++*, *Java*, *PHP*, *Matlab*, *Labview*, *R* i drugi.

Python interpreter može da radi u dva môda: interaktivnom môdu i môdu izvršavanja skripte. Interaktivni môd podrazumeva da se kôd ukucava liniju po liniju, a da interpreter prikazuje rezultat izvršavanja svake od linija.

[3] Prevod: „Leteći cirkus Montija Pajtona" – britanska humoristična TV serija koja se emitovala na *British Broadcasting Corporation* (BBC) televiziji 1969-1974.
[4] Preuzimanje *Python* izvornih kôdova je dostupno preko sledećeg linka: https://www.python.org/downloads/ (poslednji pregled Decembar 2024)

Môd izvršavanja skripte podrazumeva da se interpreteru pošalje ime datoteke sa ekstenzijom *.py* koja sadrži linije kôda i koja se naziva „skripta“, a interpreter u ovom môdu izvršava sadržaj „skripte“. *Python* interpreter ima instalirane i neke od *Python* biblioteka, ali u većini slučajeva programer treba po potrebi da dodatno instalira biblioteke. U praksi se pokazalo da je za istraživački rad i edukaciju udobnije koristiti neku od *Python* distribucija (*Anaconda, Miniconda, Python(x,y), WinPython, Canopy* i sl.) koje već sadrže instalirano razvojno okruženje i veliki broj *Python* biblioteka za analizu i vizuelizaciju podataka, pri čemu je moguće i dodavanje pojedinačnih biblioteka. Najčešće korišćena *Python* distribucija je *Anaconda*[5] koja će nadalje biti korišćena u okviru ovog udžbenika, a dostupna je besplatno za akademsku zajednicu. Sve instrukcije za manipulaciju u okviru *Anaconda* distribucije će biti date za *Windows* operativni sistem.

2.1. *Anaconda* distribucija za *Python*

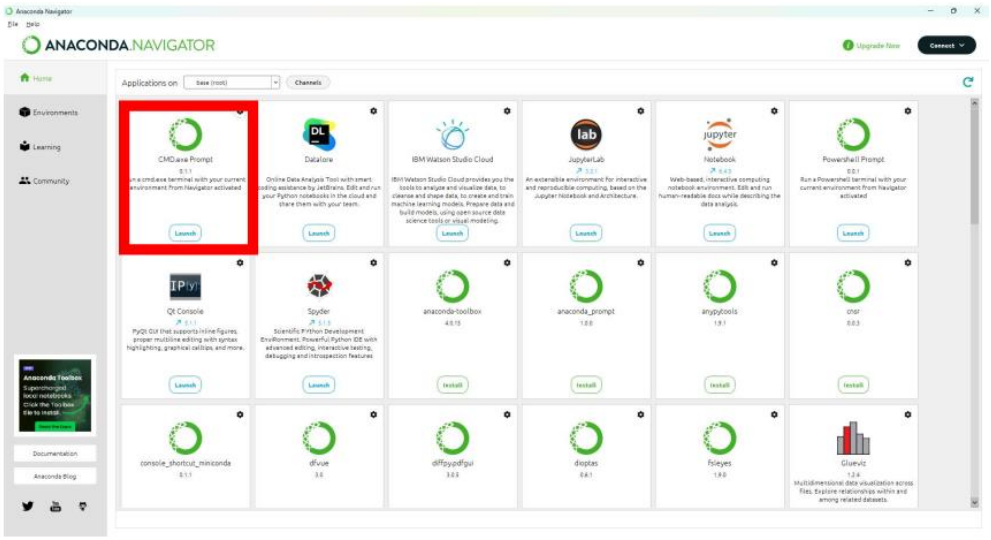
Anaconda distribucija za *Python* je kreirana 2012. godine od strane kompanije *Continuum Analytics* (današnji naziv *Anaconda Inc*.). Ova distribucija obuhvata *Python* interpreter, preko 250 instaliranih *Python* biblioteka uz mogućnost instalacije preko 7500 dodatnih biblioteka, a obuhvata i različite aplikacije:

- *Anaconda Navigator* - grafički korisnički interfejs koji omogućava instaliranje, ažuriranje i upravljanje bibliotekama i okruženjima bez korišćenja komandne linije, Sl. 2.1
- *Conda* - menadžer za instalaciju i upravljanje bibliotekama i okruženjima korišćenjem komandne linije
- *Spyder* - integrisano razvojno okruženje specijalno dizajnirano za istraživanje.
- *Jupyter Notebook* - interaktivno okruženje za pisanje i izvršavanje *Python* kôda koje omogućava kombinovanje kôda, teksta, slike i rezultata izvršavanja kôda
- *JupyterLab* - naprednije okruženje u odnosu na *Jupyter Notebook*, koje omogućava rad sa više dokumenata i radnih prostora istovremeno.

U okviru *Anaconda* distribucije za *Python*, moguće je koristiti *Python* interpreter u oba môda, i u interaktivnom môdu i u môdu izvršavanja skripte. Za korišćenje *Python* interpretera u interaktivnom môdu, potrebno je

[5] Link za preuzimanje *Anaconda* distribucije za *Python*: https://www.anaconda.com/ (poslednji pregled Decembar 2024)

pokrenuti ili terminal *Anaconda Prompt* iz *Start* menija *Windows*-a ili komandni terminal *Windows*-a klikom na opciju *CMD.exe Prompt* u okviru *Anaconda Navigator*-a, Sl. 2.1.



Slika 2.1. ***Anaconda Navigator***

Nakon pokretanja terminala, potrebno je ukucati komandu `python`, nakon čega će se pojaviti ispis verzije *Python* implementacije sličan onome na Sl. 2.2. Simbol „>>>“ označava spremnost za interaktivni unos *Python* kôda, nakon čega je u primeru na Sl. 2.2. uneta jednostavna instrukcija za ispisivanje natpisa „Ovo je knjiga iz analize biomedicinske slike“ na ekranu:

```
print('Ovo je knjiga iz analize biomedicinske slike')
```

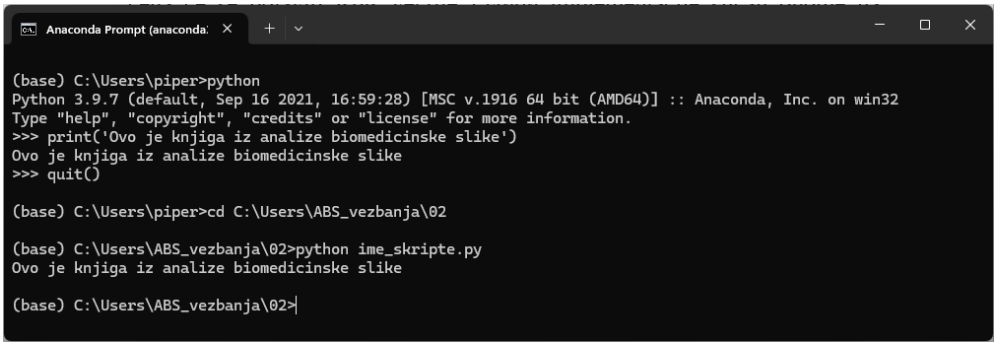
Pomoću instrukcije `quit()` se interaktivni *Python* interpreter zatvara.

Za korišćenje *Python* interpretera u môdu izvršavanja skripte u kome se izvršava ceo kôd sačuvan u datoteci, potrebno je u terminalu izvršiti pokretanje *.py* skripte pomoću instrukcije `python ime_skripte.py`. Za pokretanje skripte, neophodno je da trenutno aktivni direktorijum tj. direktorijum u kome se korisnik nalazi, sadrži skriptu *ime_skripte.py*. U ovom konkretnom primeru direktorijum *C:\Users\ABS_vezbanja\02* sadrži datoteku za pokretanje *ime_skripte.py*. Pozicioniranje u direktorijum *C:\Users\ABS_vezbanja\02* u primeru na Sl. 2.2 je urađeno pomoću instrukcije `cd`:

```
cd C:\Users\ABS_vezbanja\
```

Nakon pozicioniranja u odgovarajući direktorijum, skripta *ime_skripte.py* je pokrenuta pomoću instrukcije:

```
python ime_skripte.py
```



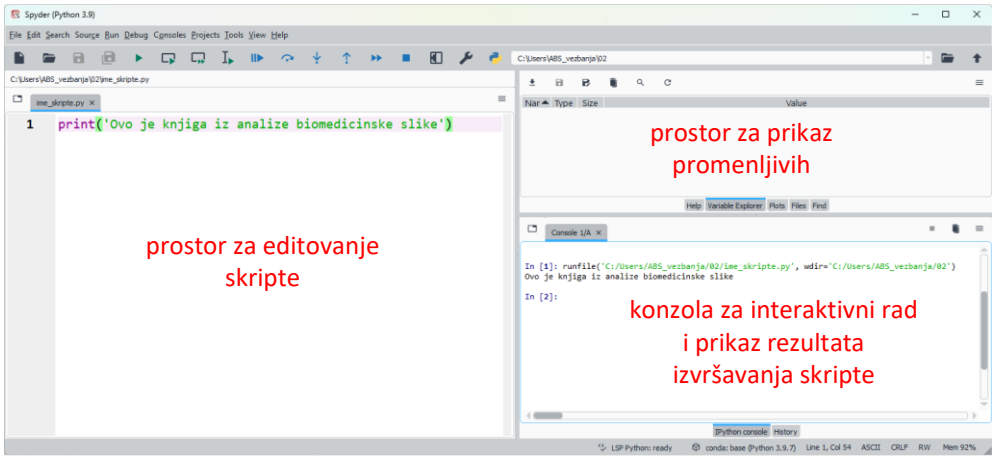
Slika 2.2. Primena *Python interpretera* u terminalu *Anaconda Prompt*

Za pokretanje *.py* skripti se obično ne koristi terminal već neko od integrisanih razvojnih okruženja (eng. *Integrated Development Environment*) koje omogućava i pisanje kôda i debagovanje (*IDLE*, *Spyder*, *Jupiter Notebook/Lab*, *PyCharm*, *Visual Studio Code* i dr.). Za potrebe ove knjige svi kôdovi su testirani u *Spyder* razvojnom okruženju.

2.2. *Spyder* razvojno okruženje

Spyder razvojno okruženje je dostupno u okviru *Anaconda* distribucije i jedno je od najčešće korišćenih razvojnih okruženja. Može se pokrenuti direktno iz *Start* menija *Windows*-a ili odabirom opcije u *Anaconda Navigator*-u.

Na Sl. 2.3 je prikazan izgled *Spyder* okruženja sa otvorenom i pokrenutom datotekom *ime_skripte.py* iz prethodnog poglavlja. *Spyder* okruženje sadrži prostor za editovanje skripte, prostor za prikaz vrednosti i tipova promenljivih (*Variable Explorer*), kao i konzolu za interaktivan rad i prikaz rezultata izvršavanja skripte (*Console 1/A*).



Slika 2.3. *Spyder* okruženje

U *Spyder* okruženju, nova skripta se otvara klikom levog tastera miša na dugme *New file*  , a za otvaranje postojeće skripte treba kliknuti mišem na dugme *Open file*  . Čuvanje skripte se postiže opcijom Save file  Pokretanje izvršavanja skripte se vrši klikom miša na *Run File* dugme  , a rezultat izvršavanje se prikazuje u konzoli. Ukoliko je potrebno da se prikazuju grafici i slike tokom izvršavanja programa u prozoru *Plots* umesto u zasebnim prozorima, potrebno je podesiti *Tools/Preferences/IPython console/Graphics/Backend* na *Inline* vrednost (po *default*-u je vrednost *Automatic* i u tom slučaju se grafici i slike prikazuju odvojeno, svaki u svom prozoru, što može rezultovati prevelikim brojem prozora i nepreglednošću rezultata skripte, pa je preporuka da se podesi *Inline* vrednost).

U *Spyder* okruženju postoje tri opcije za „korak po korak“ izvršavanje skripte:

- *Run current cell*  - izvršavanje izabrane „ćelije”, tj. „ćelije” na kojoj je kursor („ćelija” se označava komentarom koji počije sa `#%%`, videti Pr. 2.1 u kome je prikazan primer *run_opcije.py* sa tri ćelije). Uzastopno izvršavanje ove opcije kao rezultat ima ponavljanje izvršavanja izabrane „ćelije”.
- *Run current cell and go to next one*  - izvršavanje izabrane „ćelije” i prelazak na sledeću „ćeliju”. Uzastopno izvršavanje ove opcije kao rezultat ima sukcesivno izvršavanje „ćelije” po „ćelije”.
- *Run selection or current line*  - izvršavanje izabrane linije kôda ili grupe izabranih linija kôda.

Ukoliko bi se kursor postavio na prvu ćeliju i pokrenula opcija *Run current cell*, rezultat izvršavanja bi bio ispisivanje natpisa `Ovo je knjiga iz analize biomedicinske slike`, Pr. 2.1B.

Pri ponovnom pokretanju na isti način dobio bi se identičan rezultat. Međutim, ukoliko bi se za kursor pozicioniran na prvu „ćeliju” i tri puta za redom pokrenula opcija *Run current cell and go to next one*, na kraju svakog izvršavanja bi bila izabrana naredna ćelija, a rezultat izvršavanja bi bio ispis kao na Pr. 2.1C.

Pr. 2.1. Primer *run_opcije.py*

A. Programski kôd:

```
#%% Prva celija
print('Ovo je knjiga iz analize biomedicinske slike')

#%% Druga celija
print('primenom programskog jezika Python')

#%% Treca celija
print('kroz prakticne primere')
```

B. Rezultat izvršavanja za pokretanje *Run current cell* ako je kursor pozicioniran na početku na prvoj „ćeliji":

```
In [1]: runcell('Prva celija', 'C:/Users/ABS_vezbanja/02/run_opcije.py')
Ovo je knjiga iz analize biomedicinske slike
```

C. Rezultat izvršavanja za tri pokretanja *Run current cell and go to next one* ako je kursor pozicioniran na početku prvog pokretanja na prvoj „ćeliji":

```
In [1]: runcell('Prva celija', 'C:/Users/ABS_vezbanja/02/run_opcije.py')
Ovo je knjiga iz analize biomedicinske slike

In [2]: runcell('Druga celija', 'C:/Users/ABS_vezbanja/02/run_opcije.py')
primenom programskog jezika Python

In [3]: runcell('Treca celija', 'C:/Users/ABS_vezbanja/02/run_opcije.py')
kroz prakticne primere

In [4]:
```

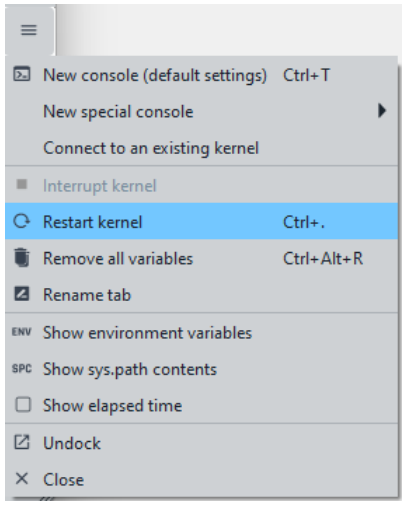
U okviru *Spyder* okruženja moguće je koristiti i môd za debagovanje (testiranje funkcionalnosti) u kome se kôd izvršava korak po korak, uz mogućnost debagovanja i primene pojedinih funkcija:

- *Debug File* - omogućava ulazak u môd za debagovanje
- *Run current line* - izvršavanje trenutne linije kôda
- *Step into function or method of current line* - ulaženje u funkciju ili metodu trenutne linije

- *Run until current function or method returns* - izlazak iz funkcije ili metode po završetku
- *Continue execution until next breakpoint* - izvršavanje do naredne prekidne tačke (eng. *breakpoint*)
- *Stop debugging* - završetak debagovanja.

Ukoliko je potrebno da se obriše prethodna istorija izvršavanja u *Spyder*-u, to je najefikasnije uraditi primenom opcije *Restart kernel* koja se nalazi u meniju u gornjem desnom uglu konzole, Sl. 2.4.

Brisanje svih promenljivih se može postići izborom opcije *Remove all variables* koja se takođe nalazi u meniju u gornjem desnom uglu konzole. Isti efekat se postiže i primenom komandi: `%reset` ili `%reset -f` u konzoli. Razlika u primeni ove dve komande je u tome što je kod prve neophodno potvrditi brisanje promenljivih, a kod druge se promenljive brišu bez prethodnog potvrđivanja.



Slika 2.4. Ponovno pokretanje kernela u *Spyder*-u

U ovom poglavlju je dat samo pregled osnovnih funkcionalnosti *Spyder* okruženja neophodnih za rad u narednim poglavljima, a detaljniji opis rada u ovom okruženju se može naći na dostupnom *online* repozitorijumu[6].

2.3. Osnovna *Python* sintaksa

Komentarisanje kôda je deo dobre prakse u programiranju. U *Python*-u, moguće je uneti jednu ili više linija komentara. Jedna linija komentara započinje simbolom #, a više linija komentara započinju i završavaju se sa

[6] https://docs.spyder-ide.org/current/index.html (poslednji pregled Decembar 2024)

'''. Na Pr. 2.2 je prikazan primer *sintaksa1.py* na čijem početku je demonstrirana primena oba načina komentarisanja. Najpre je definisan višelinijski komentar između simbola '''':

```
Ilustracija Python sintakse:
-komentari
-dinamicki jezik
```

a potom je definisan i komentar:

```
Python je dinamicki jezik
```

u jednoj liniji na čijem početku se nalazi simbol #. Pri pokretanju kôda, komentari se ne ispisuju, već služe samo da učine kôd jasnijim za razumevanje.

Pr. 2.2. Primer *sintaksa1.py* - primena komentara i dinamičkih osobina

---

A. Programski kôd:

```
'''
Ilustracija Python sintakse:
-komentari
-dinamički jezik
'''

# Python je dinamički jezik
x = 10
print('x je tip:', type(x))

x = 'Sada sam string'
print('x je tip:', type(x))
```

B. Rezultat izvršavanja:

```
In [1]: runfile('C:/Users/ABS_vezbanja/02/sintaksa1.py', wdir='C:/Users/ABS_vezbanja/02')
x je tip: <class 'int'>
x je tip: <class 'str'>

In [2]:
```

Python je dinamički programski jezik, tj. tipovi promenljivih i funkcije se mogu menjati ili određivati u toku izvršavanja programa. Na primer, ista

promenljiva `x` u programu može da ima najpre jedan tip podataka, a potom neki drugi tip podataka. U primeru *sintaksa1.py* na Pr. 2.2 je promenljiva `x` najpre definisana kao ceo broj (eng. *integer*) i dodeljena joj je vrednost `10`, a odmah potom je definisana kao tip niska (eng. *string*) i dodeljena joj je vrednost `'Sada sam string'`. Pomoću funkcije `type(x)` je za svaku od dodela vrednosti za `x` odredjen tip promenljive i ispisan je pomoću funkcije `print`. Rezultat izvršavanja je prikazan na Pr. 2.2B.

U *Python*-u, uvlačenje reda (indentacija) je ključan deo sintakse i određuje hijerarhiju, tj. označava pripadnost određenim blokovima kôda. Na primer, kada se definiše `for` petlja, sve instrukcije koje se izvršavaju u okviru tela `for` petlje treba da budu uvučene u odnosu na samu `for` petlju. Na Pr. 2.3 je prikazan primer *sintaksa2.py* koji ilustruje indentaciju kroz primenu `for` petlje i `if-else` uslova na osnovnoj i najčešće korišćenoj strukturi podataka u *Python*-u koja se zove lista. U primeru je najpre definisana lista od pet brojeva [1,5,3,4,2] i prikazana pomoću `print` instrukcije. Potom je uvedena `for` petlja u čijem telu se nalazi uvučena funkcija `print` za prikaz pojedinačnog elementa liste u svakoj od iteracija `for` petlje (broj iteracije je jednak broju elemenata liste). Poslednji deo kôda u primeru *sintaksa2.py* novom `for` petljom prolazi kroz sve elemente liste i pomoću uvučenog `if-else` uslova proverava da li je element paran ili neparan. Parnost elementa se proverava pomoću operatora jednakosti `==` (ostatak elementa liste i broja 2 se poredi da li je jednak nuli). Za prikaz natpisa o parnosti tj. neparnosti elementa liste, koristi se tzv. *f-string*, tj. niska karaktera u kojoj je umetnuta promenljiva `broj` u vitičastoj zagradi {broj}. Ovakvom dinamički građenom niskom je moguće prilagoditi prikaz natpisa za svaki pojedinačni element liste.

U primeru *sintaksa3.py* na Pr. 2.4 je prikazan alternativni način zadavanja iteracija `for` petlje pomoću instrukcije `range` i dužine liste. Najpre je pomoću `len` funkcije određena dužina liste, tj. broj elemenata liste, a potom je broj iteracija `for` petlje definisan pomoću `range` funkcije (opseg iteracija je zadat da je od 0 do dužine liste – prva vrednost iteratora `broj` će biti 0, poslednja vrednost iteratora `broj` će biti `duzina-1`, a ukupan broj iteracija će biti jednak duzini liste). Pomoću `lista[broj]` se pristupa svakom pojedinačnom elementu liste, počev od nultog elementa, pa sve do elementa sa rednim brojem `duzina-1`. Rezultat prikaza je dat na Pr. 2.4B i identičan je odgovarajućem prikazu sa Pr. 2.3B.

Pr. 2.3. Primer *sintaksa2.py* - indentacija

A. Programski kôd:

```
'''
Ilustracija Python sintakse:
-indentacija (uvlačenje reda)
-lista
-for petlja i if-else uslov
'''
# Definisanje liste brojeva i ispisivanje
lista = [1, 5, 3, 4, 2]
print('Lista:', lista)

# Iteracija kroz listu koristeći for petlju
print('Iteracija kroz listu:')
for broj in lista:
    print(broj)                              # Uvlačenje

# Korišćenje if-else uslova
print('Provera brojeva u listi:')
for broj in lista:
    if broj % 2 == 0:                        # Uvlačenje
        print(f'{broj} je paran broj.')      # Uvlačenje
    else:                                    # Uvlačenje
        print(f'{broj} je neparan broj.')    # Uvlačenje
```

B. Rezultat izvršavanja:

```
Lista: [1, 5, 3, 4, 2]
Iteracija kroz listu:
1
5
3
4
2
Provera brojeva u listi:
1 je neparan broj.
5 je neparan broj.
3 je neparan broj.
4 je paran broj.
2 je paran broj.
```

Pr. 2.4. Primer *sintaksa3.py* - primena `len` i `range` u `for` petlji

---

A. Programski kôd:

```
# Definisanje liste brojeva i ispisivanje
lista = [1, 5, 3, 4, 2]
print('Lista:', lista)

# Ispisivanje dužine liste
duzina=len(lista)
print('Dužina liste:', duzina)

# Iteracija kroz listu koristeći for petlju
print('Iteracija kroz listu:')
for broj in range(0,duzina):
    print(lista[broj])
```

B. Rezultat izvršavanja:

```
Lista: [1, 5, 3, 4, 2]
Dužina liste: 5
Iteracija kroz listu:
1
5
3
4
2

In [2]:
```

U primeru *sintaksa4.py* na Pr. 2.5 je demonstrirano i definisanje liste čiji elementi su tipa string, kao i korišćenje metode `sort` za sortiranje, `append` za dodavanje i `remove` za brisanje elemenata liste.

Opšta sintaksa za primenu metoda u *Python*-u na neku promenljivu je `ime_promenljive.ime_metode`. U kôdu na Pr. 2.5 je najpre formirana lista koja sadrži stringove 'mozak', 'srce', 'pluca', 'bubrezi'. Potom je lista sortirana po abecednom redosledu primenom `sort` metode i rezultat sortiranja je prikazan. Zatim je dodat string 'skelet' primenom `append` metode i rezultujuća lista je prikazana. Na kraju je iz liste uklonjen element 'srce' primenom metode `remove`. Rezultat izvršavanja je prikazan na Pr. 2.5B.

Pr. 2.5. Primer *sintaksa4.py* - lista stringova i primena metoda

A. Programski kôd:

```
# Definisanje liste stringova i ispisivanje
lista = ['mozak', 'srce', 'pluca', 'bubrezi']
lista.sort()
print('Lista:', lista)

# Dodavanje elementa u listu - metoda append
lista.append('skelet')
print('Lista nakon dodavanja:', lista)

# Uklanjanje elementa iz liste - metoda remove
lista.remove('srce')
print('Lista nakon uklanjanja:', lista)
```

B. Rezultat izvršavanja:

```
Lista: ['bubrezi', 'mozak', 'pluca', 'srce']
Lista nakon dodavanja: ['bubrezi', 'mozak', 'pluca', 'srce', 'skelet']
Lista nakon uklanjanja: ['bubrezi', 'mozak', 'pluca', 'skelet']
```

Za medicinske primene je od posebnog interesa još jedan tip podataka karakterističan za *Python*, a to su rečnici. Rečnik je struktura podataka koja skladišti parove „ključ-vrednost“ (eng. *key-value*). Ključ je jedinstven, a vrednost može imati bilo koju vrednost. Slično listama, i elemente rečnika je moguće brisati ili dodavati nove. Na Pr. 2.6. je dat primer *sintaksa5.py* koji definiše `recnik` sa pet parova „ključ-vrednost“ (npr. `'ime'` je ključ, a `'Petar'` je vrednost), a potom ispisuje rečnik. Zatim se pomoću metode `recnik.keys()` daje pregled svih ključeva. Pojedinačnom elementu rečnika može da se pristupi preko `recnik['ključ']` instrukcije i u ovom primeru je pristupljeno elementu sa ključnom reči `'ime'` i njegova vrednost `'Petar'` je ispisana na ekranu. Dodavanje elemenata u rečnik je moguće sprovesti putem instrukcije oblika `recnik['novi_ključ']= 'nova_vrednost'` što je iskorišćeno za dodavanje elementa koji opisuje `modalitet` slikanja (`CT`). Brisanje elementa je moguće instrukcijom `del recnik['ključ']` i u primeru je obrisan element koji se tiče informacije o doktoru. Na kraju je realizovana iteracija kroz `kljuc, vrednost` parove definisane sa instrukcijom `recnik.items()` i ispis, Pr. 2.6B.

Pr. 2.6. Primer *sintaksa5.py* - rečnik

A. Programski kôd:

```
# Definisanje rečnika i ispisivanje
recnik = {'ime': 'Petar',
          'prezime': 'Petrovic',
          'godine': 25,
          'pregled': 'pluca',
          'doktor': 'Dr Pavlovic'}
print('Rečnik:', recnik)

print('Kljucevi:', recnik.keys()) #Pregled kljuceva
print('Ime iz rečnika:', recnik['ime'])   # Pristup

recnik['modalitet']='CT'       # Dodavanje elementa
print('Rečnik nakon dodavanja:', recnik)

del recnik['doktor']           # Brisanje elementa
print('Rečnik nakon uklanjanja:', recnik)

# Iteracija kroz rečnik
print('Iteracija kroz rečnik:')
for kljuc, vrednost in recnik.items():
    print(f"{kljuc}: {vrednost}")
```

B. Rezultat izvršavanja:

```
Rečnik: {'ime': 'Petar', 'prezime': 'Petrovic', 'godine': 25, 'pregled': 'pluca', 'doktor': 'Dr
Pavlovic'}
Kljucevi: dict_keys(['ime', 'prezime', 'godine', 'pregled', 'doktor'])
Ime iz rečnika: Petar
Rečnik nakon dodavanja: {'ime': 'Petar', 'prezime': 'Petrovic', 'godine': 25, 'pregled':
'pluca', 'doktor': 'Dr Pavlovic', 'modalitet': 'CT'}
Rečnik nakon uklanjanja: {'ime': 'Petar', 'prezime': 'Petrovic', 'godine': 25, 'pregled':
'pluca', 'modalitet': 'CT'}
Iteracija kroz rečnik:
ime: Petar
prezime: Petrovic
godine: 25
pregled: pluca
modalitet: CT
```

U ovom poglavlju su kroz primere iznete osnove *Python* sintakse neophodne za razumevanje kôdova koji će uslediti u narednim poglavljima.

Za detaljnije informisanje o *Python* sintaksi preporučena je dodatna literatura [Downey 2024], [Ceder 2018].

2.4. *Python* biblioteke

Python ima bogatu arhivu biblioteka (preko 300 000) koje sadrže pripremljene funkcije, klase i module dostupne za dalje korišćenje i kombinovanje. U okviru *Anaconda* distribucije, instalacija biblioteka se može obaviti ili u okviru *Anaconda Navigator*-a ili u okviru *Anaconda Prompt*-a primenom *Conda* menadžera.

Svaka *Python* biblioteka čije funkcionalnosti se koriste mora biti ubačena (importovana) na početku programa pomoću funkcije `import ime_biblioteke`. Primer importovanja *Pyplot* modula *Matplotlib* biblioteke je dat na Pr. 2.7 gde se radi čitljivosti programskog kôda uvodi skraćenica `plt` za pristup svim funkcijama *Pyplot* modula. Kao primer je navedeno korišćenje `close ('all')` instrukcije koja zatvara sve otvorene grafičke predstave. Korišćenje ove instrukcije na početku programa koji prikazuje grafičke predstave rezultata je deo dobre programerske prakse (pod uslovom da je *Bakend* podešen na *Automatic*, vidi Poglavlje 2.2).

Pr. 2.7. Importovanje *Pyplot modula Matplotlib* biblioteke

Programski kôd:

```
import matplotlib.pyplot as plt
plt.close('all')
```

Biblioteke koje će biti korišćene u nastavku ove knjige su:

- *Imageio* – funkcije koje omogućavaju učitavanje i čuvanje slika. Biblioteka je obično instalirana u okviru *Anaconda* distribucije. Za importovanje se koristi: `import imageio`.
- *Numpy* (*Numerical Python*) - osnovna biblioteka za rad sa numeričkim podacima u *Python*-u. Omogućava rad sa velikim, višedimenzionalnim nizovima i matricama, poseduje širok spektar matematičkih funkcija koje omogućavaju efikasne operacije nad nizovima i matricama. Biblioteka je instalirana u okviru *Anaconda* distribucije. Za importovanje se koristi: `import numpy as np`
- *Matplotlib* - kreiranje statičnih, animiranih i interaktivnih vizuelizacija podataka. Biblioteka je obično instalirana u okviru *Anaconda* distribucije. Za importovanje se koristi: `import matplotlib`

- *Ndimage* modul u okviru *Scipy* (*Scientific Python*) biblioteke - funkcije za filtriranje, morfološke operacije, geometrijske transformacije i analizu slike. Biblioteka je obično instalirana u okviru *Anaconda* distribucije. Za importovanje se koristi: `from scipy import ndimage as ndi`.

- *OpenCV* (*Open Source Computer Vision*) – funkcije za filtriranje, morfološke operacije, geometrijske transformacije i analizu slike. Biblioteka nije instalirana u okviru *Anaconda* distribucije i neophodno ju je naknadno instalirati.[7] Za importovanje se koristi: `import cv2`.

- *Pandas* – funkcije koje omogućavaju učitavanje i čuvanje podataka (.csv, .xls, .json i dr. formati), rad sa tabelama (koristi *DataFrame* strukturu sličnu *Excel* tabelama), manipulaciju podacima (filtriranje, grupisanje, uređivanje, „čišćenje“, povezivanje više tabela i sl.), analizu podataka (podržava i rad sa vremenskim serijama). Biblioteka je obično instalirana u okviru *Anaconda* distribucije. Za importovanje se koristi: `import pandas`.

- *PIL/Pillow* – funkcije koje omogućavaju učitavanje i čuvanje 2D slika kao i obradu slika. Biblioteka je obično preinstalirana u okviru *Anaconda* distribucije. Za importovanje se koristi: `import PIL`.

- *Pydicom* – funkcije koje omogućavaju učitavanje i čuvanje podataka u DICOM formatu (detalje formata pogledati u Poglavlju 3.2). Biblioteka nije instalirana u okviru *Anaconda* distribucije i neophodno ju je naknadno instalirati[8]. Za importovanje se koristi: `import pydicom`.

- *SimpleITK* (*Simple Insight Segmentation and Registration Toolkit*) - algoritmi za obradu medicinskih slika, segmentaciju i registraciju. Biblioteka nije instalirana u okviru *Anaconda* distribucije i neophodno ju je naknadno instalirati.[9] Za importovanje se koristi: `import SimpleITK as sitk`.

- *Skimage* (*Scikit-image*) modul u okviru *Scikits* (*Scientific Kits*) – algoritmi za filtriranje, morfološke operacije i analizu slike. Biblioteka je obično instalirana u okviru *Anaconda* distribucije. Za importovanje se koristi: `import skimage`. Za importovanje

[7] Conda instalacija za *OpenCV*: `conda install -c conda-forge opencv`
[8] Conda instalacija za *Pydicom*: `Conda install -c conda-forge pydicom`
[9] Conda instalacija za *SimpleITK*: `conda install simpleitk::simpleitk`

podmodula `filters` u okviru `skimage` modula, koristi se `from skimage import filters`.

- *Scikit-learn* modul - definisanje i treniranje modela mašinskog učenja. Biblioteka je obično instalirana u okviru *Anaconda* distribucije. Za importovanje se koristi: `import sklearn`[10].
- *TensorFlow* - definisanje i treniranje modela mašinskog učenja, pre svega dubokih modela mašinskog učenja. Biblioteka nije instalirana u okviru *Anaconda* distribucije i neophodno ju je naknadno instalirati. Pri primeni *TensorFlow* biblioteke, preporuka je da se formira poseban *TensorFlow environment* (tf)[11] i da se aktivira[12], a u okviru njega da se instaliraju sve neophodne biblioteke za rad, jer instalacija *TensorFlow*-a i njemu kompatibilnih i zavisnih biblioteka može da dovede do kolizije sa već instaliranim *Python* bibliotekama.
- *VTK* (*Visualization Toolkit*) – modul za 3D računarstvo, vizuelizaciju i obradu podataka. Biblioteka nije instalirana u okviru *Anaconda* distribucije i neophodno ju je naknadno instalirati.[13] Za importovanje se koristi: `import vtk`.

Po potrebi, mogu se kreirati i novi moduli sa funkcijama koje se mogu importovati kao i standardne *Python* biblioteke. Na Pr. 2.8 su prikazani: A) programski kôd *mikroskopska_slika.py* koji sadrži definisanu jednu funkciju `prikazi_mikroskopsku_sliku()`, B) programski kôd *glavni_program.py* koji importuje i poziva definisanu funkciju `prikazi_mikroskopsku_sliku()` i C) rezultat izvršavanja glavnog programa *glavni_program.py*.

U datoteci *mikroskopska_slika.py* se najpre importuju neophodni moduli (podmodul *Pyplot* i modul *Numpy*), a potom se pod ključnom reči `def` definiše funkcija sa nazivom `prikazi_mikroskopsku_sliku` i argumentima za širinu i visinu slike (sirina, visina). Za *default*-ne vrednost širine i visine je postavljeno 100 piksela (ukoliko se pri pozivanju funkcije ove vrednosti ne budu definisale, koristiće se vrednosti postavljene po *default*-u). Vrednosti piksela matrice `slika` sa dimenzijama `sirina` i `visina` se kreiraju generisanjem slučajnih brojeva pomoću instrukcije

[10] Često se importuju sledeći podmoduli:
```
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
```
(slično se importuju i drugi modeli mašinskog učenja)
```
from sklearn.metrics import accuracy_score
```
[11] Kreiranje *TensorFlow environment-a* (*tf*): `conda create -n tf tensorflow`
[12] Aktiviranje *tf environment*-a: `conda activate tf`
[13] Conda instalacija za *VTK:* `conda install -c conda-forge vtk`

`random.rand(visina, sirina)` importovane *Numpy* biblioteke. Za prikaz kreirane slike se koristi funkcija `imshow(slika)` importovanog *Pyplot* podmodula.

U datoteci *glavni_program.py* se najpre importuje funkcija `prikazi_mikroskopsku_sliku` definisana u okviru kreiranog modula *mikroskopska_slika*, a potom se na odgovarajući način poziva funkcija `prikazi_mikroskopsku_sliku` sa argumentima `sirina=150` i `visina=150`. Primer simulirane mikroskopske slike je prikazan na Pr. 2.12C.

Pr. 2.8. Pozivanje kreiranog modula za simuliranje mikroskopske slike

---

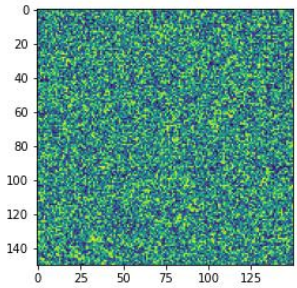
A. Programski kôd *mikroskopska_slika.py*:

```
import matplotlib.pyplot as plt
import numpy as np

def prikazi_mikroskopsku_sliku(sirina=100, visina=100):
    '''
    Funkcija za generisanje i prikaz mikroskopske
    slike sa slučajnim podacima.
    Parametri:
    sirina (int): Širina slike
    visina (int): Visina slike
    '''
    # Kreiranje slike sa slučajnim podacima
    slika = np.random.rand(visina, sirina)

    # Prikaz slike
    plt.imshow(slika)
```

B. Programski kôd *glavni_program.py*:

```
from mikroskopska_slika import prikazi_mikroskopsku_sliku
# Pozovi funkciju
prikazi_mikroskopsku_sliku(150,150)
```

C. Rezultat izvršavanja za *glavni_program.py*:



2.5. Rezime poglavlja

- *Python* programski jezik spada u interpretatorske programske jezike visokog nivoa
- *Python* kôd se u neizmenjenom obliku može izvršavati na različitim platformama (*Windows, Mac, Linux, Unix*).
- *Python* kôd je čitljiviji i lakše se pronalazi greška (debaguje) u odnosu na kôdove pisane u programskim jezicima niskog nivoa.
- *Python* ima smanjenu brzinu izvršavanja u odnosu na jezike nižeg nivoa.
- *Anaconda* je distribucija za *Python* koja omogućava rad *Python* interpretera i u interaktivnom môdu i u môdu izvršavanja skripte.
- *Spyder* razvojno okruženje je dostupno u okviru *Anaconda* distribucije koje omogućava editovanje i debagovanje skripti.
- *Python* je dinamički jezik, tj. tipove promenljivih i funkcije je moguće određivati i menjati u toku izvršavanja programa.
- Komentarisanje kôda je dobra praksa u programiranju.
- Indentacija (uvlačenje reda) je ključni element sintakse u *Python*-u i određuje hijerarhiju i organizaciju blokova kôda.
- *F-string* je opcija korisna za dinamičko kreiranje natpisa i omogućava kombinovanje konstantnog stringa sa vrednostima promenljivih.
- Lista je osnovna i najčešće korišćena struktura podataka u *Python*-u.
- Rečnik je struktura podataka koja skladišti parove „ključ-vrednost“ u *Python*-u.
- *Numpy, Matplotlib, Ndimage, OpenCV, Pandas, SimpleITK, Skimage, Scikit-learn, TensorFlow, VTK* su moduli koji će biti korišćeni u okviru ove knjige. Po potrebi, moguće je kreirati i importovati sopstvene module.

3. PRIKAZ I ČUVANJE U DATOTEKU

3.1. Predstavljanje slika

Bitska dubina. U Poglavlju 1 je objašnjeno da digitalne biomedicinske slike imaju matematičku prezentaciju u vidu matrica. Vrednosti piksela/voksela biomedicinskih slika su u najvećem broju slučajeva celobrojne jer ih je lakše arhivirati od realnih vrednosti. Svaka celobrojna vrednost piksela/voksela se može zapisati i u binarnom obliku. Npr. broj 10 se u binarnom obliku zapisuje kao 1010 ($1*2^3+0*2^2+1*2^1+0*2^0=10$), pri čemu se broj bita koji su potrebni za zapis piksela/voksela slike u binarnom obliku naziva bitska dubina (eng. *bit depth*). Na primer, za 8-bitnu sliku je broj mogućih vrednosti piksela/voksela 256 ($=2^8$), a opseg vrednosti je $[0,2^8-1]=[0,255]$ ako ne sadrži negativne vrednosti. Većina fotografskih formata koristi 8 bita, a vrednosti piksela su samo pozitivne. Medicinske slike često imaju veću bitsku dubinu što zahteva više prostora za arhiviranje. Na primer, 16-bitna medicinska slika ima ukupno 65536 ($=2^{16}$) mogućih vrednosti piksela/voksela, a opseg vrednosti je $[0,2^{16}-1]=[0,65535]$ ako ne sadrži negativne vrednosti tj. $[-2^{15}, +2^{15}-1]=[-32768, +32767]$ ako sadrži i negativne i pozitivne vrednosti i nulu.

Palete boja. Primer Pr. 3.1 *glavni_program_mod.py* je modifikovana verzija programa Pr. 2.8 iz prethodnog Poglavlja 2 koja simulira prikaz mikroskopske slike. Modifikacija se sastoji u tome što se jednostavnosti radi posmatra mala matrica dimenzija 3x3 pri čemu su elementi matrice `slika` celobrojni nenegativni brojevi (omogućeno funkcijom `np.random.randint()`) u opsegu [0,256) (odgovarajući argumenti funkcije su `donja_granica=0` i `gornja_granica=256`, a neoznačeni celobrojni tip sa 8 bita je zadat sa `tip=np.uint8`). Za vizuelno predstavljanje matrica slika se koriste palete boja (eng. *colormap* ili skraćeno *cmap*). Paleta boja je niz boja koji sadrži onoliko elemenata koliko ima mogućih vrednosti piksela/voksela slike. U ovom primeru je matrica `slika` vizuelno prikazana pomoću funkcije `plt.imshow()` primenom *grayscale* palete tj. palete sivog (`cmap='gray'`), a ceo niz boja primenjene palete je takođe vizuelizovan korišćenjem funkcije `plt.colorbar()`. Pr. 3.1C prikazuje rezultat izvršavanja kôda, tj. prikazuje najpre numeričke vrednosti matrice `slika`, a potom i vizuelni prikaz slike u paleti sivog sa nizom boja (eng. *colorbar*) sa desne strane slike. Crna boja odgovara minimalnoj mogućoj vrednosti piksela slike (0), a bela boja odgovara maksimalnoj mogućoj vrednosti piksela slike (255). Može se uočiti i da se element matrice slike *I(0,0)* tj. element u 0-tom redu i 0-toj koloni matrice nalazi u gornjem levom ćošku slike, tj. koordinatni sistem slike je postavljen uvek kao na Sl. 3.1A.

Pr. 3.1. Prikaz simulirane slike u paleti sivog

A. Programski kôd *mikroskopska_slika_mod.py*:

```
import matplotlib.pyplot as plt
import numpy as np

def prikazi_mikroskopsku_sliku(donja_granica=0,
                               gornja_granica=256,
                               sirina=100,
                               visina=100,
                               tip=np.uint8):
    '''
    Funkcija za generisanje i prikaz mikroskopske
    slike sa slučajnim podacima.
    Parametri:
    sirina (int): Širina slike
    visina (int): Visina slike
    '''
    # Kreiranje slike sa slučajnim podacima
    slika = np.random.randint(donja_granica,
                              gornja_granica,
                              size=(visina, sirina),
                              dtype=tip)
    print('Matrica simulirane slike je:')
    print(slika)

    # Prikaz slike
    plt.imshow(slika, cmap='gray')
    plt.colorbar()
```

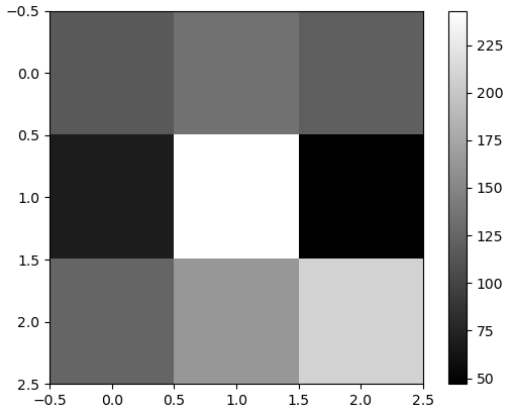
B. Programski kôd *glavni_program_mod.py*:

```
from mikroskopska_slika_mod import prikazi_mikroskopsku_sliku
import numpy as np
```

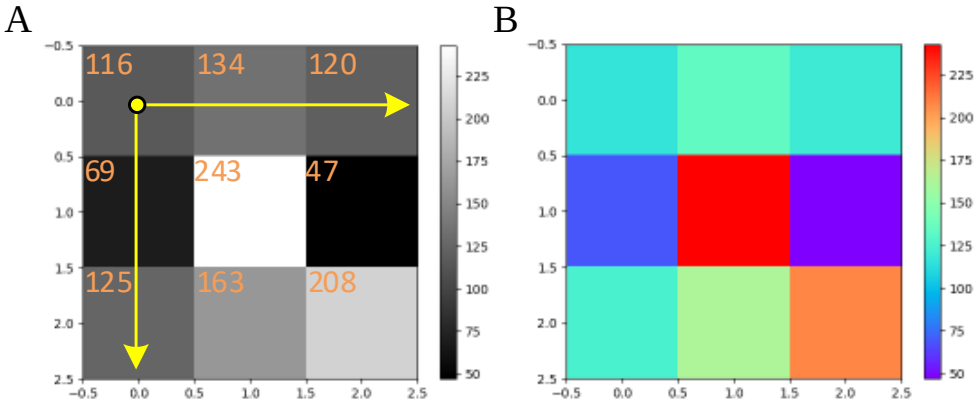
```
# Pozovi funkciju
prikazi_mikroskopsku_sliku(0,256,3,3,np.uint8)
```

C. Rezultat izvršavanja za *glavni_program_mod.py*:

```
Matrica simulirane slike je:
[[116 134 120]
 [ 69 243  47]
 [125 163 208]]
```



Najčešće korišćene palete boja u biomedicinskom slikanju su osim već prikazane palete sivog (*gray*) i sledeće: *binary*, *rainbow*, *hot*, *jet*, *bone*, *viridis* (ova paleta je po *default*-u postavljena u *Python*-u). Na Sl. 3.1B je prikazana slika iz Pr. 3.1. prikazana u *rainbow* paleti (argument `cmap='rainbow'` u `plt.imshow()`). Izbor palete boja zavisi od konkretnog cilja vizuelizacije, tj. detalja na biomedicinskoj slici koje treba istaći.



Slika 3.1. A) Koordinatni sistem slike (paleta sivog), B) Prikaz slike u *rainbow* paleti

RGB sistem. RGB je sistem koji za definisanje boja koristi uređenu trojku intenziteta tri osnovne boje: crvenu (eng. *red*) – R, zelenu (eng. *green*) – G i plavu (eng. *blue*) – B. Na primer, za 8-bitnu sliku u kojoj je opseg intenziteta za svaku od R,G,B boja [0,255], uređena trojka (255,0,0) označava crvenu boju, uređena trojka (0,255,0) označava zelenu boju, uređena trojka (0,0,255) označava plavu boju, a sve ostale boje su kombinacija R,G,B vrednosti. Za slike čije vrednosti piksela su zabeležene u RGB sistemu, moguće je izvršiti konverziju iz RGB sistema u skalu sivog, a najčešće korišćeni metod za to je metod osvetljenosti:

$$\rho = 0.299 * R + 0.587 * G + 0.114 * B \tag{3.1}$$

gde je ρ intenzitet na skali sivog, a R, G i B su vrednosti odgovarajućih intenziteta u RGB sistemu. Jednačina (3.1) je deo *National Television System Committee* (*NTSC*) standarda iz 1953. godine koji je omogućio kompatibilnost između crno-belih i kolor televizora, a bazirana je na istraživanjima percepcije boja ljudskog oka. Ljudsko oko može da vidi svetlost iz vidljivog dela spektra (400-700 nm, pogledati Tabelu 1.1), ali nije jednako osetljivo na sve talasne dužine. Ako se posmatraju tri osnovne boje (crvena, zelena i plava), pokazano je da je spektralna osetljivost ljudskog oka najveća za talasne dužine zelenog dela spektra (zelena svetlost doprinosi 58.7% ljudske percepcije osvetljenosti), potom slede talasne dužine crvenog dela spektra (crvena svetlost doprinosi 29.9% ljudske percepcije osvetljenosti), najmanja je spektralna osetljivost na talasne dužine plavog dela spektra (plava svetlost doprinosi 11.4% ljudske percepcije osvetljenosti).

3.2. Formati datoteka

Za zapisivanje biomedicinskih slika u datoteke mogu se koristiti različiti tipovi formata: formati sirovih 2D podataka, formati sa kompresijom 2D slika i formati posebno namenjeni medicinskim slikama.

Formati sirovih 2D podataka (eng. *raw data*) su zapisi kod kojih se vrednosti intenziteta piksela 2D slike upisuju redom u fajl, bez kompresije. Primeri ovakvih formata su *Portable Gray Map* (PGM) i *BitMaP* (BMP). *PGM* zapis je predviđen samo za *grayscale* slike (vrednosti intenziteta mogu biti upisane u tekstualnom ili u binarnom obliku), a *BMP* (samo binarni) format dozvoljava i čuvanje informacija o bojama prema RGB sistemu. Primer Pr. 3.2 i Sl. 3.2. ilustruju implementaciju upisa i strukturu tekstualnog *PGM* fajla koji sadrži: 1) zaglavlje datoteke (oznaka P2 za *PGM* format u prvom redu datoteke, širina i visina matrice u drugom redu datoteke, maksimalna vrednost intenziteta u paleti sivog u trećem redu datoteke) i 2) telo datoteke (3x3 matrica vrednosti intenziteta piksela gde jedan red matrice odgovara jednom redu u tekstualnoj datoteci).

Pr. 3.2. Primer *PGM_format.py* – zapis tekstualnog PGM fajla

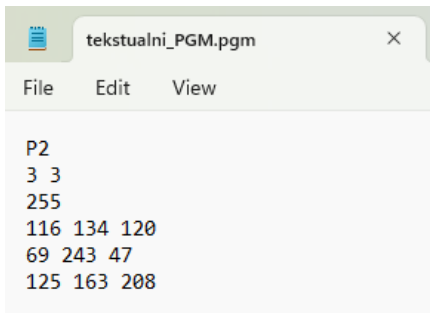
---

A. Programski kôd:

```
import numpy as np

# Matrica slika 3x3
matrica = np.array([
    [116, 134, 120],
    [69, 243,  47],
    [125, 163, 208]],
    dtype=np.uint8)

# Čuvanje u tekstualnom PGM formatu
with open("tekstualni_PGM.pgm", "w") as f:
    # Upis oznake za PGM
    f.write("P2\n")
    # Upis širine i visine matrice
    f.write(f"{matrica.shape[1]} {matrica.shape[0]}\n")
    # Upis maksimalne vrednosti u paleti sivog
    f.write("255\n")
    # Upis svakog reda matrice kao niske,
    # sa razmakom između elemenata matrice
    for red in matrica:
        f.write(" ".join(map(str, red)) + "\n")
```

B. Rezultat izvršavanja:

Kreirana je datoteka tekstualni_PGM.pgm.



Slika 3.2. Primer PGM formata – datoteka *tekstualni_PGM.pgm*

Formati sa kompresijom 2D slika mogu biti sa ili bez gubitaka informacija. *Portable Network Graphics (PNG)* i *Tagged Image File Format (TIFF)* su primeri formata za zapisivanje 2D slika kod kojih se primenjuje tzv. *Run Length Encoding (RLE)* kompresija bez gubitaka. *RLE* kompresija identifikuje delove podataka koji se ponavljaju i svaku sekvencu ponovljenih vrednosti zamenjuje brojem ponavljanja i vrednošću koja se ponavljala. Na primer, za niz karaktera AAAAABBBBAACCCCC (16 bajtova) će rezultat *RLE* kompresije biti 5A4B2A5C (8 bajtova), tj. broj bajtova koji se koristi za zapisivanje će biti redukovan. *Joint Photographic Experts Group (JPEG)* je primer standarda za efikasno zapisivanje 2D slika tj. kompresiju sa gubicima. *JPEG* ne koristi RGB sistem već YCbCr sistem gde je Y osvetljenost tj. intenzitet piksela dat jednačinom (3.1), a informacije o boji (hrominacija) su date sa: 1) Cb-*Chroma Blue-difference* tj. razlika između plave komponente i osvetljenosti i 2) Cr-*Chroma Red-difference* tj. razlika između crvene komponente i osvetljenosti). Kod primene JPEG standarda, slika se deli na manje blokove veličine 8x8 na koje se primenjuje diskretna kosinusna transformacija (eng. *Discrete Cosine Transform, DCT*), a koeficijenti u DCT domenu se kvantizuju. S obzirom na to da je ljudsko oko više osetljivo na osvetljenje nego na boje, kanali o boji (Cb i Cr) se mogu više kvantizovati od Y kanala. Nakon kvantizacije se primenjuje neki od algoritama za kompresiju bez gubitaka npr. *RLE* ili Hafmanovo kodiranje (entropijsko kodiranje koje dodeljuje kraće binarne kodove simbolima koje se češće pojavljuju, a duže binarne kodove simbolima koji se ređe pojavljuju).

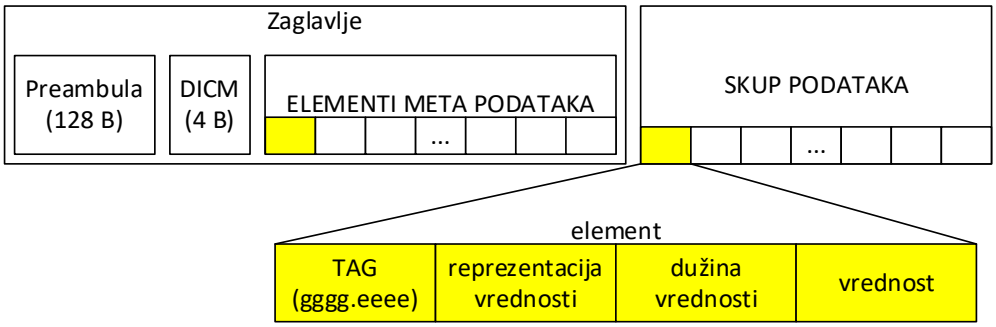
Formati posebno namenjeni medicinskim slikama pored zapisa 2D slika mogu imati podršku i za zapis 3D ili 4D slike. *Digital Imaging and Communication in Medicine (DICOM)* je najzastupljeniji format medicinskih slika i koristi se ne samo za arhiviranje podataka već i za komunikaciju između modaliteta medicinskog slikanja. *DICOM* je standard za većinu sistema za arhiviranje i komunikaciju tzv. *Picture Archiving and Communications Systems (PACS)*. Definisan je 1993. godine od strane *American College of Radiology* (*ACR*) i *National Electrical Manufacturers Association* (*NEMA*)[14]. Struktura DICOM datoteke sadrži, Sl. 3.3:

- zaglavlje datoteke koje se sastoji iz preambule od 128 bajtova, 4 bajta sa oznakom 'DICM' za DICOM format, elemenata meta podataka (informacije koje se koriste za identifikaciju datoteke, određivanje transfera sintakse i integraciju različitih uređaja i softvera) i
- skup podataka koji se sastoji iz elemenata kao što su informacije o pacijentu, studiji, seriji slika, slici kao i same piksele slike. Svaki

[14] DICOM standard je dostupan na sledećem linku: https://www.dicomstandard.org/ (poslednji pregled Decembar 2024).

element meta podataka i svaki element skupa podataka ima strukturu koja se sastoji iz:

- taga (broj grupe i elementa, npr. (0008,0060) za naziv modaliteta slikanja),
- reprezentacije vrednosti (tip podataka, npr. TM za vreme),
- dužine vrednosti elementa podataka,
- vrednosti elementa.



Slika 3.3. Struktura DICOM datoteke

Od ostalih specijalizovnih formata se izdvajaju: Analyze 7.5 (naziv potiče od softvera razvijenog od strane *Mayo Clinic*, SAD za obradu medicinskih slika), *Neuroimaging Informatics Technology Initiative* (NIfTI), *Nearly Raw Raster Data* (NRRD), *MetaImage Header/MetaImage Data* (MHA/MHD). Uporedni pregled karakteristika različitih specijalizovanih formata je dat u Tabeli 3.1.

Tabela 3.1. Pregled specijalizovanih formata podataka za medicinske slike

	DICOM	ANALYZE 7.5	NIFTI	NRRD	MHA/ MHD
Vrsta podataka	Klinički standard za slike	Neuroslike	Neuroslikanje standard	Multidimenzi-oni podaci	3D medicinske slike
Broj datoteka	1 (.dcm)	2 (.hdr i .img)	1 (.nii) ili 2 (.hdr i .img)	1 (.nrrd) ili 2 (.nrrd i .raw)	1 (.mha) ili 2 (.mhd i .raw)
Podaci o pacijentu	Polja u zaglavlju	-	-	-	-
Kompresija	Sa i bez gubitaka	-	-	-	-
Dimenzije	2D, 3D, 4D	3D	3D, 4D	3D, 4D	3D, 4D
Složenost	++	+	+	+	+
Integracija sa drugim sistemima	Da	Ograničena	Ograničena na neuroslikanje	Istraživačka primena	Istraživačka primena

3.3. Čitanje/upis i prikaz slika

U Prilogu A dat je pregled često korišćenih besplatnih alata za čitanje i vizuelizaciju biomedicinskih slika, zajedno sa linkovima za pristup.

3.3.1. Funkcije za čitanje/upis 2D slika

Biblioteka *OpenCV* podržava čitanje datoteka sa 2D slikama sa sledećim formatima/ekstenzijama: .bmp, .jpg, .jp2[15], .png, .pgm, .tiff. Takođe, biblioteka *OpenCV* podržava upis 2D slika u datoteke sledećih formata/ekstenzija: .jpeg, .png, .pgm, .tiff. Na Pr. 3.3A je prikazan programski kôd *opencv_citanje.py* koji pomoću funkcije `cv2.imread()` učitava datoteku *tekstualni_PGM.pgm* kreiranu u Pr. 3.2, a potom vrši upis pročitanih podataka u datoteke u .pgm i .png formatu pomoću funkcije `cv2.imwrite()`. Promenljiva `slika_RGB` u koju se smeštaju pročitani podaci matrice dimenzija 3x3 ima format (3,3,3) gde prve dve cifre „3“ odgovaraju dimenzijama matrice, a poslednja cifra „3“ odgovara trokanalnom RGB sistemu. PGM format datoteke zahteva *grayscale* sliku za upis, pa je zbog toga neophodno uraditi konverziju matrice `slika_RGB` u *grayscale* format pomoću funkcije:

```
cv2.cvtColor(slika_RGB, cv2.COLOR_BGR2GRAY).
```

Pr. 3.3. Primer *opencv_citanje.py* – čitanje/upis datoteke pomoću *OpenCV* biblioteke

---

A. Programski kôd:

```
import cv2
import matplotlib.pyplot as plt

# Čitanje podataka iz PGM datoteke
slika_RGB=cv2.imread('tekstualni_PGM.pgm')
plt.imshow(slika_RGB)

# Konverzija iz RGB sistema u grayscale
slika_grayscale=cv2.cvtColor(slika_RGB, cv2.COLOR_BGR2GRAY)
```

[15] .jp2 je ekstenzija za JPEG 2000 format koji koristi diskretnu talasnu transformaciju (eng. *Discrete Wavelet Transform, DWT*), ne primenjuje podelu na blokove i omogućava kompresiju bez gubitaka za razliku od JPEG formata.

```
# Upis u novu PGM datoteku
cv2.imwrite('opencv_tekstualni_PGM.pgm', slika_grayscale)

# Upis u PNG datoteku
cv2.imwrite('opencv_tekstualni_PGM.png', slika_RGB)
```

B. Rezultat izvršavanja:

Kreirane su datoteke opencv_tekstualni_PGM.pgm i opencv_tekstualni_PGM.png.

Osim *OpenCV* biblioteke, podrška za čitanje i upis 2D slika (tipa BMP, JPEG, PNG, PGM, TIFF i sl.) postoji i u sledećim *Python* bibliotekama: *imageio, scikit-image, PIL/Pillow, SimpleITK* itd. U Tabeli 3.2 su prikazane funkcije/metode za čitanje/upis slika različitih biblioteka, kao i formati slika koji su podržani svakom od biblioteka.

3.3.2. Čitanje/upis i prikaz DICOM slika

Python biblioteke koje se najčešće koriste za čitanje DICOM datoteka su: *imageio, scikit-image, pydicom* i *SimpleITK*. Na Pr. 3.4A je prikazan programski kôd *Imageio_citanje_CT_slika.py* koji pomoću funkcije `imageio.imread()` čita DICOM datoteku „0.dcm“. Ova datoteka je deo dostupne baze podataka *Computed Tomography (CT) of the Brain*[16], i nalazi se u podskupu ove baze koji sadrži CT tumore glave. Primenom *Numpy* funkcije `np.shape()` na učitanu matricu `slika` određuje se dimenzija izražena u broju piksela po *x* i po *y* osi (dimenzija je 512x512 piksela). Potom se pomoću metode `slika.meta` iz zaglavlja DICOM datoteke čitaju i prikazuju `meta_podaci` koji su tipa podataka rečnik. Prikaz vrednosti jednog od meta podataka koji se odnosi na ključ `Modality` je ilustrovan u primeru. Na kraju je prikazana učitana slika u *gray* paleti boja, a pomoću `plt.axis('off')` su ukinute oznake na *x* i *y* osama.

[16] Baza podataka sadrži CT snimke kancera, tumora i aneurizmi. Dostupna je na sledećem linku: https://www.kaggle.com/datasets/trainingdatapro/computed-tomography-ct-of-the-brain?resource=download (poslednji pregled Decembar 2024) i podeljena je pod CC BY-NC-ND 4.0 licencom (https://creativecommons.org/licenses/by-nc-nd/4.0/).

Tabela 3.2. Pregled *Python* biblioteka za čitanje/upis slika

	FUNKCIIJA ZA ČITANJE	FUNKCIJA/ METODA ZA UPIS	DICOM, NIfTI PODRŠKA	OSTALI FORMATI
OpenCV	*import cv2*		-	BMP, JPEG, JP2, PNG, PPM/PGM, TIFF, WEBP
	cv2.imread	*cv2.imwrite*		
Imageio	*import imageio*		DICOM, NIfTI	BMP, JPEG, PNG, TIFF, GIF, PPM/PGM, ICO, PSD, WEBP, MP4…
	imageio.imread	*imageio.imwrite*		
Scikit-image	*from skimage import io*		DICOM, NIfTI	BMP, JPEG, PNG, TIFF, GIF, PPM/PGM
	io.imread	*io.imsave*		
PIL/Pillow	*from PIL import Image*		-	BMP, JPEG, PNG, TIFF, GIF, ICO, PSD, PPM/PGM, WEBP
	Image.open	slika.*save*		
Pydicom	*import pydicom*		DICOM	-
	pydicom.dcmread	slika.*save_as*		
SimpleITK	*import SimpleITK as sitk*		DICOM, NIfTI	BMP, PNG, TIFF
	sitk.ReadImage	*sitk.WriteImage*		

Pr. 3.4. Primer *Imageio_citanje_CT_slika.py* – čitanje i prikaz DICOM datoteke sa CT snimkom glave

---

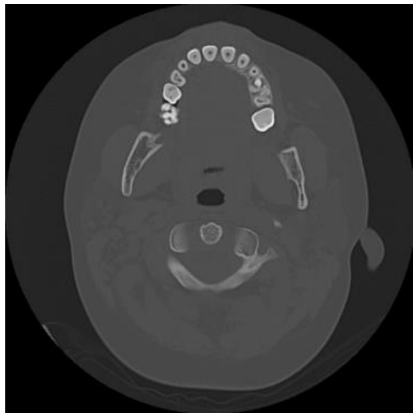
A. Programski kôd:

```
import imageio
import matplotlib.pyplot as plt
import numpy as np

plt.close("all")
# Čitanje DICOM datoteke
slika = imageio.imread('0.dcm')
# Prikaz dimenzija DICOM slike
dimenzija_slika = np.shape(slika)
print('Dimenzije učitane slike:',dimenzija_slika)
# Čitanje meta podataka
meta_podaci = slika.meta
print(meta_podaci.keys())
```

```
modalitet_slikanja = meta_podaci['Modality']
print('Modalitet slikanja:', modalitet_slikanja)


plt.figure()
plt.imshow(slika, cmap = 'gray')
plt.axis('off')
```

B. Rezultat izvršavanja:

```
Dimenzije učitane slike: (512, 512)
odict_keys(['TransferSyntaxUID', 'SOPClassUID', 'SOPInstanceUID', 'StudyDate', 'Modality',
'Manufacturer', 'InstitutionName', 'StudyDescription', 'SeriesDescription', 'PatientID',
'PatientSex', 'PatientAge', 'StudyInstanceUID', 'SeriesInstanceUID', 'SeriesNumber',
'AcquisitionNumber', 'InstanceNumber', 'PatientOrientation', 'ImagePositionPatient',
'ImageOrientationPatient', 'SamplesPerPixel', 'Rows', 'Columns', 'PixelSpacing', 'BitsAllocated',
'BitsStored', 'HighBit', 'PixelRepresentation', 'RescaleIntercept', 'RescaleSlope', 'PixelData',
'shape', 'sampling'])
Modalitet slikanja: CT
```



Na Pr. 3.5A je prikazan programski kôd *Imageio_citanje_CT_volumena.py* koji pomoću funkcije `imageio.volread()` čita DICOM datoteke u direktorijumu `CT_tumor` dostupnog u okviru baze podataka *Computed Tomography (CT) of the Brain* koja je korišćena i u prethodnom primeru Pr. 3.4. Primenom *Numpy* funkcije `np.shape()` na učitanu 3D matricu `volumen` određuje se dimenzija izražena u broju slika tj. slajseva (eng. *slice*) po *z* osi, i broju piksela po *x* i po *y* osi (dimenzija je 84x512x512, tj. volumen ima 84 slajsa dimenzija 512x512 piksela). Potom se iz meta podataka datoteke metodom `volumen.meta` iščitavaju fizičke dimenzije voksela (ključ `sampling` u rečniku meta podataka) `d0`, `d1` i `d2` tj. dimenzije voksela po *z*, *x* i *y* osi osi respektivno, u milimetrima (konkretno: *d0*=2 mm, *d1*=0.407 mm i *d2*=0.407 mm). Fizičke dimenzije volumena se proračunavaju množenjem dimenzija voksela sa dimenzijama volumena (npr. fizička dimenzija po *z* osi je dobijena množenjem broja slajseva sa `d0`). Za prikaz slika, neophodno je

uzeti u obzir odnose fizičkih dimenzija voksela za onu ravan u kojoj se prikaz vrši (moguće ravni su: frontalna ili koronarna, transverzalna ili aksijalna i sagitalna[17] ravan). Konačno, pomoću funkcije `plt.subplots()` čiji argumenti su broj redova i kolona matrice za prikaz slika (`nrows`=2, `ncols`=2) je formiran matrični prikaz 2x2 slike gde je:

- na poziciji [0,0] prikaz 150. slajsa u frontalnoj ravni pri čemu nije uzeto u obzir skaliranje spram različitih fizičkih dimenzija u ravni (*x*,*z*)
- na poziciji [0,1] prikaz 150. slajsa u frontalnoj ravni pri čemu je uzeto u obzir skaliranje spram različitih fizičkih dimenzija u ravni (*x*,*z*) pomoću argumenta `aspect=d0/d2`
- na poziciji [1,0] prikaz 15. slajsa u transverzalnoj ravni pri čemu je uzeto u obzir skaliranje spram različitih fizičkih dimenzija u ravni (*x*,*y*) pomoću argumenta `aspect=d1/d2` (u ovom konkretnom slučaju je `d1=d2`, pa bi i neskalirani prikaz bio identičan skaliranom)
- na poziciji [1,1] prikaz 256. slajsa u sagitalnoj ravni pri čemu je uzeto u obzir skaliranje spram različitih fizičkih dimenzija u ravni (*y*,*z*) pomoću argumenta `aspect=d0/d1` .

Pomoću *Numpy* funkcije `np.flipud()` je urađeno vertikalno flipovanje (eng. *flip up-down*) za frontalnu i sagitalnu ravan. Naslovi i natpisi oznaka za ose su podešeni pomoću `set_title`, `set_xlabel` i `set_ylabel`, respektivno.

Pr. 3.5. Primer *Imageio_citanje_CT_volumena.py* – čitanje i prikaz DICOM datoteka iz direktorijuma sa CT snimcima glave

---

A. Programski kôd:

```
import imageio
import matplotlib.pyplot as plt
import numpy as np
import os


plt.close("all")
# Podešavanje putanje do direktorijuma sa slikama
```

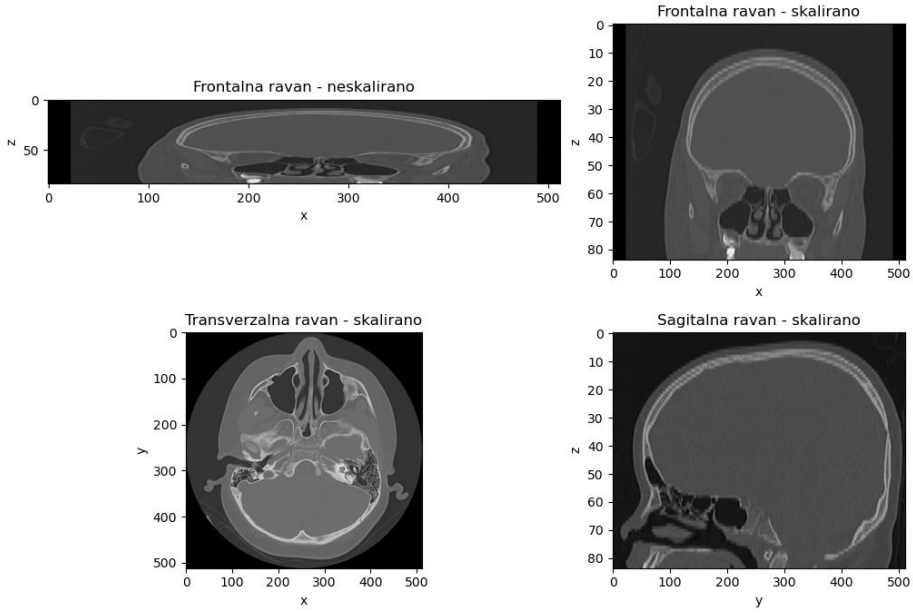
[17] Frontalna (koronarna) ravan deli telo na prednji (eng. *anterior*) i zadnji (eng. *posterior*) deo. Transverzalna (aksijalna) ravan deli telo na gornji (eng. *superior*) i donji (eng. *inferior*) deo. Sagitalna ravan deli telo na levu i desnu polovinu tela.

```
trenutni_dir= os.getcwd()
print(trenutni_dir)
putanja_CT_direktorijum = os.path.join(
    trenutni_dir, 'CT_tumor')

# Čitanje DICOM volumena
volumen=imageio.volread('CT_tumor','DICOM')
# Prikaz dimenzija volumena
dimenzija_volumen = np.shape(volumen)
print('Dimenzije volumena:', dimenzija_volumen)
# Prikaz dimenzija voksela
d0,d1,d2 = volumen.meta['sampling']
print('Dimenzija voksela po z osi [mm]:', d0)
print('Dimenzija voksela po x osi [mm]:', d1)
print('Dimenzija voksela po y osi [mm]:', d2)
z_fizicka_dimenzija = dimenzija_volumen[0]*d0
x_fizicka_dimenzija = dimenzija_volumen[1]*d1
y_fizicka_dimenzija = dimenzija_volumen[2]*d2
print('Volumen - fizička dimenzija z [mm]:',
      z_fizicka_dimenzija)
print('Volumen - fizička dimenzija x [mm]:',
      x_fizicka_dimenzija)
print('Volumen - fizička dimenzija y [mm]:',
      y_fizicka_dimenzija)

# Prikaz frontalne, transverzalne i sagitalne ravni
fig,axes=plt.subplots(nrows=2, ncols=2)
axes[0,0].imshow(np.flipud(volumen[:,150,:]), cmap='gray')
axes[0,0].set_title('Frontalna ravan - neskalirano')
axes[0,0].set_xlabel('x')
axes[0,0].set_ylabel('z')
axes[0,1].imshow(np.flipud(volumen[:,150,:]),
                 cmap='gray', aspect=d0/d2)
axes[0,1].set_title('Frontalna ravan - skalirano')
```

```
axes[0,1].set_xlabel('x')
axes[0,1].set_ylabel('z')
axes[1,0].imshow(volumen[15,:,:],
                 cmap='gray', aspect=d1/d2)
axes[1,0].set_title('Transverzalna ravan - skalirano')
axes[1,0].set_xlabel('x')
axes[1,0].set_ylabel('y')
axes[1,1].imshow(np.flipud(volumen[:,:,256]),
                 cmap='gray', aspect=d0/d1)
axes[1,1].set_title('Sagitalna ravan - skalirano')
axes[1,1].set_xlabel('y')
axes[1,1].set_ylabel('z')
```

B. Rezultat izvršavanja:

```
Reading DICOM (examining files): 168/168 files (100.0%
  Found 1 correct series.
Reading DICOM (loading data): 84/84  (100.0%)
Dimenzije volumena: (84, 512, 512)
Dimenzija voksela po z osi [mm]: 2.0
Dimenzija voksela po x osi [mm]: 0.407
Dimenzija voksela po y osi [mm]: 0.407
Volumen - fizička dimenzija z [mm]: 168.0
Volumen - fizička dimenzija x [mm]: 208.384
Volumen - fizička dimenzija y [mm]: 208.384
```



Na Pr. 3.6A je prikazan *SimpleITK_citanje_dinamika.py* programski kôd koji pomoću funkcije `sitk.ReadImage()` čita DICOM datoteku „*drsbru_001_POST.dcm*“ dinamske renološke studije iz baze sa otvorenim pristupom *Database of dynamic renal scintigraphy - drsbru*[18] [Tondeur et al. 2013]. Datoteka sadrži 120 slika bubrega formata 128x128 snimanih u toku dinamske scintigrafije nakon aplikacije radiofarmaka ^{99m}Tc-MAG3 intravenski u organizam, pri čemu je trajanje snimanja svake slike tj. frejma (eng. *frame*) 10 s. U primeru Pr. 3.6 je omogućeno interaktivno listanje kroz učitane 3D podatke korišćenjem dostupne klase `IndexTracker_mod_mod`[19] biblioteke *Matplotlib* [Hunter 2007]. Ova klasa sadrži konstruktor `_init_`(inicijalizuje početne vrednosti objekta klase), metodu `onscroll` (registrovanje događaja skrolovanja miša) i metodu `update` (ažurira prikaz slajsa). Nakon inicijalizacije objekta `tracker` sa učitanim volumenom, događaj skrolovanja miša se povezuje sa metodom `onscroll` klase `IndexTracker_mod_mod` pomoću *Matplotlib* metode:

```
fig.canvas.mpl_connect('scroll_event', tracker.onscroll).
```

Pr. 3.6. Primer *SimpleITK_citanje_dinamika.py* – čitanje i interaktivni prikaz DICOM datoteke koja sadrži niz slika tj. frejmova

---

A. Programski kôd:

```
# Ovaj fajl koristi se pod PSF licencom.
# Pogledajte PSF-license.txt za detalje.
import SimpleITK as sitk
import matplotlib.pyplot as plt

plt.close('all')

class IndexTracker_mod_mod(object):
    def __init__(self, ax, X):
        self.ax = ax
        ax.set_title('Koristiti scroll na mišu za listanje')
```

[18]Baza *Database of dynamic renal scintigraphy* je dostupna zahvaljujući: https://dynamicrenalstudy.org (poslednji pregled Decembar 2024).

[19] Kôd za kreiranje klase je preuzet i modifikovan sa sledećeg linka: https://matplotlib.org/2.1.2/gallery/animation/image_slices_viewer.html (poslednji pregled Decembar 2024), a dostupan je pod PSF licencom.

```
        self.X = X
        self.slices, rows, cols = X.shape
        self.ind = self.slices//2

        self.im = ax.imshow(self.X[self.ind, :, :],
                            cmap='gray')
        self.update()

    def onscroll(self, event):
        print("%s %s" % (event.button, event.step))
        if event.button == 'up':
            self.ind = (self.ind + 1) % self.slices
        else:
            self.ind = (self.ind - 1) % self.slices
        self.update()

    def update(self):
        self.im.set_data(self.X[self.ind, :, :])
        ax.set_ylabel('frejm %s' % self.ind)
        self.im.axes.figure.canvas.draw()

# Ucitavanje DICOM slike
slika = sitk.ReadImage("drsbru_001_POST.dcm", sitk.sitkInt16)
slika_matrica = sitk.GetArrayFromImage(slika)

X = slika_matrica
fig, ax = plt.subplots(1, 1)
tracker = IndexTracker_mod_mod(ax, X)

fig.canvas.mpl_connect('scroll_event', tracker.onscroll)
plt.show()
```

B. Rezultat izvršavanja:

Rezultat je prikaz frejmova koji se izlistavaju tako što se skroluje mišem.

3.4. Rezime poglavlja

- Bitska dubina je broj bita koji omogućava zapis vrednosti piksela/voksela slike u binarnom obliku.
- Paleta boja je niz boja koji sadrži onoliko elemenata koliko ima mogućih vrednosti piksela/voksela slike. Za biomedicinsko slikanje, najčešće korišćene palete boja su: *gray, binary, rainbow, hot, jet, bone, viridis.*
- RGB sistem definiše boje kao uređenu trojku intenziteta tri osnovne boje: crvene (eng. *red*) – R, zelene (eng. *green*) – G i plave (eng. *blue*) – B
- Biomedicinske slike se mogu zapisati u datoteku korišćenjem sirovih formata podataka (PGM, BMP), formata sa kompresijom bez gubitaka (npr. PNG, TIFF), formata sa kompresijom sa gubicima (npr. JPEG) ili specijalizovanih formata za arhiviranje 2D, 3D ili 4D slika (DICOM, Analyze 7.5, NifTI, NRRD, MHA/MHD itd.).
- *Python* biblioteke koje se najčešće koriste za čitanje DICOM datoteka su: *imageio, scikit-image, pydicom* i *SimpleITK.*

4. TRANSFORMACIJE INTENZITETA

Transformacije intenziteta piksela omogućavaju poboljšanje vizuelnog prikaza slike tako što transformišu vrednosti piksela iz opsega [O_1,O_2] primenom funkcije transformacije intenziteta (eng. *Intensity Transform Function, ITF*) u opseg intenziteta piksela [N_1,N_2]. Transformacije intenziteta je moguće primeniti i na samo određeni podopseg vrednosti intenziteta piksela originalne slike, pri čemu se u tom slučaju prilikom vizuelizacije ostatak vrednosti intenziteta piksela postavlja na nulu. Ovakav vid primene transformacija se naziva „prozorovanje“ (eng. *windowing*) i posebno je koristan kod slika kod kojih je na osnovu vrednosti intenziteta piksela moguće segmentirati određena tkiva (npr. kod *CT* slika) što će detaljnije biti opisano u *Poglavlju 6 Segmentacija*.

4.1. Kontrast

Kontrast slike je razlika između najsvetlijih i najtamnijih delova slike. Slika sa visokim kontrastom ima veliki raspon intenziteta vrednosti piksela, a detalji na takvoj slici su jasno vidljivi. Kontrast *K* se može matematički izraziti na različite načine, kao na primer:

- mera raspona vrednosti intenziteta piksela unutar slike:

$$K = \rho_{max} - \rho_{min} \tag{4.1}$$

  gde su ρ_{max} i ρ_{min} maksimalna i minimalna vrednost intenziteta piksela na slici *I*, respektivno.

- standardna devijacija vrednosti intenziteta piksela na slici:

$$K = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(\rho_i - \rho_{sr})^2} \tag{4.2}$$

  gde je *N* ukupan broj piksela na slici *I*, ρ_i su vrednosti intenziteta piksela, a ρ_{sr} je srednja vrednost intenziteta piksela.

- odnos raspona intenziteta piksela između objekta na slici i pozadine:

$$K = \frac{\rho_{sr\_objekat} - \rho_{sr\_pozadina}}{\rho_{sr\_pozadina}} \tag{4.3}$$

  gde je $\rho_{sr\_objekat}$ srednja vrednost intenziteta piksela objekta na slici, a $\rho_{sr\_pozadina}$ je srednja vrednost intenziteta piksela pozadine.

- odnos raspona intenziteta piksela na slici i standardne devijacije pozadine:

$$K = \frac{\rho_{max} - \rho_{min}}{\sigma_{pozadina}} \quad (4.4)$$

gde su ρ_{max} i ρ_{min} maksimalna i minimalna vrednost intenziteta piksela na slici *I*, respektivno, a $\sigma_{pozadina}$ je standardna devijacija vrednosti intenziteta piksela pozadine.

Različite transformacije vrednosti intenziteta piksela (linearna, gama, logaritamska, sigmoidna transformacija, ekvalizacija histograma i dr.) mogu uticati na promenu tj. poboljšanje kontrasta slike (lokalno, tj. za deo slike ili globalno, tj. za celu sliku) i biće detaljnije opisane u narednim poglavljima.

4.2. Linearna transformacija intenziteta

Najčešće korišćena transformacija intenziteta u biomedicinskom slikanju je linearno mapiranje vrednosti intenziteta piksela iz opsega [O_1,O_2] u opseg [N_1,N_2], tj. linearna transformacija intenziteta kod koje je *ITF* definisana na sledeći način:

$$\rho_{xy\_lin} = \frac{\rho_{xy} - \rho_{min}}{\rho_{max} - \rho_{min}} opseg + \rho_{xy\_lin\_min} = A \cdot opseg + B \quad (4.5)$$

gde je ρ_{xy_lin} linearno transformisana vrednost intenziteta piksela *(x,y)*, ρ_{xy}, ρ_{min}, ρ_{max} su redom originalna, minimalna i maksimalna vrednost intenziteta piksela *(x,y)* u originalnom opsegu [O_1,O_2] i $\rho_{xy\_lin\_min}$ je minimalna vrednost intenziteta piksela u novom opsegu [N_1,N_2]. Ako je *N1=0*, *opseg* je novi maksimum opsega vrednosti intenziteta piksela *N2*. Ako je *N1<0* novi opseg vrednosti intenziteta piksela na transformisanoj slici je *N2-N1+1*. Na primer, ako je željeni opseg transformacije [0,255], *opseg* će biti jednak *N2*=255, a ako je željeni opseg transformacije [-127,127] onda je *opseg* jednak *N2-N1*+1=127-(-127)+1= 255.

Linearna transformacija omogućava: 1) rastezanje (ili sabijanje) kontrasta tj. normalizaciju slike na zadati novi opseg vrednosti intenziteta piksela i 2) podešavanje osvetljenja zadavanjem željene minimalne vrednosti intenziteta piksela na transformisanoj slici (B=$\rho_{xy\_lin\_min}$) – za B>0 transformisana slika je svetlija u odnosu na originalnu sliku, a za B<0 transformisana slika postaje tamnija u odnosu na originalnu sliku.

4.3. Inverzna slika

Najjednostavniji oblik transformacije intenziteta je *ITF* koja omogućava da se „tamni“ pikseli (obično pikseli sa malim vrednostima intenziteta) transformišu u „svetle“ piksele (obično pikseli sa velikim vrednostima intenziteta) i obrnuto. Ovakvom transformacijom se dobija tzv. inverzna slika, a transformacija se primenjuje radi poboljšanja vizuelne

percepcije detalja u svetlim segmentima slike. Za sliku kod koje je originalni opseg intenziteta piksela $[O_1,O_2]=[0,O_2]$ *ITF* će imati sledeći oblik:

$$\rho_{xy\_inv} = O_2 - \rho_{xy} \tag{4.6}$$

gde su ρ_{xy} i ρ_{xy_inv} originalna i transformisana vrednost intenziteta piksela *(x,y),* respektivno. Za slike koje imaju negativne vrednosti za O_1 potrebno je pre primene inverza najpre uraditi transformaciju koja će vrednosti intenziteta piksela translirati u nenegativne (kao na primer kod *CT* slika). Često se pre primene inverzne transformacija radi najpre normalizacija na opseg vrednosti intenziteta piksela [0,255], tj. linearna transformacija na opseg [0,255] kao što je to opisano u prethodnom *Poglavlju 4.2*. Nakon primene linearne transformacije se primenjuje *ITF* za inverziju koja onda ima sledeći oblik:

$$\rho_{xy\_inv} = 255 - \rho_{xy\_lin} \tag{4.7}$$

gde je ρ_{xy_lin} linearno transformisana vrednost piksela *(x,y)* na opseg [0,255], a ρ_{xy_inv} je njena inverzno transformisana vrednost.

4.4. Gama korekcija intenziteta

Gama korekcija intenziteta primenjuje *ITF* koja se definiše na sledeći način:

$$\rho_{xy\_gama} = c_{gama} \cdot \rho_{xy\_norm}{}^{\gamma} \tag{4.8}$$

gde je c_{gama} konstanta skaliranja, ρ_{xy_norm} je normalizovana vrednost piksela *(x,y)* na opseg [0,1], a ρ_{xy_gama} je transformisana vrednost piksela *(x,y)* gama korekcijom.

Za $\gamma<1$ sve vrednosti intenziteta piksela se povećavaju gama transformacijom, a posebno male vrednosti intenziteta piksela doživljavaju značajan rast (eksponencijalna funkcija povećava male vrednosti proporcionalno više nego velike vrednosti). Za $\gamma>1$ sve vrednosti intenziteta piksela se smanjuju gama transformacijom, a posebno male vrednosti intenziteta piksela doživljavaju značajan pad (eksponencijalna funkcija smanjuje male vrednosti proporcionalno više nego velike vrednosti). Prema tome, za $\gamma<1$ slika transformisana gama korekcijom postaje svetlija (tamni delovi se naglašavaju), a za $\gamma>1$ slika transformisana gama korekcijom postaje tamnija.

4.5. Logaritamska transformacija intenziteta

Logaritamska transformacija vrednosti intenziteta piksela koristi logaritamsku *ITF* prema sledećoj formuli:

$$\rho_{xy\_log} = c_{log} \cdot \log\left(1 + \rho_{xy\_norm}\right) \tag{4.9}$$

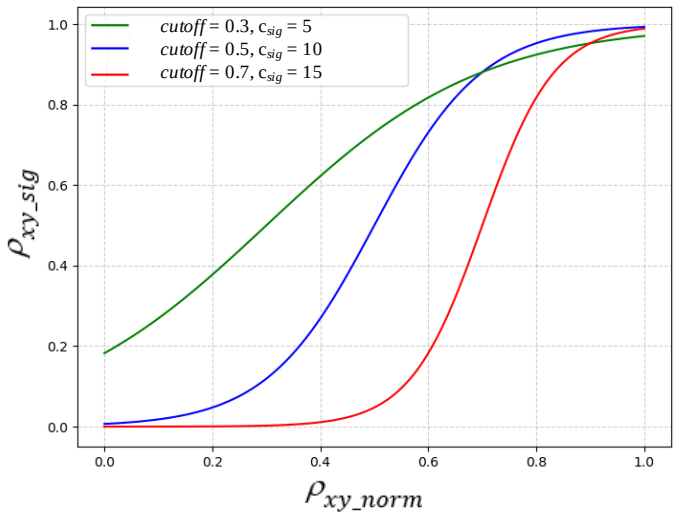
gde je c_{log} konstanta skaliranja, ρ_{xy_norm} je normalizovana vrednost piksela *(x,y)* na opseg [0,1], a ρ_{xy_log} je vrednost intenziteta piksela *(x,y)* na logaritamski transformisanoj slici (u logaritamskoj funkciji se dodaje 1 na ρ_{xy} kako bi se za ρ_{xy}=0 izbeglo računanje logaritma od nulte vrednosti). Ovakva transformacija naglašava tamne delove slike, tj. delove slike sa nižim vrednostima intenziteta piksela.

4.6. Sigmoidna transformacija intenziteta

Sigmoidna transformacija intenziteta primenjuje tzv. sigmoidnu (logističku) funkciju za lokalno poboljšavanje kontrasta, tj. za određeni opseg vrednosti intenziteta piksela, uz kontrolisanje stepena promene kontrasta. *ITF* je definisana na sledeći način, Sl. 4.1:

$$\rho_{xy\_sig} = \frac{1}{1+e^{-c_{sig}\cdot(\rho_{xy\_norm}-cutoff)}} \tag{4.10}$$

gde je ρ_{xy_norm} normalizovana vrednost piksela *(x,y)* na opseg [0,1], c_{sig} je parametar koji definiše nagib sigmoidne funkcije (tj. stepen strmine prelaza izmedju tamnih i svetlih oblasti), *cutoff* je parametar koji definiše centralnu vrednost ρ_{xy_norm} oko koje se odvija najveća promena kontrasta, a ρ_{xy_sig} je vrednost intenziteta piksela *(x,y)* na sigmoidno transformisanoj slici.



Slika 4.1. *ITF* za tri sigmoidne krive sa različitim centralnim intenzitetom (*cutoff*) i različitim nagibima (c_{sig})

Na Sl. 4.1 su prikazane *ITF* za tri sigmoidne funkcije sa različitim centralnim intenzitetima (*cutoff*) oko kojih se dešava transformacija kontrasta (podešenim na 0.3; 0.5; 0.7) i sa različitim strminama (c_{sig}) promene kontrasta (podešenim na 5; 10; 15 – sa porastom ovog parametra strmina raste).

4.7. Ekvalizacija histograma

4.7.1. Histogram slike

Histogram slike je grafički prikaz raspodele vrednosti piksela. Apscisa histograma prikazuje vrednosti intenziteta piksela grupisane u tzv. binove tj. opsege vrednosti piksela, a ordinata histograma prikazuje učestanost pojavljivanja piksela tj. broj piksela sa određenim vrednostima.

Na Pr. 4.1A je dat programski kôd koji na primeru slike dimenzija 10x10 piksela (datoteka „mala_slika.png“) izračunava i prikazuje histogram sa 16 binova i sa 256 binova pomoću *Matplotlib* funkcije `plt.hist()` kojoj se kao argumenti zadaju: slika pretvorena u niz (pomoću funkcije `slika.ravel()`), broj binova (argument `bins`) i opseg intenziteta (`range=(0, 255)`). U rezultatu izvršavanja Pr. 4.1B se može uočiti da kod histograma sa 16 binova, jedan bin obuhvata sve vrednosti intenziteta piksela date slike. Na Pr. 4.1B su dati i prikazi originalne slike sa paletom boja u opsegu 0-255 (`vmin=0`, `vmax=255` u `imshow()` funkciji) i sa automatski podešenom paletom boja (`Originalna slika - skalirano`) spram minimalnog i maksimalnog intenziteta piksela originalne slike (opseg 50-57).

Pr. 4.1. Primer *prikaz_histograma.py* – ilustracija efekata različitih transformacija intenziteta

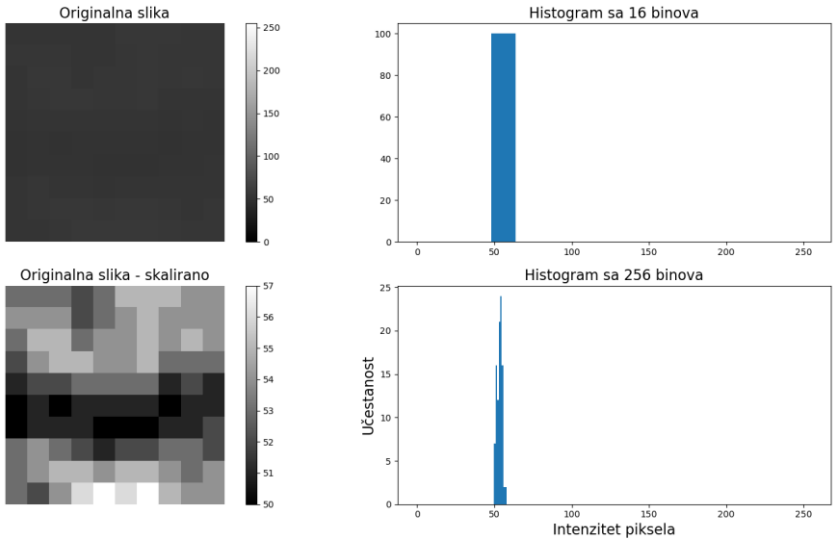
---

A. Programski kôd:

```
import matplotlib.pyplot as plt
import cv2

plt.close('all')
# Ucitavanje PNG slike
slika_RGB=cv2.imread('mala_slika.png')
# Konverzija iz RGB sistema u grayscale
slika=cv2.cvtColor(slika_RGB, cv2.COLOR_BGR2GRAY)

# Prikaz histograma sa 16 i 256 binova
```

```
plt.subplot(2, 2, 1)
plt.title('Originalna slika', fontsize=16)
plt.imshow(slika, cmap='gray', vmin=0, vmax=255)
plt.axis('off'); plt.colorbar()
plt.subplot(2, 2, 2)
plt.title('Histogram sa 16 binova', fontsize=16)
plt.hist(slika.ravel(), bins=16, range=(0, 255))
plt.subplot(2, 2, 3)
plt.title('Originalna slika - skalirano', fontsize=16)
plt.imshow(slika, cmap='gray')
plt.axis('off'); plt.colorbar()
plt.ylabel('Učestanost', fontsize=16)
plt.subplot(2, 2, 4)
plt.title('Histogram sa 256 binova', fontsize=16)
plt.hist[20](slika.ravel(), bins=256, range=(0, 255))
plt.xlabel('Intenzitet piksela', fontsize=16)
plt.ylabel('Učestanost', fontsize=16)
```

B. Rezultat izvršavanja:



[20] Identičan prikaz se može dobiti i primenom *Numpy* funkcije `np.histogram`:

```
histogram, bin_granice = np.histogram(slika, bins=256, range=(0, 255))
plt.bar(bin_granice[:-1], histogram)
```

Idealan histogram slike bi trebalo da za najmanje vrednosti intenziteta piksela ima nisku učestanost pojavljivanja (u suprotnom to može biti indikator nedostajućih vrednosti tj. vrednosti koje nisu snimljene), kao i da za najveće vrednosti intenziteta piksela takođe ima nisku učestanost pojavljivanja (u suprotnom to može biti indikator zasićenja).

4.7.2. Algoritam ekvalizacije histograma

Ekvalizacija histograma je vrsta transformacije intenziteta koja se primenjuje radi globalnog poboljšanja kontrasta na slici. Cilj ekvalizacije histograma je postizanje ravnomerne raspodele vrednosti intenziteta piksela u celom opsegu. Na Sl. 4.2A1 je prikazan primer slike malih dimenzija 10x10 sa niskim kontrastom. Koraci algoritma ekvalizacije histograma će biti ilustrovani na ovom primeru, uz uporedni prikaz originalne i ekvalizovane slike, Sl. 4.2. Originalna i ekvalizovana slika su prikazane na Sl. 4.2A1 i Sl. 4.2B1, respektivno. Matrični oblici originalne slike i ekvalizovane slike su prikazani na Sl. 4.2A2 i Sl. 4.2B2, respektivno. Prikazi histograma/kumulativnog histograma za originalnu i ekvalizovanu sliku su dati na Sl. 4.2A3 i Sl. 4.2B3, respektivno.

Algoritam ekvalizacije histograma obuhvata sledeće korake:

- izračunavanje histograma originalne slike *H(i)* kao učestanosti tj. broja piksela sa intenzitetom *i*, Sl. 4.2A3 (crveni grafik).
- izračunavanje kumulativnog histograma *KH(i)* kao broja piksela sa intenzitetom manjim ili jednakim *i*, Tabela 4.1 i Sl. 4.2A3 (plavi grafik):

$$KH(i) = \sum_{j=0}^{i} H(j) \tag{4.11}$$

Tabela 4.1. Proračun učestanosti pojavljivanja intenziteta piksela, kumulativnog histograma i ekvalizovane vrednosti za originalnu sliku prikazanu na Sl. 4.2A1

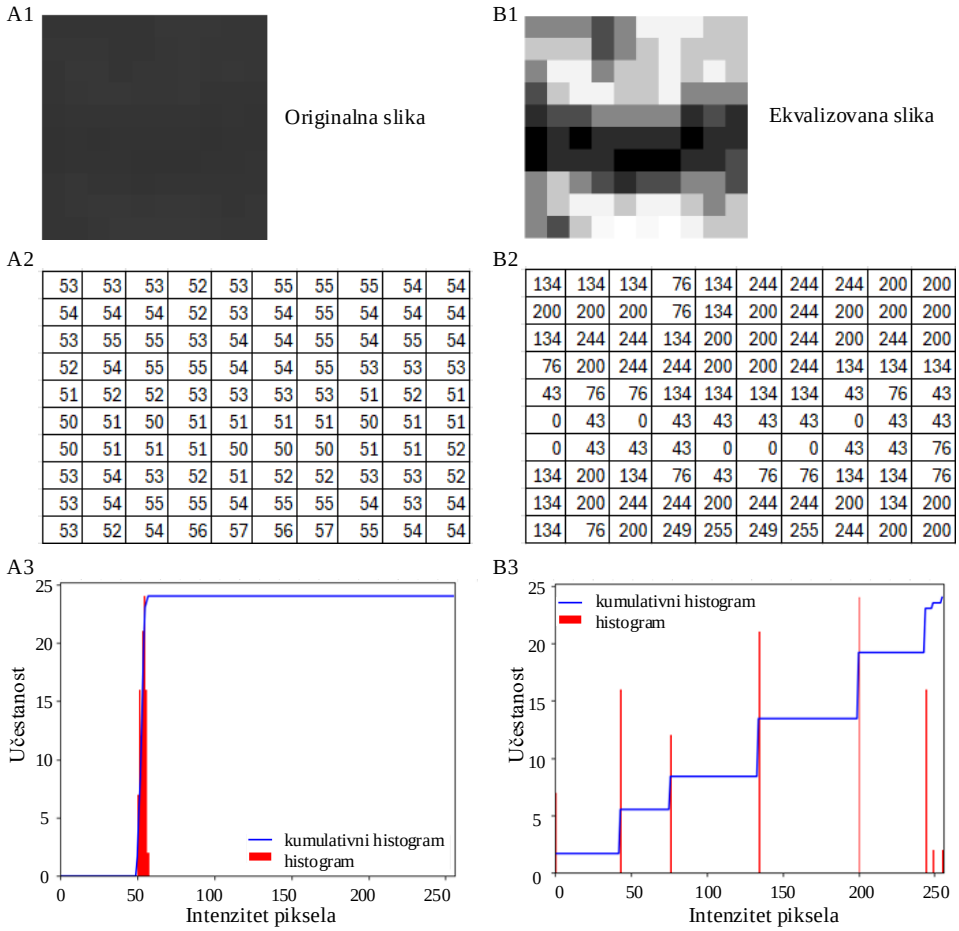
intenzitet piksela, *i*	Učestanost, *H*	kumulativni histogram, *KH*	ekvalizovana vrednost, *EH*
50	7	7	0
51	16	23	44
52	12	35	77
53	21	56	134
54	24	80	200
55	16	96	244
56	2	98	250
57	2	100	255

- izračunavanje ekvalizovane vrednosti intenziteta *EH(i)*, Tabela 4.1:

$$EH\ (i) = round\left(\frac{KH(i) - KH_{min}}{KH_{max} - KH_{min}} \cdot (L-1)\right) \qquad (4.12)$$

gde su KH_{min} i KH_{max} minimalna nenulta vrednost i maksimalna vrednost kumulativnog histograma respektivno, *KH(i)* je vrednost kumulativnog histograma za piksel intenziteta *i*, a *L* je maksimalni intenzitet ekvalizovane slike (npr. 256 za 8-bitne slike). *Round* označava da će rezultat jednačine biti zaokruženi ceo broj.

- mapiranje ekvalizovanih vrednosti intenziteta tj. zamena svakog originalnog nenultog intenziteta *i* na *(x,y)* poziciji, sa proračunatom ekvalizovanom vrednošću *EH(i)*, Sl. 4.3B1.



Slika 4.2. A1) Originalna slika, B2) Matrica originalne slike, A3) Kumulativni histogram i histogram originalne slike, B1) Ekvalizovana slika, B2) Matrica ekvalizovane slike, B3) Kumulativni histogram i histogram ekvalizovane slike.

4.7.3. Adaptivna ekvalizacija histograma

Adaptivna ekvalizacija histograma (eng. *Contrast Limited Adaptive Histogram Equalization, CLAHE*) primenjuje ekvalizaciju histograma na male regione slike, za razliku od algoritma opisanog u prethodnom Poglavlju 4.7.2, gde je ekvalizacija bila primenjena na celoj slici. Algoritam CLAHE obuhvata sledeće korake:

Korak 1. Slika se deli na manje regione (obično je *default*-na vrednost 8x8 piksela).

Korak 2. Za svaki region se proračunava histogram intenziteta piksela.

Korak 3. Postavlja se prag za ograničavanje histograma i ukoliko je broj piksela u binu veći od praga, onda se pikseli iznad praga ravnomerno raspodeljuju po celom histogramu – na ovaj način se sprečava preveliko povećanje kontrasta zbog ekstremnih vrednosti.

Korak 4. Za svaki piksel u okviru svakog regiona se proračunava ekvalizovana vrednost prema algoritmu opisanom u prethodnom Poglavlju 4.7.2.

Korak 5. Na granicama regiona se primenjuje interpolacija da bi se izbegli artefakti na spojevima regiona.

Pomoću *CLAHE* algoritma je omogućeno lokalno poboljšanje kontrasta i izbegavanje da ekstremne vrednosti piksela imaju veliki uticaj na promenu kontrasta. S obzirom na to da se pojačavaju lokalne razlike, *CLAHE* može uticati na povećanje vidljivosti šuma.

4.8. Primer primene transformacija intenziteta

Na Pr. 4.2A je prikazan programski kôd koji pomoću *exposure* modula *skimage* biblioteke ilustruje primenu svih načina transformacije intenziteta opisanih u ovom pogljavlju na DICOM slici „0.dcm“ korišćenoj i u *Poglavlju 3.3.2*. U pitanju je *CT* slika čija je minimalna vrednost intenziteta piksela za dati primer -2048 HU[21], a maksimalna vrednost 4702 HU. Najpre se vrši normalizacija slike na standardni opseg vrednosti intenziteta piksela [0,255], neoznačenog celobrojnog tipa sa 8 bita (`astype(np.uint8)`) pomoću

[21] HU je oznaka za Haunsfildovu jedinicu (eng. *Hounsfield Unit*) što je mera gustine tkiva na *CT* slikama. Gustina u [HU] se definiše kao $1000 \cdot (\mu_{tkivo}-\mu_{voda})/\mu_{voda}$ gde su μ_{tkivo} i μ_{voda} koeficijenti apsorpcije rendgenskih zraka za tkivo i vodu, respektivno. Različiti materijali imaju različite vrednosti gustine u [HU]. Tipične vrednosti gustine u [HU] su: ~ -1000 HU za vazduh, ~[-100, 50] HU za masti, ~[10, 40] HU za meka tkiva, ~[200, 1500] HU za kosti itd. Vrednost -2048 HU se obično koristi za označavanje piksela van dinamičkog opsega *CT* uređaja.

funkcije `exposure.rescale_intensity()` sa odgovarajućim argumentom `out_range=(0,255)`. Potom se redom primenjuju transformacije intenziteta:

- linearna transformacija vrednosti intenziteta piksela (*Poglavlje 4.2*) pomoću funkcije `exposure.rescale_intensity()` gde se zadaje „prozorovanje“ pomoću argumenta `in_range=(50,100)`, tj. samo na opsegu intenziteta piksela [50,100] se vrši linearna transformacija intenziteta, a sve ostale vrednosti piksela se na transformisanoj slici prikazuju sa vrednostima nula.
- inverzna transformacija vrednosti intenziteta piksela (Poglavlje 4.3) pomoću funkcije `util.invert()`
- gama korekcija (Poglavlje 4.4) koja se vrši pomoću funkcije `exposure.adjust_gamma`() sa argumentom γ=0.5 (funkcija ne poseduje argument za podešavanje skalarne konstante c_{gama} iz jednačine (4.8)). Ova funkcija normalizuje originalnu sliku na njenu maksimalnu vrednost i na njoj primenjuje jednačinu (4.8), a potom vraća izračunate intenzitete u opseg originalne slike.
- logaritamska transformacija vrednosti intenziteta piksela (Poglavlje 4.5) pomoću funkcije `exposure.adjust_log()` pri čemu je konstanta skaliranja c_{log} iz jednačine (4.9) postavljena na vrednost 1 (`gain=1`). Ova funkcija normalizuje originalnu sliku na njenu maksimalnu vrednost i na njoj primenjuje jednačinu (4.9), a potom vraća izračunate intenzitete u opseg originalne slike.
- sigmoidna transformacija vrednosti intenziteta piksela (Poglavlje 4.6) pomoću funkcije `exposure.adjust_sigmoid()`. Ova funkcija normalizuje originalnu sliku na njenu maksimalnu vrednost i na njoj primenjuje jednačinu (4.10), pri čemu je za centralni intenzitet piksela izabrana vrednost 0.3, a za nagib c_{sig} iz jednačine (4.10) vrednost 10 (argumenti `cutoff=0.3`, `gain=10`). Potom vraća izračunate intenzitete u opsegu originalne slike.
- ekvalizacija histograma intenziteta piksela (Poglavlje 4.7.2) pomoću funkcije `exposure.equalize_hist()` nakon koje je dobijeni opseg intenziteta [0,1] preskaliran ponovo na opseg [0,255] pomoću `img_as_ubyte`.
- adaptivna ekvalizacija histograma intenziteta piksela (*CLAHE*, Poglavlje 4.7.3) koju je moguće izvršiti pomoću funkcije `exposure.equalize_adapthist()`, nakon koje je dobijeni opseg intenziteta [0,1] preskaliran ponovo na opseg [0,255] pomoću

`img_as_ubyte`. Argument `clip_limit=0.1` definiše prag za ograničenje histograma, tj. za maksimalnu učestanost binova histograma. Za unapred zadatu veličinu malih regiona (`kernel_size=(10,10)`), prag će biti *clip_limit* x *kernel_size*=0.1x10x10=10, što znači da nijedan bin histograma neće imati veću učestanost od 10 u *Koraku 2 CLAHE* algoritma.

Na Pr. 4.2B je prikazan rezultat izvršavanja programskog kôda. Normalizovana slika prikazuje originalno učitanu *CT* sliku glave, ali skaliranu na standardni opseg intenziteta [0,255]. Uočljivo je da je linearna transformacija u opsegu [50,100] poboljšala kontrast u ovom opsegu intenziteta piksela i da je eliminisan prikaz piksela izvan glave pacijenta. Gama korekcija sa argumentum γ=0.5<1 je uticala na formiranje svetlije slike gde su tamniji detalji došli više do izražaja. Logaritamska transformacija je dala kao rezultat malo svetliju sliku u odnosu na originalnu. Sigmoidna transformacija je istakla kontrast oko normalizovanog intenziteta piksela 0.3. Algoritam ekvalizacije histograma je globalno poboljšao kontrast slike, ali je dobijena slika previše svetla. *CLAHE* je otklonio problem prevelike osvetljenosti, a omogućio poboljšanje kontrasta.

Pr. 4.2. Primer *transformacije_intenziteta.py* – ilustracija efekata različitih transformacija intenziteta

---

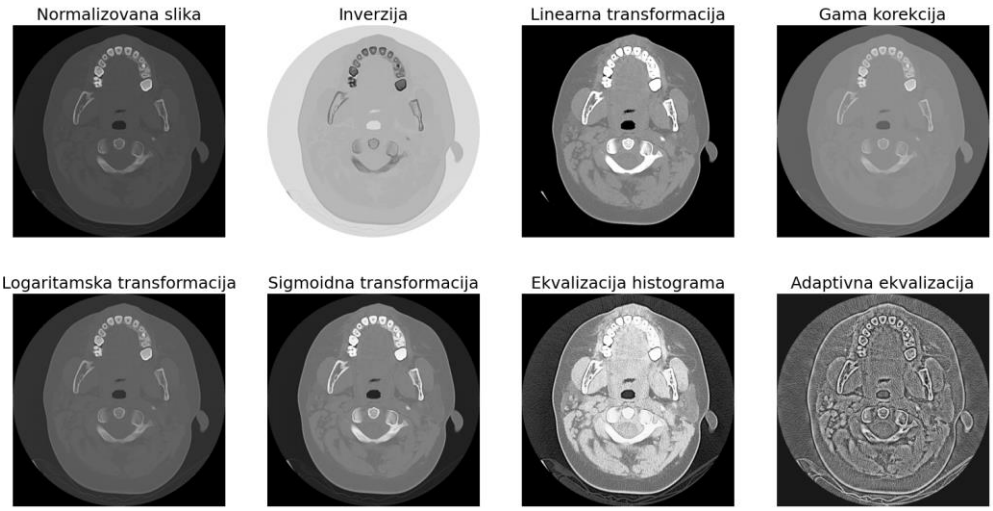
A. Programski kôd:

```
import imageio
import numpy as np
from skimage import exposure, util, img_as_ubyte
import matplotlib.pyplot as plt

plt.close('all')

# Ucitavanje DICOM slike
slika = imageio.imread('0.dcm')

# Normalizacija na opseg [0, 255]
norm_slika = exposure.rescale_intensity(
    slika, out_range=(0, 255)).astype(np.uint8)

# Linearna transformacija (reskaliranje intenziteta)
```

```
linearno_tran_slika = exposure.rescale_intensity(
    norm_slika, in_range=(50, 100), out_range=(0, 255))

# Inverzna slika
inverzna_slika = util.invert(norm_slika)

# Gama korekcija
gama_slika = exposure.adjust_gamma(norm_slika, gamma=0.5)

# Logaritamska transformacija
logaritamska_slika = exposure.adjust_log(norm_slika, gain=1)

# Sigmoidna transformacija
sigmoidna_slika = exposure.adjust_sigmoid(
    norm_slika, cutoff=0.3, gain=10)

# Ekvalizacija histograma
# Opseg [0, 1])
ekvalizovana_slika = exposure.equalize_hist(norm_slika)
# Opseg [0, 255]
ekvalizovana_slika = img_as_ubyte(ekvalizovana_slika)

# Adaptivna ekvalizacija histograma
adaptivno_ekv_slika = exposure.equalize_adapthist(
    norm_slika, clip_limit=0.1, kernel_size=(10,10))
adaptivno_ekv_slika = img_as_ubyte(adaptivno_ekv_slika)

# Uporedni prikaz rezultata transformacija intenziteta
naslovi = [
    'Normalizovana slika', 'Inverzija',
    'Linearna transformacija', 'Gama korekcija',
    'Logaritamska transformacija', 'Sigmoidna transformacija',
    'Ekvalizacija histograma', 'Adaptivna ekvalizacija'
]
```

```
slike = [
    norm_slika, inverzna_slika, linearno_tran_slika,
    gama_slika, logaritamska_slika, sigmoidna_slika,
    ekvalizovana_slika, adaptivno_ekv_slika
]

fig, axes = plt.subplots(2, 4)
axes = axes.ravel()
for ax, img, naslov in zip(axes, slike, naslovi):
    ax.imshow(img, cmap='gray')
    ax.set_title(naslov, fontsize=18)
    ax.axis('off')
```

B. Rezultat izvršavanja:



Zadatak za samostalni rad 4.1. Dopuniti Pr. 4.2 tako da se prikaže histogram slike normalizovane na opseg [0,1]. Dati i uporedni prikaz rezultata sigmoidne transformacije intenziteta za sledeći izbor parametara:

1) *cutoff*=0.15, c_{sig} = 5

2) *cutoff*=0.15, c_{sig} = 10

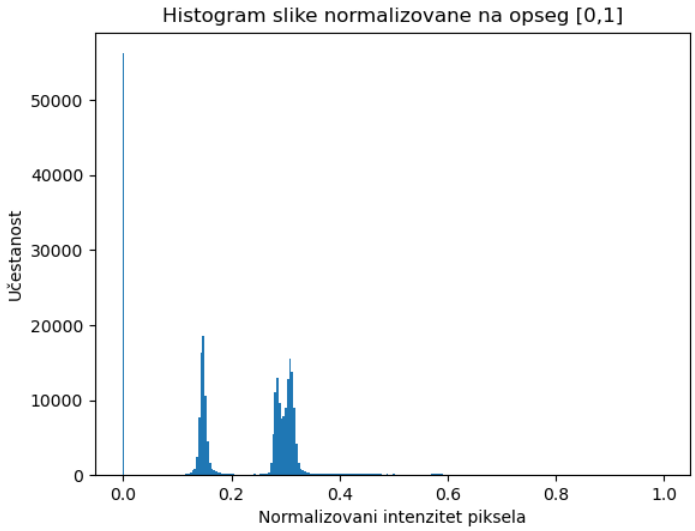
3) *cutoff*=0.3, c_{sig} = 5

4) *cutoff*=0.3, c_{sig} = 10

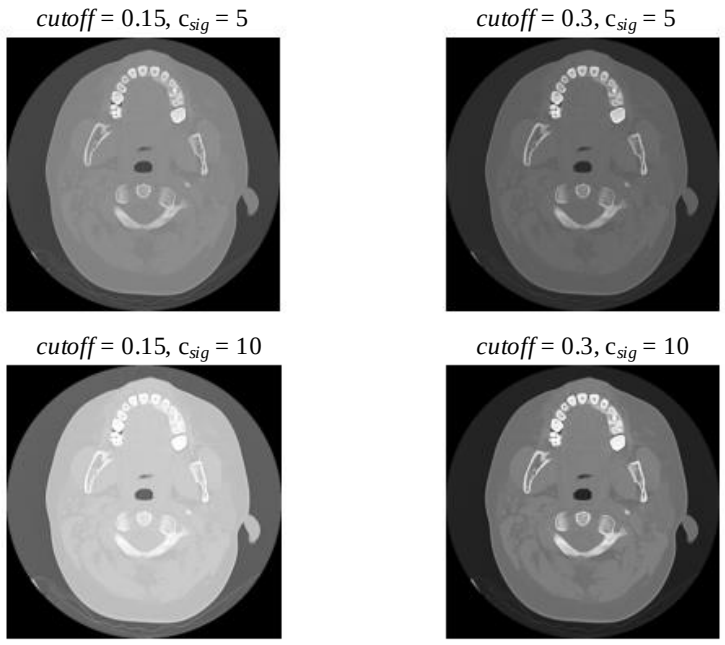
Na osnovu histograma slike normalizovane na opseg [0,1] obrazložiti zašto su predložene gore navedene vrednosti parametara *cutoff* i c_{sig}.

Rešenje

Rešenje zadatka je prikazano na Sl. 4.3 (histogram slike normalizovane na opseg [0,1]) i Sl. 4.4 (rezultati primene sigmoidne transformacije intenziteta).



Slika 4.3. Histogram slike normalizovane na opseg [0,1]

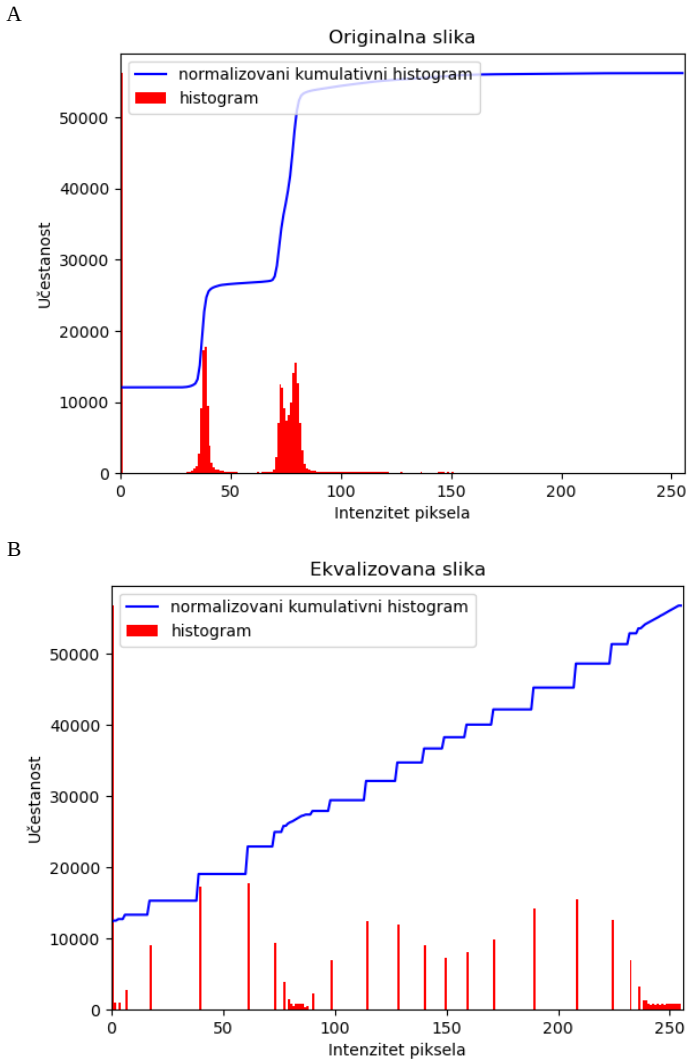


Slika 4.4. Prikaz rezultata primene sigmoidne transformacije intenziteta

Zadatak za samostalni rad 4.2. Sledeći opis koraka algoritma za ekvalizaciju histograma datog u Poglavlju 4.7.2, implementirati algoritam ekvalizacije histograma i primeniti ga na slici „0.dcm“. Prikazati normalizovani kumulativni histogram uporedo sa histogramom originalne i ekvalizovane slike (za proračun kumulativnog histograma koristiti funkciju `cumsum()`).

Rešenje

Rešenje je prikazano na Sl. 4.5. Radi prikaza histograma i kumulativnog histograma na istom grafiku, kumulativni histogram je normalizovan na maksimum histograma.



Slika 4.5. A) Histogram i normalizovani kumulativni histogram originalne slike, B) Histogram i normalizovani kumulativni histogram ekvalizovane slike

4.9. Rezime poglavlja

- Transformacija intenziteta piksela je funkcija koja transformiše vrednosti piksela iz jednog opsega u drugi.
- „Prozorovanje“ je primena transformacije intenziteta piksela na podopseg, pri čemu se preostale vrednosti intenziteta piksela predstavljaju nulom.
- Histogram slike predstavlja zavisnost učestanosti pojavljivanja piksela od vrednosti intenziteta piksela. Vrednosti intenziteta piksela na apscisi histograma su grupisane u opsege koji se zovu binovi.
- Idealan histogram ima nisku učestanost pojavljivanja za najmanje i za najveće vrednosti intenziteta piksela.
- Često korišćene transformacije intenziteta su: linearna, inverzna, gama, logaritamska, sigmoidna, ekvalizacija histograma i adaptivna ekvalizacija histograma.
- Kontrast slike je razlika između najsvetlijih i najtamnijih delova slike. Moguće ga je poboljšati lokalno ili globalno pomoću transformacija intenziteta piksela.

5. FILTRIRANJE

Filtriranje biomedicinske slike podrazumeva primenu matematičkih operacija na biomedicinske slike radi poboljšanja kvaliteta slike, uklanjanja artefakata[22] i šumova, poboljšanja kontrasta, isticanja određenih karakteristika slike (morfoloških ili teksturalnih) i pripreme slike za dalje vrste analize: segmentaciju, tj. izdvajanje regiona od interesa (*Poglavlje 6*), analizu teksture (*Poglavlje 7*), registraciju, tj. poravnavanje slika snimljenih za isti objekat, ali pod različitim uslovima spram vremenskih trenutaka i/ili spram modaliteta snimanja (*Poglavlje 8*), vizuelizaciju (*Poglavlje 9*) i klasifikaciju (*Poglavlje 10*). Prema domenu u kome se vrši filtriranje slike, osnovna podela načina filtriranja obuhvata frekvencijsko filtriranje i prostorno filtriranje.

5.1. Frekvencijsko filtriranje

Frekvencijsko filtriranje slike je tehnika digitalne obrade slike koja obuhvata:

1) transformaciju slike iz prostornog (x,y) domena u frekvencijski (p,q) domen primenom diskretne Furijeove transformacije[23] (eng. *Discrete Fourier Transform, DFT*)
2) primenu operacija filtriranja na frekvencijske komponente slike kako bi se:
    a. slika „ugladila“ ili uklonio šum (primena frekvencijskog filtra za propuštanje niskih učestanosti, eng. *low-pass filter*),
    b. izoštrile ili detektovale ivice (primena frekvencijskog filtra za propuštanje visokih učestanosti, eng. *high-pass filter*)
    c. selektivno eliminisali ili istakli specifični detalji (primena frekvencijskog filtra nepropusnika ili propusnika opsega učestanosti, eng. *band-reject filter* ili *band-pass filter*, respektivno)

[22] Artefakti na slici su nepoželjni elementi koji se pojavljuju na slici kao rezultat poremećaja u procesu snimanja slike (npr. usled neprecizno podešene instrumentacije za snimanje, usled mehaničkih pokreta tokom snimanja i sl.). Šum na slici je slučajna varijacija intenziteta piksela (npr. usled električnih i kvantnih šumova u detektorima instrumentacije za snimanje).
[23] Francuski matematičar Žan-Batist Žozef Furije je uočio da svaki signal može da se napiše u obliku beskonačne sume sinusnih i kosinusnih funkcija različitih frekvencija, amplituda i faza (rezultat objavljen 1822. godine u knjizi *La Theorie Analitique de la Chaleur*). Ovakva transformacija je nazvana Furijeova transformacija, a njen diskretan oblik je diskretna Furijeova transformacija.

3) transformaciju filtrirane slike iz frekvencijskog (*p*,*q*) domena u prostorni (*x*,*y*) domen primenom inverzne diskretne Furijeove transformacije (eng. *Inverse Discrete Fourier Transform*, *IDFT*).

Za detaljnije informisanje o *DFT*-u, preporučena literatura je [Milić et al. 2015], [Proakis et al. 2006], a ovde će biti navedene 2D *DFT* (5.1) i 2D *IDFT* (5.2) transformacione jednačine za digitalnu sliku *I(x,y)* čije su dimenzije *m* x *n* ($x \in \{0, 1, \ldots, m-1\}$, $y \in \{0, 1, \ldots, n-1\}$):

$$F(p,q) = \frac{1}{mn} \sum_{x=0}^{m-1} \sum_{y=0}^{n-1} I(x,y) e^{-i2\pi \frac{px}{m}} e^{-i2\pi \frac{qy}{n}} \tag{5.1}$$

$$I(x,y) = \sum_{p=0}^{m-1} \sum_{q=0}^{n-1} F(p,q) e^{i2\pi \frac{px}{m}} e^{i2\pi \frac{qy}{n}} \tag{5.2}$$

gde je *F(p,q)* koeficijent 2D *DTF*-a, *i* je imaginarna jedinica, a *p* i *q* su frekvencije u istim opsezima kao *x* i *y*, respektivno ($p \in \{0, 1, \ldots, m-1\}$, $q \in \{0, 1, \ldots, n-1\}$). Iz relacije (5.1), može se uočiti da je *F(0,0)* ($p=q=0$) jednako srednjoj vrednosti svih piksela na slici. Najčešće je srednja vrednost svih piksela na slici veća od vrednosti ostalih koeficijenata *F(p,q)* ($p,q \neq 0$), pa je obično piksel *(0,0)* u frekvencijskom domenu najsvetliji piksel.

Iz relacija (5.1) i (5.2) sledi da je kompleksnost proračuna *DFT*-a $(m \cdot n)^2$. Brza Furijeova transformacija (eng. *Fast Fourier Transform*, *FFT*) je efikasan algoritam koji se koristi za proračun *DFT*-a. Najčešće korišćen *FFT* algoritam je *Cooley-Tukey* algoritam[24] [Cooley and Tukey 1965] koji se za 2D *DFT* primenjuje duž svih *m* redova, a potom duž svih *n* kolona, pri čemu se kompleksnost računanja ovim algoritmom smanjuje na $m \cdot n \cdot \log_2(m \cdot n)$.

F(p,q) su kompleksni brojevi čija magnituda $|F(p,q)|$ i snaga *P(p,q)* se definišu kao:

$$|F(p,q)| = \sqrt{Re(F)^2 + Im(F)^2} \tag{5.3}$$

$$P(p,q) = |F(p,q)|^2 \tag{5.4}$$

i čija faza $\theta(p,q)$ se definiše kao:

$$\theta(p,q) = arctan\left[\frac{Im(F)}{Re(F)}\right] \tag{5.5}$$

gde su *Re(F)* i *Im(F)* realni i imaginarni deo kompleksnog koeficijenta *F(p,q)*, respektivno.

Na Pr. 5.1A je prikazan primer primene 2D *FFT* na sintetičku sliku *horizontalna_ivica_16x16.png* dimenzija 16x16 koja sadrži horizontalnu

[24] *Cooley-Turkey* algoritam pri proračunu *DFT*-a vrši podelu signala na parne i neparne indekse, a potom primenjuje osobine simetrije i periodičnosti eksponencijala, predefinisane vrednosti i rekurziju.

ivicu. Sintetička slika je učitana primenom *OpenCV* funkcije `cv2.imread()` i konvertovana u *grayscale* primenom argumenta `cv2.IMREAD_GRAYSCALE`. Funkcija `np.fft.fft2()` je primenjena na sliku radi izračunavanja *F(p,q)* kompleksnih koeficijenta ($p \in [0, 15]$, $q \in [0, 15]$) korišćenjem *FFT* algoritma. Magnituda za Furijeove koeficijente se izračunava pomoću funkcije `np.abs()`, a prema relaciji (5.3). Za poboljšanje kontrasta pri prikazu magnitude Furijeovih koeficijenata je iskorišćena logaritamska funkcija `np.log()` (pogledati Poglavlje 4.5). Konačno, za pomeranje vrednosti *F(0,0)* iz gornjeg levog ugla u centar slike u frekvencijskom domenu, iskorišćena je funkcija `np.fft.fftshift()`. Na Pr. 5.1B su prikazane redom: 1) sintetička slika u prostornom domenu, 2) slika magnituda u frekvencijskom domenu, 3) slika magnituda uz poboljšanje kontrasta logaritamskom funkcijom, 4) slika magnituda nakon pomeraja vrednosti *F(0,0)* iz gornjeg levog ugla u centar slike magnituda uz poboljšanje kontrasta logaritamskom funkcijom (može se primetiti da je dobijena slika magnituda vertikalna linija).

Pr. 5.1. Primer *2D_FFT_horizontalna_ivica.py* – primena 2D brze Furijeove transformacije

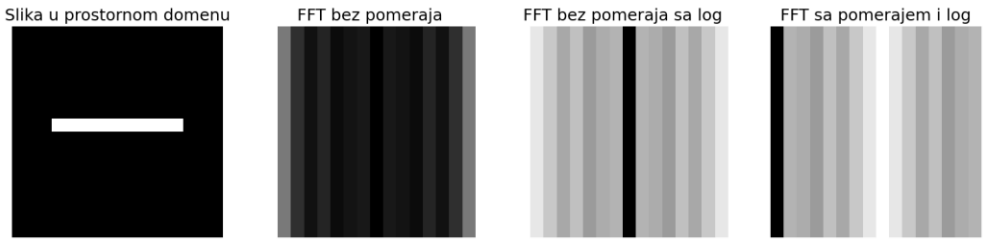
---

A. Programski kôd:

```
import numpy as np
import matplotlib.pyplot as plt
import cv2

plt.close("all")

slika=cv2.imread('horizontalna_ivica_16x16.png',
                 cv2.IMREAD_GRAYSCALE)

plt.figure(1)
# Prikaz slike
plt.subplot(1, 4, 1)
plt.title("Slika u prostornom domenu", fontsize=18)
plt.imshow(slika, cmap='gray')
plt.axis("off")
```

```
# Primena FFT-a na sliku
f = np.fft.fft2(slika)

# Prikaz rezultata FFT-a
plt.subplot(1, 4, 2)
plt.title("FFT bez pomeraja", fontsize=18)
plt.imshow(np.abs(f), cmap='gray')
plt.axis("off")

# Prikaz rezultata FFT-a sa logaritmom
plt.subplot(1, 4, 3)
plt.title("FFT bez pomeraja sa log", fontsize=18)
plt.imshow(np.log(1+np.abs(f)), cmap='gray')
plt.axis("off")

# Primena pomeraja na FFT
fshift = np.fft.fftshift(f)

# Prikaz rezultata FFT-a sa pomerajem (nula u centru)
plt.subplot(1, 4, 4)
plt.title("FFT sa pomerajem i log", fontsize=18)
plt.imshow(np.log(1+np.abs(fshift)), cmap='gray')
plt.axis("off")
```

B. Rezultat izvršavanja:

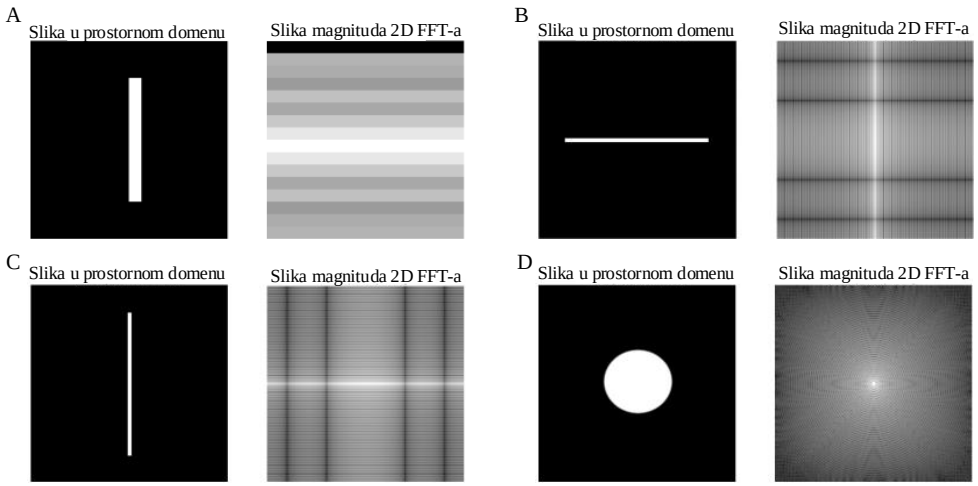


Zadatak za samostalni rad 5.1. Modifikovati Pr. 5.1. tako da se prikaže efekat primene 2D *FFT*-a na sliku koja sadrži vertikalnu ivicu (slika dobijena

rotacijom za 90^o iz Pr. 5.1). Potom kreirati slike veće rezolucije (npr. 256 x 256) koje sadrže: 1) horizontalnu ivicu, 2) vertikalnu ivicu, 3) krug u centru. Prikazati rezultate primene 2D *FFT* za kreirane slike.

Rešenje:

Na Sl. 5.1A,B,C,D su redom prikazane slike magnituda 2D *FFT*-a za vertikalnu ivicu na slici dimenzija 16 x 16, kao i slike magnituda 2D *FFT*-a primenjenog na 256 x 256 slikama horizontalne ivice, vertikalne ivice i centralnog kruga.



Slika 5.1. Slika u prostornom domenu (levo) i slika magnituda 2D *FFT*-a (desno) za: A) vertikalnu ivicu na slici dimenzija 16 x 16, B) horizontalnu ivicu na slici dimenzija 256 x 256, C) vertikalnu ivicu na slici dimenzija 256 x 256, D) centralni krug na slici dimenzija 256 x 256

Filter u frekvencijskom domenu ima definisanu svoju prenosnu funkciju *H(p,q)*, a proces filtriranja podrazumeva konvoluciju[25] u frekvencijskom domenu. Diskretna konvolucija za filtriranu sliku $I_{filtriran}$ se definiše kao:

$$I_{filtriran}(x,y) = |I * h|(x,y) = \sum_k \sum_l I(x-k, y-l) \cdot h(k,l) \qquad (5.6)$$

gde je *I* originalna slika, *h* funkcija filtra u prostornom domenu, a (*x*,*y*) pozicija piksela na slici.

Konvolucija u frekvencijskom domenu se dobija množenjem Furijeove transformacije originalne slike *F(p,q)* sa prenosnom funkcijom filtra *H(p,q)*:

$$F_{filtriran}(p,q) = F(p,q) \cdot H(p,q) \qquad (5.7)$$

[25] Konvolucija $|f * h|(t)$ je matematička operacija koja predstavlja integral preklapanja između funkcija *f(t)* i *h(t)*: $|f * h|(t) = \int_0^t f(\tau)h(t-\tau)d\tau$.

gde je $F_{filtriran}(p,q)$ *DFT* filtrirane slike u frekvencijskom domenu iz koje se filtrirana slika $I_{fitriran}(x,y)$ može dobiti primenom *IDFT* tj. relacije (5.2).

Frekvencijsko filtriranje je efikasno u uklanjanju šuma i detekciji periodičnih obrazaca (npr. ivica), primenjuje se na celu sliku (a ne parcijalno) i nije zavisno od orijentacije slike. Razlikuju su četiri grupe filtera: filtri propusnici niskih učestanosti, filtri propusnici visokih učestanosti, filtri nepropusnici opsega učestanosti i filtri propusnici opsega učestanosti.

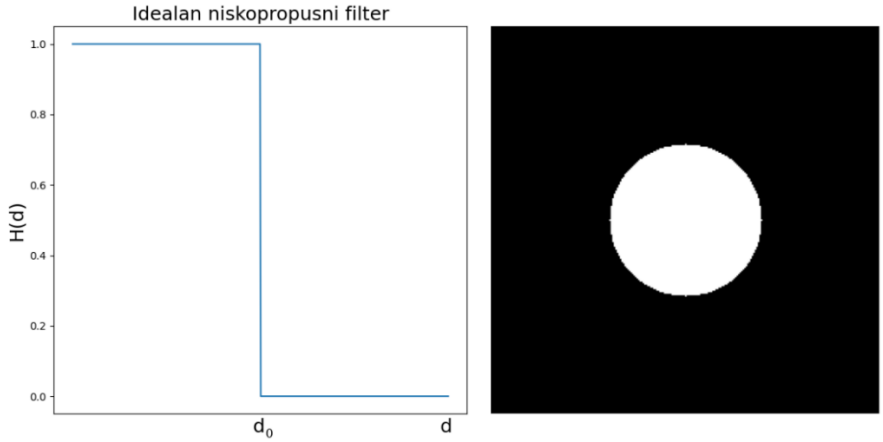
5.1.1. Filtri propusnici niskih učestanosti

Idealan 2D niskopropusni filter u frekvencijskom domenu ima sledeću prenosnu funkciju *H(p,q)*:

$$H(p,q) = \begin{cases} 1, & za\ d(p,q) \le d_0 \\ 0, & za\ d(p,q) > d_0 \end{cases} \tag{5.8}$$

$$d(p,q) = \sqrt{(p - \frac{m}{2})^2 + (q - \frac{n}{2})^2} \tag{5.9}$$

gde je $d(p,q)$ ($p \in \{0, 1, \ldots, m-1\}$, $q \in \{0, 1, \ldots, n-1\}$) Euklidsko rastojanje tačke (p,q) od centra Furijeovog prostora ($m/2$, $n/2$), a d_0 je granična frekvencija. Na Sl. 5.2 su predstavljeni prenosna funkcija *H(d)* i njen 2D prikaz u frekvencijskom *(p,q)* prostoru za $m=n=256$ i $d_0=50$. Zbog oštrog prelaza u frekvencijskom domenu, u prostornom domenu se javlja efekat „zvonjenja“ (eng. *ringing*) vidljiv kao niz svetlijih i tamnijih prstenova oko objekta ili linije.



Slika 5.2. Prenosna funkcija *H(d)* i 2D prikaz prenosne funkcije u frekvencijskom *(p,q)* prostoru za idealan niskopropusni filter

Na Pr. 5.2A je prikazana odgovarajuća implementacija modula *idealan_niskopropusni_filter* koji vrši „glađenje“ 61. snimka dinamske renografije *drsbru_001_POST.dcm*, opisane i korišćene u Poglavlju 3. Ulazni argumenti modula *idealan_niskopropusni_filter* su `snimak` i `d0`, tj. slika koja treba da se filtrira i granična frekvencija filtra, respektivno. U okviru modula se vrše sledeći koraci:

- primena FFT-a na ulaznu sliku pomoću funkcije `np.fft.fft2()` i pomeranje *F(0,0)* u centar pomoću funkcije `np.fft.fftshift()`
- kreiranje prenosne funkcije *H(p,q)* primenom relacija (5.8) i (5.9) – pomoću funkcije `np.zeros()` su najpre vrednosti svih elemenata matrice *H(p,q)* inicijalizovane na vrednost „0“ (kao neoznačeni celobrojni tip sa 16 bita), a potom je dodeljena vrednost „1“ elementima na Euklidskom rastojanju manjem ili jednakom graničnoj učestanosti (`d <= d0`). Funkcija `np.ogrid()`[26] generiše 1D vektore koji odgovaraju horizontalnoj i vertikalnoj koordinati.
- primena filtriranja u frekvencijskom domenu, tj. primena konvolucije u frekvencijskom domenu prema relaciji (5.7)
- *IDFT* primenom inverznog *FFT*-a. Najpre se centar Furijeovog prostora vraća u gornji levi ugao korišćenjem funkcije `np.fft.ifftshift()`, a potom se inverzni FFT nalazi funkcijom `np.fft.ifft2()`. Radi uklanjanja malih imaginarnih delova koji nastaju kao posledica matematičkih proračuna, iz rezultata dobijenog inverznim *FFT*-om se traži moduo pomoću `np.abs()` funkcije.

Na Pr. 5.2B je prikazan rezultat primene idealnog niskopropusnog filtra na ulaznu scintigrafsku sliku bubrega sa radijusom tj. graničnom frekvencijom `d0=30`. Može se primetiti da je filtrirana slika „uglađena“ u odnosu na originalnu sliku.

[26] Predstavlja alternativu `np.meshgrid()` funkciji koja je memorijski zahtevnija jer formira celu 2D koordinatnu mrežu.

Pr. 5.2. Primer *idealan_niskopropusni_filter.py* – implementacija i primer primene idealnog niskopropusnog filtra

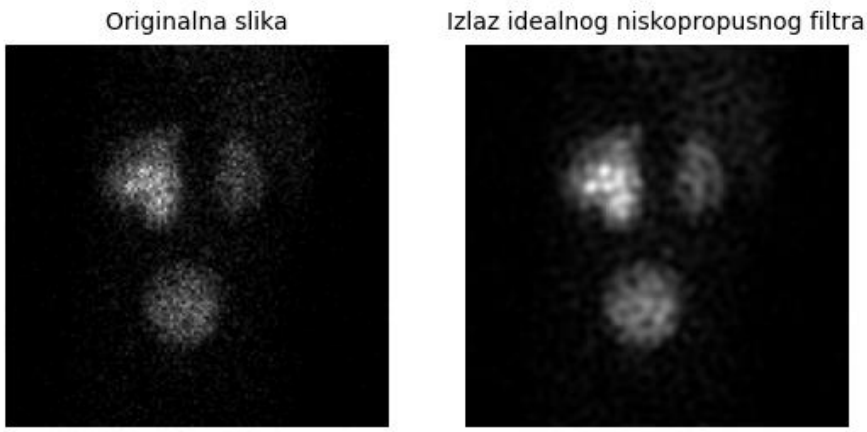
---

A. Programski kôd:

```
import numpy as np
import matplotlib.pyplot as plt
import SimpleITK as sitk

plt.close("all")

def idealan_niskopropusni_filter(slika, d0):
    # Primena FFT-a na sliku i pomeranje
    f = np.fft.fft2(slika)
    fshift = np.fft.fftshift(f)

    # Kreiranje idealnog niskopropusnog filtra
    m, n = slika.shape
    centar_p = m // 2
    centar_q = n // 2

    H = np.zeros((m, n), dtype=np.uint16)
    q, p = np.ogrid[:m, :n]
    d=np.sqrt((p - centar_p)**2 + (q - centar_q)**2)
    H[d <= d0] = 1

    # Primena frekvencijskog filtra
    fshift_filtriran = fshift * H

    # Inverzna Furijeova transformacija
    f_ishift = np.fft.ifftshift(fshift_filtriran)
    filtrirana_slika = np.fft.ifft2(f_ishift)
    filtrirana_slika = np.abs(filtrirana_slika)

    return filtrirana_slika
```

```
# Ucitavanje DICOM sekvence slika
# Izdvajanje 61. snimka
dicom_sekvenca = sitk.ReadImage("drsbru_001_POST.dcm",
                                sitk.sitkInt16)
dicom_sekvenca = sitk.GetArrayFromImage(dicom_sekvenca)
snimak=dicom_sekvenca[60,:,:]

# Primena idealnog niskopropusnog filtra
d0 = 30
filtrirana_slika = idealan_niskopropusni_filter(snimak, d0)

# Prikaz originalne i filtrirane slike
plt.subplot(1, 2, 1)
plt.title("Originalna slika", fontsize=10)
plt.imshow(snimak, cmap='gray')
plt.axis('off')

plt.subplot(1, 2, 2)
plt.title("Izlaz idealnog niskopropusnog filtra", fontsize=10)
plt.imshow(filtrirana_slika, cmap='gray')
plt.axis('off')
```

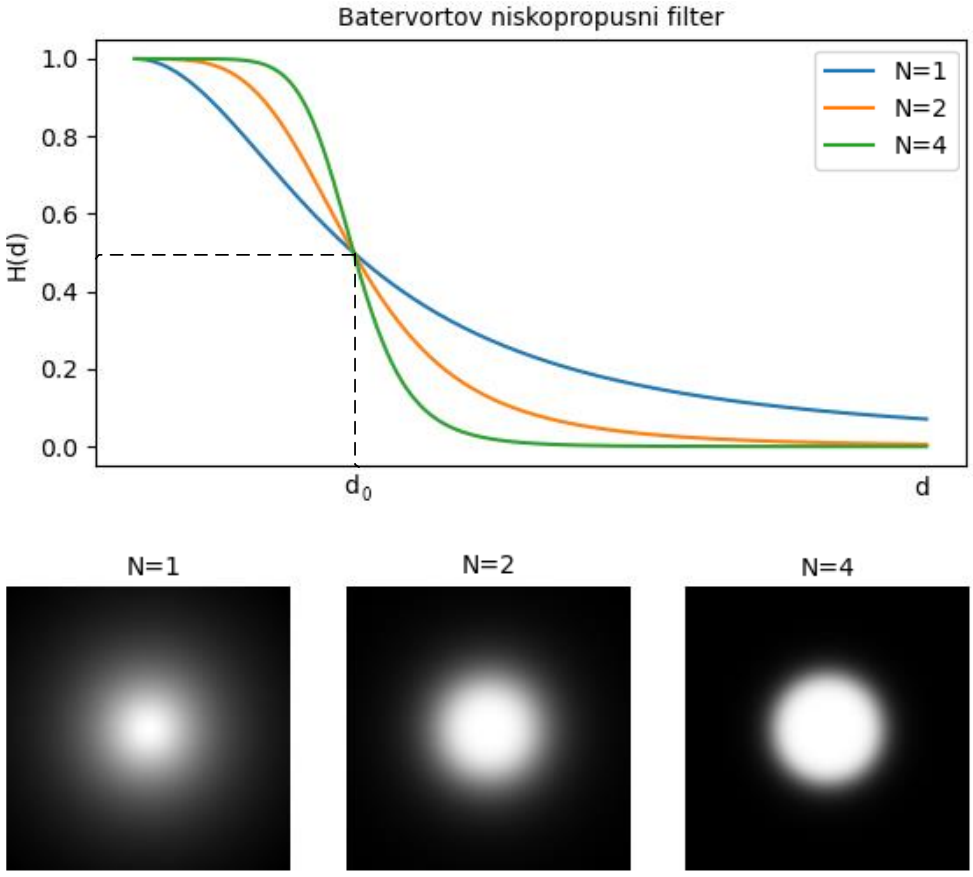
B. Rezultat izvršavanja:



Batervortov (eng. *Butterworth*) 2D niskopropusni filter u frekvencijskom domenu ima sledeću prenosnu funkciju *H(p,q)*:

$$H(p,q) = \frac{1}{1+\left(\frac{d(p,q)}{d_0}\right)^{2N}} \tag{5.10}$$

gde je *d(p,q)* ($p \in \{0, 1, \ldots, m-1\}$, $q \in \{0, 1, \ldots, n-1\}$) Euklidsko rastojanje tačke (*p,q*) od centra Furijeovog prostora (*m/2, n/2*) prema relaciji (5.9), d_0 je granična frekvencija, a *N* je red filtra. Na Sl. 5.3 su predstavljeni prenosna funkcija *H(d)* i njen 2D prikaz u frekvencijskom *(p,q)* prostoru za *N*=1,2,4, *m*=*n*=256 i d_0=50. Za $d(p,q)=d_0$ prenosna funkcija *H(d)* je 50% svoje maksimalne vrednosti. Za razliku od idealnog niskopropusnog filtra, Batervortov niskopropusni filter nema strmi prelazak između propuštenih i filtriranih frekvencija. Sa porastom reda *N* Batervortovog niskopropusnog filtra prenosna funkcija *H(d)* je strmija. Kod višeg reda Batervortovog niskopropusnog filtra kada je prelaz u frekvencijskom domenu oštar, može doći do efekta „zvonjenja“ u prostornom domenu.

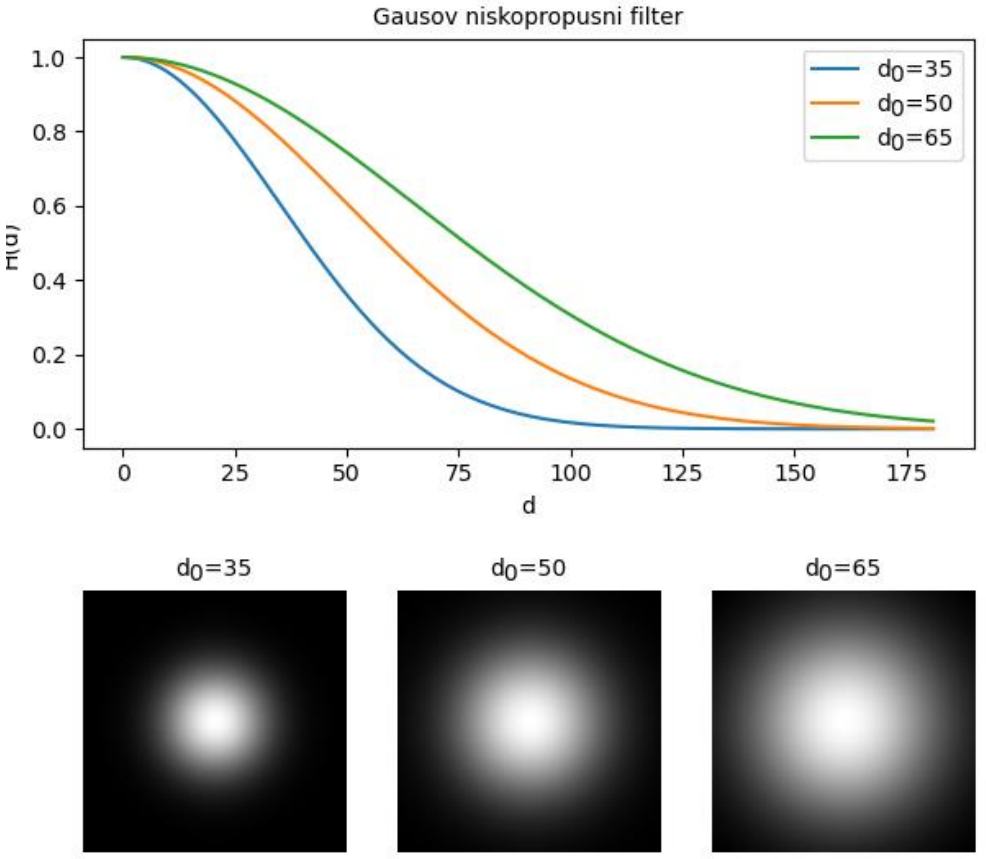


Slika 5.3. Prenosna funkcija *H(d)* i 2D prikaz prenosne funkcije u frekvencijskom *(p,q)* prostoru za Batervortov niskopropusni filter

Gausov (eng. *Gaussian*) 2D niskopropusni filter u frekvencijskom domenu ima sledeću prenosnu funkciju *H(p,q)*:

$$H(p,q) = e^{-\frac{d(p,q)^2}{2d_0^2}} \tag{5.11}$$

gde je *d(p,q)* ($p \in \{0, 1, \ldots, m-1\}$, $q \in \{0, 1, \ldots, n-1\}$) Euklidsko rastojanje tačke (*p,q*) od centra Furijeovog prostora (*m/2, n/2*) prema relaciji (5.9), a d_0 je granična frekvencija. Na Sl. 5.4 su predstavljeni prenosna funkcija *H(d)* i njen 2D prikaz u frekvencijskom *(p,q)* prostoru za d_0=35, 50, 65 i *m*=*n*=256. Za $d(p,q)=d_0$ prenosna funkcija *H(d)* je 0.607 svoje maksimalne vrednosti. Kao i Batervortov niskopropusni filter, i Gausov niskopropusni filter nema strmi prelazak između propuštenih i filtriranih frekvencija. Sa smanjenjem d_0 prenosna funkcija *H(d)* Gausovog filtra je strmija.



Slika 5.4. Prenosna funkcija *H(d)* i 2D prikaz prenosne funkcije u frekvencijskom *(p,q)* prostoru za Gausov niskopropusni filter

Zadatak za samostalni rad 5.2. Modifikovati Pr. 5.2. tako da implementacija odgovara: 1) Batervortovom niskopropusnom filtru drugog reda za d_0=30 i 2) Gausovom niskopropusnom filtru za d_0=30. Za prikaz rezultat filtriranja koristiti istu ulaznu sliku kao u Pr. 5.2.

Rešenje:

Na Sl. 5.5 su prikazane originalna ulazna slika (61. snimak dinamske scintigrafije „drsbru_001_POST.dcm*"* opisane i korišćene u Poglavlju 3), kao i prikazi rezultata filtriranja Batervortovim i Gausovim niskopropusnim filtrom.

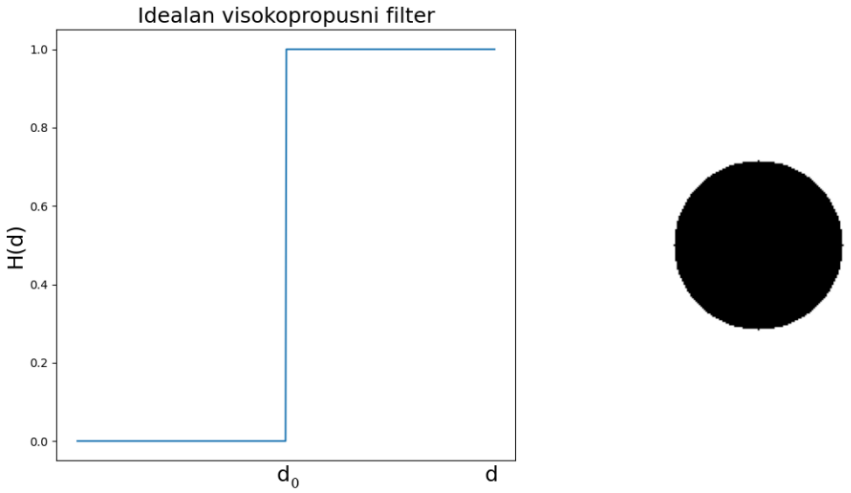
Slika 5.5. Prikaz originalne slike i izlaza Batervortovog i Gausovog niskopropusnog filtra

5.1.2. Filtri propusnici visokih učestanosti

Idealan 2D visokopropusni filter u frekvencijskom domenu ima sledeću prenosnu funkciju *H(p,q)*:

$$H(p,q) = \begin{cases} 0, & za\ d(p,q) \le d_0 \\ 1, & za\ d(p,q) > d_0 \end{cases} \tag{5.12}$$

gde je *d(p,q)* ($p \in \{0, 1, \ldots, m\text{-}1\}$, $q \in \{0, 1, \ldots, n\text{-}1\}$) Euklidsko rastojanje tačke (*p,q*) od centra Furijeovog prostora (*m/2, n/2*) prema relaciji (5.9), a d_0 je granična frekvencija. Na Sl. 5.6 su predstavljeni prenosna funkcija *H(d)* i njen 2D prikaz u frekvencijskom *(p,q)* prostoru za *m*=*n*=256 i d_0=50. I kod idealnog visokopropusnog filtra zbog oštrog prelaza u frekvencijskom domenu, u prostornom domenu se javlja efekat „zvonjenja".

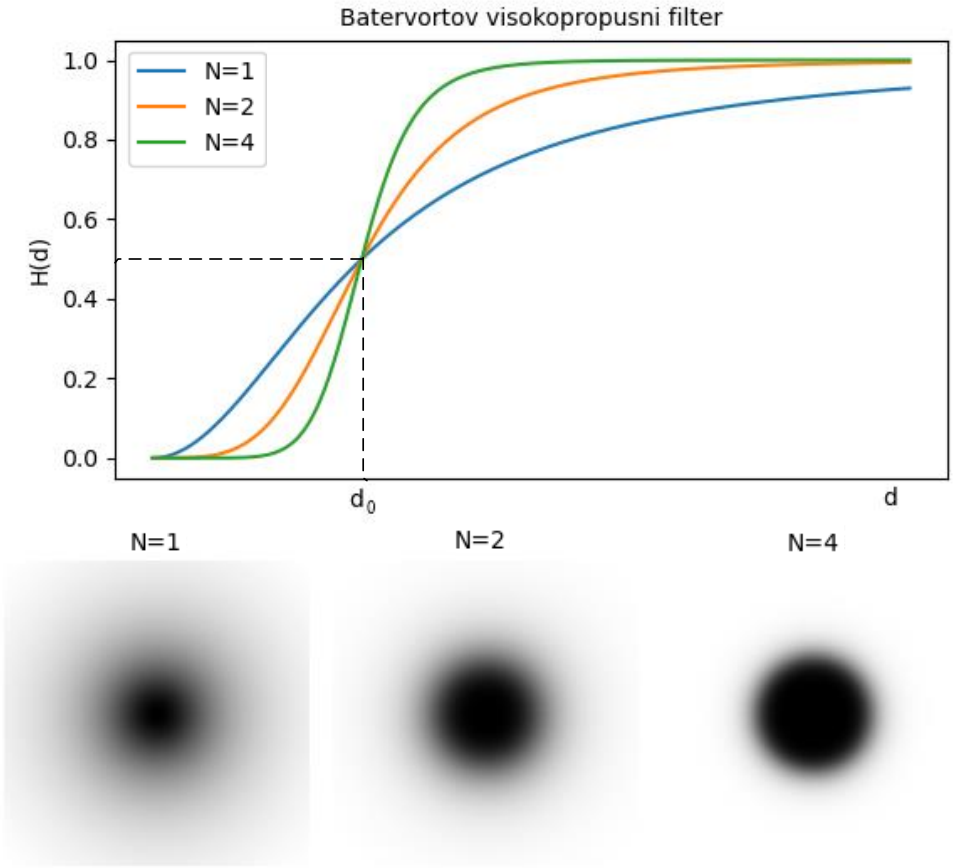


Slika 5.6. Prenosna funkcija *H(d)* i 2D prikaz prenosne funkcije u frekvencijskom *(p,q)* prostoru za idealan visokopropusni filter

Batervortov 2D visokopropusni filter u frekvencijskom domenu ima sledeću prenosnu funkciju *H(p,q)*:

$$H(p,q) = \frac{1}{1+\left(\frac{d_0}{d(p,q)}\right)^{2N}} \tag{5.13}$$

gde je *d(p,q)* ($p \in \{0, 1, \ldots, m-1\}$, $q \in \{0, 1, \ldots, n-1\}$) Euklidsko rastojanje tačke (*p,q*) od centra Furijeovog prostora (*m/2, n/2*) prema relaciji (5.9), d_0 je granična frekvencija, a *N* je red filtra. Na Sl. 5.7 su predstavljeni prenosna funkcija *H(d)* i njen 2D prikaz u frekvencijskom *(p,q)* prostoru za *N*=1,2,4, *m*=*n*=256 i d_0=50. Za $d(p,q)=d_0$ prenosna funkcija *H(d)* je 50% svoje maksimalne vrednosti. Za razliku od idealnog visokopropusnog filtra, Batervortov visokopropusni filter nema strmi prelazak između propuštenih i filtriranih frekvencija. Sa porastom reda *N* Batervortovog visokopropusnog filtra prenosna funkcija *H(d)* je strmija. Kod višeg reda Batervortovog visokopropusnog filtra može doći do efekta „zvonjenja“ u prostornom domenu.

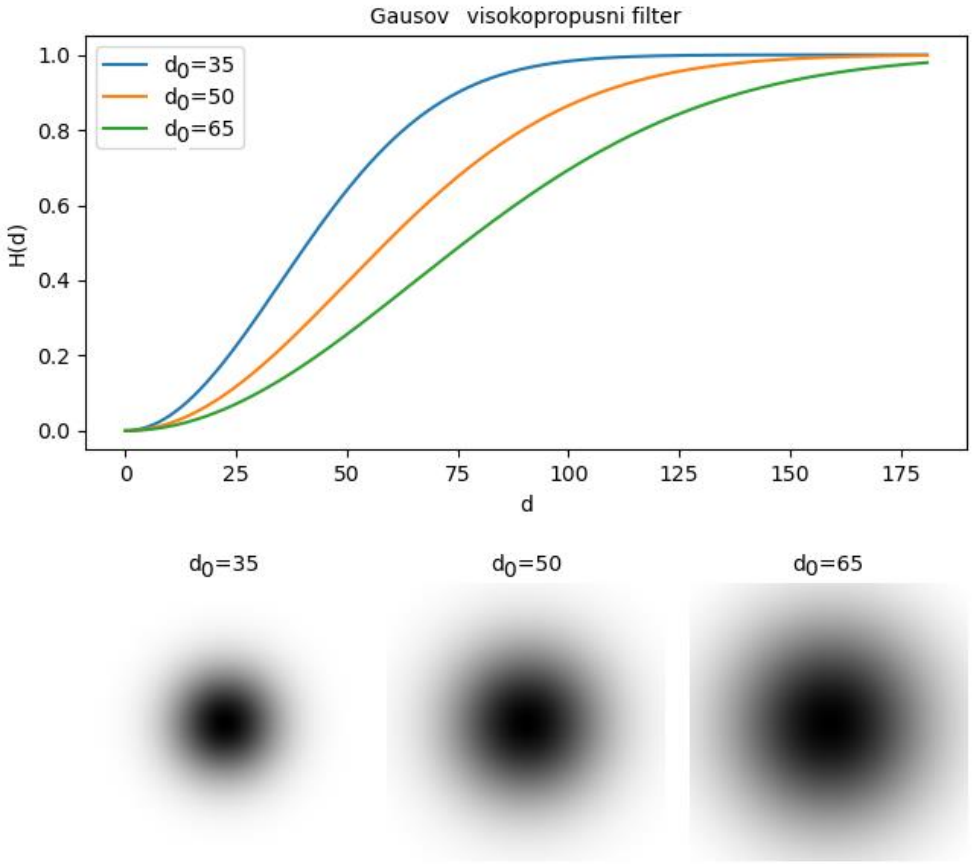


Slika 5.7. Prenosna funkcija *H(d)* i 2D prikaz prenosne funkcije u frekvencijskom *(p,q)* prostoru za Batervortov visokopropusni filter

Gausov 2D visokopropusni filter u frekvencijskom domenu ima sledeću prenosnu funkciju *H(p,q)*:

$$H(p,q) = 1 - e^{-\frac{d(p,q)^2}{2d_0^2}} \tag{5.14}$$

gde je *d(p,q)* ($p \in \{0, 1, \ldots, m\text{-}1\}$, $q \in \{0, 1, \ldots, n\text{-}1\}$) Euklidsko rastojanje tačke (*p,q*) od centra Furijeovog prostora (*m/2, n/2*) prema relaciji (5.9), a d_0 je granična frekvencija. Na Sl. 5.8 su predstavljeni prenosna funkcija *H(d)* i njen 2D prikaz u frekvencijskom *(p,q)* prostoru za d_0=35, 50, 65 i *m*=*n*=256. Za $d(p,q)=d_0$ prenosna funkcija *H(d)* je 0.393 svoje maksimalne vrednosti. Kao i Batervortov visokopropusni filter, i Gausov visokopropusni filter nema strmi prelazak između propuštenih i filtriranih frekvencija. Sa smanjenjem d_0 prenosna funkcija *H(d)* Gausovog filtra je strmija.

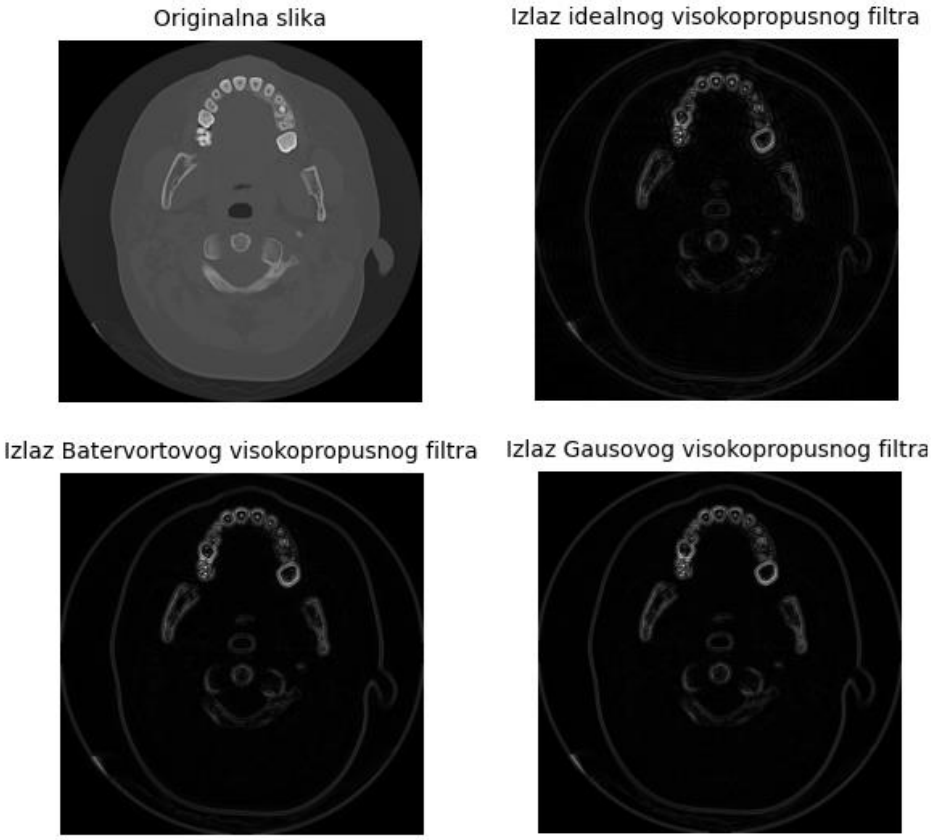


Slika 5.8. Prenosna funkcija *H(d)* i 2D prikaz prenosne funkcije u frekvencijskom *(p,q)* prostoru za Gausov visokopropusni filter

Zadatak za samostalni rad 5.3. Modifikovati Pr. 5.2. tako da implementacija odgovara: 1) idealnom visokopropusnom filtru za d_0=30, 2) Batervortovom visokopropusnom filtru drugog reda za d_0=30 i 2) Gausovom visokopropusnom filtru za d_0=30. Za ulaznu sliku za filtriranje iskoristiti „0.dcm“ datoteku korišćenu u Poglavlju 3.

Rešenje:

Na Sl. 5.5 su prikazane originalna ulazna *CT* slika glave („0.dcm*“* opisana i korišćena u Poglavlju 3), kao i prikazi rezultata filtriranja idealnim, Batervortovim i Gausovim visokopropusnim filtrima. Može se primetiti da su rezultati filtriranja pomoću sva tri tipa filtra vrlo slični, i da ističu ivice originalne slike.



Slika 5.5. Prikaz originalne slike i izlaza idealnog, Batervortovog i Gausovog visokopropusnog filtra

5.1.3. Filtri nepropusnici i propusnici opsega učestanosti

Idealan 2D filter nepropusnik opsega učestanosti u frekvencijskom domenu ima sledeću prenosnu funkciju *H(p,q)*:

$$H(p,q) = \begin{cases} 0, & za\ d_0 - \frac{W}{2} \le d(p,q) \le d_0 + \frac{W}{2} \\ 1, & za\ d(p,q) < d_0 - \frac{W}{2} \quad ili \quad d(p,q) > d_0 + \frac{W}{2} \end{cases} \tag{5.15}$$

gde je *d(p,q)* ($p \in \{0, 1, \ldots, m-1\}$, $q \in \{0, 1, \ldots, n-1\}$) Euklidsko rastojanje tačke (*p*,*q*) od centra Furijeovog prostora (*m/2*, *n/2*) prema relaciji (5.9), d_0 je granična frekvencija, a *W* frekvencijski opseg.

Batervortov filter nepropusnik opsega učestanosti u frekvencijskom domenu ima sledeću prenosnu funkciju *H(p,q)*:

$$H(p,q) = \frac{1}{1+\left[\frac{d(p,q)W}{d(p,q)^2-d_0^2}\right]^{2n}} \quad (5.16)$$

Gausov filter nepropusnik opsega učestanosti u frekvencijskom domenu ima sledeću prenosnu funkciju *H(p,q)*:

$$H(p,q) = 1 - e^{-\left[\frac{d(p,q)^2-d_0^2}{d(p,q)W}\right]^2} \quad (5.17)$$

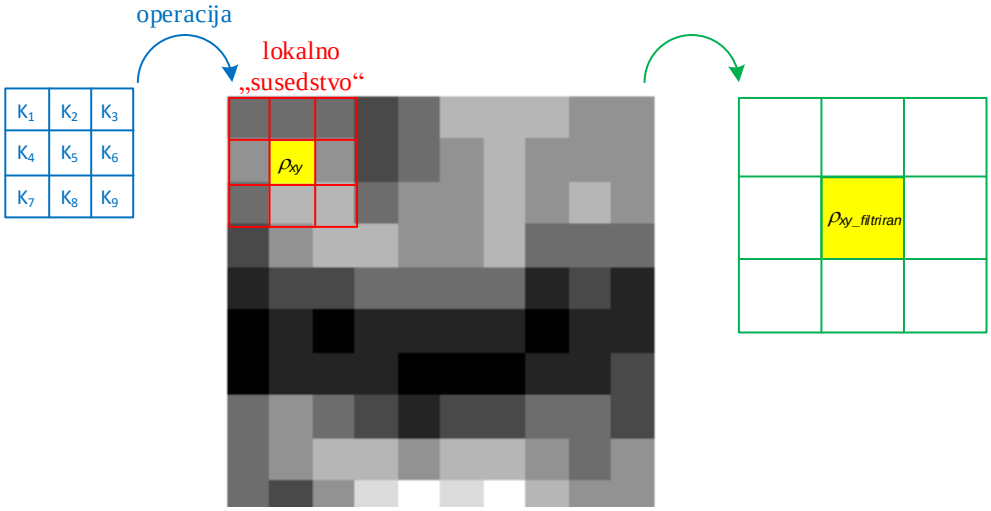
Filtri propusnici opsega učestanosti imaju sledeću prenosnu funkciju:

$$H_{propusnik}(p,q) = 1 - H_{nepropusnik}(p,q) \quad (5.18)$$

gde je $H_{nepropusnik}(p,q)$ zadat nekom od relacija (5.15), (5.16), (5.17).

5.2. Prostorno filtriranje

Prostorno filtriranje slike je tehnika digitalne obrade slike koja primenjuje prostorne filtre direktno na piksele slike u prostornom domenu. Većina prostornih filtera definiše i koristi lokalno „susedstvo“ piksela i tzv. kernel koji predstavlja malu matricu (npr. dimenzija 3x3) čiji centar se poklapa sa pikselom ρ_{xy} za koji se u procesu prostornog filtriranja formira „nova“ filtrirana vrednost $\rho_{xy\_filtriran}$. Na Sl. 5.6 je ilustrovan proces prostornog filtriranja baziranog na lokalnom „susedstvu“ piksela tj. korišćenju kernela. Između lokalnog „susedstva“ piksela ρ_{xy} i kernela se primenjuje matematička operacija koja kao rezultat daje filtriranu vrednost $\rho_{xy\_filtriran}$. Prikazani proces se ponavlja za sve piksele slike.



Slika 5.6. Princip prostornog filtriranja na bazi primene kernela

Operacija koja se u procesu filtriranja na bazi kernela primenjuje između kernela i lokalnog „susedstva“ piksela može biti linearna ili nelinearna. U zavisnosti od toga da li je operacija koja se primenjuje linearna ili nelinearna[27], razlikuju se linearni i nelinearni prostorni filtri. U narednim podpoglavljima će biti objašnjeni principi nekih od linearnih prostornih filtera (*Mean, Gaussian, Sobel, Prewitt, Laplacian*) i nelinearnih filtera (*Min, Max, Median*) kao i *Canny* filtra koji kombinuje linearne i nelinearne operacije. Prostorni filtri se prema oblasti primene mogu podeliti na:

a. filtre za „glađenje“ (eng. *smoothing*), „zamućenje“ (eng. *blurring*) ili uklanjanje šuma (propuštanje niskih učestanosti, eng. *low-pass filter*) kao što su: *Mean, Median, Min, Max, Gaussian*

b. filtre za izoštravanje ili detektovanje ivica (propuštanje visokih učestanosti, eng. *high-pass filter*) kao što su: *Sobel, Prewitt, Laplacian, Canny.*

Prostorni filtri bazirani na kernelu su jednostavni za implementaciju, efikasni za obradu lokalnih promena na slici, fleksibilni za dizajniranje i prilagođavanje specifičnim zahtevima (veličina i elementi matrice kernela se mogu prilagoditi nameni) i pogodni su za implementaciju obrade slika u realnom vremenu.

Osim prostornih filtera koji primenjuju kernele, postoje i prostorni filtri koji ih ne primenjuju već operišu na celoj slici ili koriste globalne karakteristike slike, ali je njihova primena ređa u biomedicinskom slikanju. Takvi su na primer bilateralni filter, *Non-Local Means* filter i Vinerov filter o čijim principima funkcionisanja se može pogledati detaljnije u dodatnoj literaturi [Paris et al 2009], [Buades et al 2005], [Gonzalez and Woods 2008], respektivno.

5.2.1. Linearni prostorni filtri

U slučaju linearnog prostornog filtriranja, rezultat primene filtra na bazi kernela za svaki piksel slike $I(x,y)$, ($x \in \{0, 1, \ldots, m-1\}$, $y \in \{0, 1, \ldots, n-1\}$) je piksel intenziteta $\rho_{xy\_filtriran}$:

$$\begin{aligned}\rho_{xy\_filtriran} = {} & K_1 \cdot \rho_{x-1\,y-1} + K_2 \cdot \rho_{x-1\,y} + K_3 \cdot \rho_{x-1\,y+1} \\ & + K_4 \cdot \rho_{x\,y-1} + K_5 \cdot \rho_{xy} + K_6 \cdot \rho_{x\,y+1} \\ & + K_7 \cdot \rho_{x+1\,y-1} + K_8 \cdot \rho_{x+1\,y} + K_9 \cdot \rho_{x+1\,y+1}\end{aligned} \qquad (5.19)$$

gde je ρ_{xy} originalni intenzitet piksela na (x,y) poziciji slike I, a K_i ($i \in \{0, 1, \ldots, 9\}$ su koeficijenti kernela tj. filtra ili „maske“ dimenzije 3x3, Sl. 5.6.

[27] Funkcija f je linearna ako za nju važi: $f(x+y)=f(x)+f(y)$, u suprotnom je nelinearna.

Relacija (5.19) predstavlja konvoluciju u prostornom domenu (pogledati i relaciju (5.6)) i primenjuje se na svaki piksel slike *I*.

Kada je centar kernela pozicioniran na ivicama slike *I*, neophodno je dopuniti sliku (eng. *padding*) kako bi mogla biti primenjena relacija (5.19). Dopunjeni pikseli se uzimaju u obzir samo tokom procesa konvolucije „lokalnog susedstva" piksela i kernela u prostornom domenu. Dopunjavanje se može realizovati na različite načine:

- dopunjavanje nulama
- dopunjavanjem konstantom različitom od nule
- dopunjavanje vrednostima najbližeg suseda (tj. vrednostima iz ivičnog reda ili kolone)
- refleksijom vrednosti piksela u odnosu na ivicu slike i sl.

***Mean* filter**. Za *Mean* filter (filter za usrednjavanje) su svi koeficijenti K_i ($i \in \{0, 1, \ldots, N\}$ kernela jednaki 1. *Mean* kernela dimenzija 3x3 je:

$$K_{Mean} = \begin{vmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{vmatrix}. \tag{5.20}$$

Nakon konvolucije slike *I* u prostornom domenu sa kernelom K_{Mean} se intenzitet svakog piksela podeli sa *N* gde je *N* broj piksela kernela da bi se izbeglo prekomerno uvećavanje ili smanjenje intenziteta piksela filtrirane slike (za kernel dimenzija 3x3 *N* je 9).

Alternativno, koristi se *težinski Mean* filter kod koga su svi koeficijenti K_i ($i \in \{0, 1, \ldots, N\}$ kernela jednaki $1/N$ gde je *N* broj piksela kernela. *Težinski Mean* kernel dimenzija 3x3 je:

$$K_{težinski\_Mean} = \frac{1}{9}\begin{vmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{vmatrix}. \tag{5.20}$$

Osnovna namena *Mean* filtra je smanjivanje niskofrekventnog šuma. Na Pr. 5.3A je prikazana implementacija *Mean* filtra koji vrši uklanjanje šuma na 61. snimku dinamske renografije *drsbru_001_POST.dcm* opisane i korišćene u Poglavlju 3. Nakon učitavanja 61. snimka, definisan je kernel *Mean* filtra, `mean_filter` prema relaciji (5.20), a potom je linearna prostorna konvolucija sprovedena korišćenjem funkcije `ndi.convolve()` za originalnu sliku `snimak` i kernel `mean_filter`. Na Pr. 5.3B je prikazan rezultat izvršavanja programskog kôda.

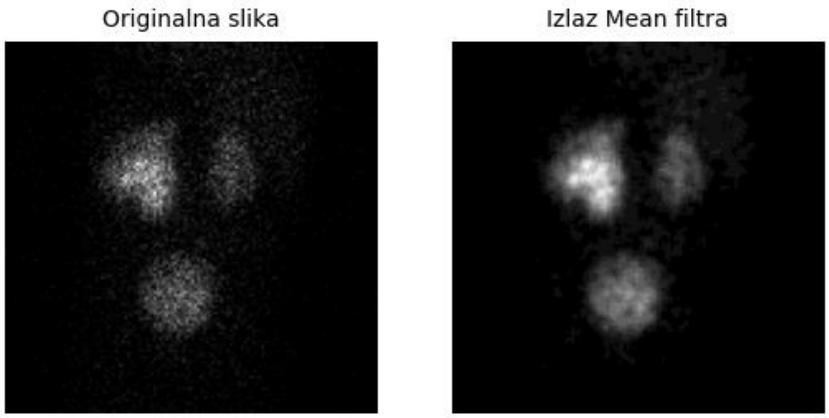
Pr. 5.3. Primer *mean_filter.py* – implementacija i primer primene *Mean* filtra

---

A. Programski kôd:

```
import SimpleITK as sitk
import numpy as np
import matplotlib.pyplot as plt
import scipy.ndimage as ndi

plt.close("all")

# Ucitavanje DICOM sekvence slika
# Izdvajanje 61. snimka
dicom_sekvenca = sitk.ReadImage("drsbru_001_POST.dcm",
                                sitk.sitkInt16)
dicom_sekvenca = sitk.GetArrayFromImage(dicom_sekvenca)
snimak=dicom_sekvenca[60,:,:]

mean_filter = 1/9 * np.array([[1,1,1], [1,1,1], [1,1,1]])
filtrirana_slika = ndi.convolve(snimak, mean_filter)

plt.subplot(1, 2, 1)
plt.title("Originalna slika", fontsize=10)
plt.imshow(snimak, cmap='gray')
plt.axis('off')

plt.subplot(1, 2, 2)
plt.title("Izlaz Mean filtra", fontsize=10)
plt.imshow(filtrirana_slika, cmap='gray')
plt.axis('off')
```

B. Rezultat izvršavanja:



***Gaussian* filter**. *Gaussian* filter sadrži koeficijente generisane Gausovom funkcijom:

$$G(x,y) = \frac{1}{2\pi\sigma^2} e^{-\frac{k_x{}^2+k_y{}^2}{2\sigma^2}}. \tag{5.21}$$

gde su (k_x,k_y) koordinate piksela kernela u odnosu na centar kernela, a σ standardna devijacija. Standardna devijacija definiše širinu i stepen zamućenja – za veće σ je Gausovo zvono šire tj. vrednosti koeficijenata sporije opadaju od centra ka ivicama kernela, a zamućenje je jače tj. detalji slike postaju manje izraženi. Vrednosti koeficijenata *Gaussian* kernela se obično zaokružuju na celobrojne vrednosti radi jednostavnije implementacije i primene, i normalizuju se tako da suma koeficijenata kernela bude 1 radi izbegavanja prekomernog uvećanja ili smanjenja intenziteta piksela filtrirane slike.

Gaussian kernel 3x3 (za σ=1) se standardno definiše na sledeći način:

$$K_{Gaussian\ 3x3} = \frac{1}{16}\begin{vmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{vmatrix}. \tag{5.22}$$

Gaussian kernel 5x5 (za σ=1.4) se standardno definiše na sledeći način:

$$K_{Gaussian\ 5x5} = \frac{1}{256}\begin{vmatrix} 1 & 4 & 6 & 4 & 1 \\ 4 & 16 & 24 & 16 & 4 \\ 6 & 24 & 36 & 24 & 6 \\ 4 & 16 & 24 & 16 & 4 \\ 1 & 4 & 6 & 4 & 1 \end{vmatrix}. \tag{5.23}$$

Osnovna namena *Gaussian* filtra je zamućivanje slike čime se slika priprema za dalju obradu.

***Sobel* filter**. *Sobel* filter spada u filtre prvog izvoda tj. aproksimira prvi izvod intenziteta piksela slike I u horizontalnom i vertikalnom pravcu. Prvi izvod meri promene u intenzitetu slike duž određenog pravca što omogućava detekciju ivica. Prvi izvod duž x-ose tj. promena intenziteta duž horizontalnog pravca se aproksimira diskretnim diferenciranjem:

$$\frac{\partial I}{\partial x} \approx I(x+1,y) - I(x-1,y) \tag{5.24}$$

a horizontalni *Sobel* kernel za detekciju vertikalnih ivica se definiše na sledeći način:

$$K_{Sobel\,h} = \begin{vmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{vmatrix} \tag{5.25}$$

pri čemu centralni red ima veće koeficijente radi povećanja uticaja piksela bližih posmatranom centralnom pikselu (x,y).

Slično tome, prvi izvod duž y-ose tj. promena intenziteta duž vertikalnog pravca se aproksimira diskretnim diferenciranjem:

$$\frac{\partial I}{\partial y} \approx I(x,y+1) - I(x,y-1) \tag{5.26}$$

a vertikalni *Sobel* kernel za detekciju horizontalnih ivica se definiše na sledeći način:

$$K_{Sobel\,v} = \begin{vmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{vmatrix} \tag{5.27}$$

pri čemu centralna kolona ima veće koeficijente radi povećanja uticaja piksela bližih posmatranom centralnom pikselu (x,y).

Konačno, magnituda gradijenta u *Sobel* filtru se računa primenom Euklidske norme na horizontalnu ($G_{Sobel\,h}$) i vertikalnu komponentu gradijenta ($G_{Sobel\,v}$) koje su dobijene konvolucijom slike sa $K_{Sobel\,h}$ i $K_{Sobel\,v}$, respektivno:

$$G_{Sobel} = \sqrt{{G_{Sobel\,h}}^2 + {G_{Sobel\,v}}^2} \approx |G_{Sobel\,h}| + |G_{Sobel\,v}| \tag{5.28}$$

Ugao gradijenta u *Sobel* filtru se računa kao:

$$\theta_{Sobel} = arctan\left(\frac{G_{Sobel\,v}}{G_{Sobel\,h}}\right). \tag{5.29}$$

***Prewitt* filter**. *Prewitt* filter je takođe filter prvog izvoda koji služi za detekciju ivica na slici, ali on za razliku od *Sobel* filtra nema koeficijente različite težine, tj. ne uzima u obzir udaljenost u odnosu na centralni piksel. Zahvaljujući tome,

njegova implementacija je jednostavnija od *Sobel*-ove. Horizontalni *Prewitt* kernel za detekciju vertikalnih ivica se definiše na sledeći način:

$$K_{Prewitt\ h} = \begin{vmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{vmatrix} \tag{5.30}$$

a vertikalni *Prewitt* kernel za detekciju horizontalnih ivica se definiše kao:

$$K_{Prewitt\ v} = \begin{vmatrix} -1 & -1 & -1 \\ 0 & 0 & 0 \\ 1 & 1 & 1 \end{vmatrix}. \tag{5.31}$$

***Laplacian* filter**. *Laplacian* filter spada u filtre izvoda drugog reda tj. proračunava izvod prvog izvoda i stoga je kompjuterski zahtevniji u odnosu na filtre prvog izvoda. Drugi izvod detektuje promene u promenama intenziteta slike duž određenog pravca što omogućava preciznu detekciju ivica, ali je osetljiviji na šum u odnosu na prvi izvod (zato se često primenjuje nakon zamućivanja slike npr. *Gaussian* filtrom). *Laplacian* operator nad slikom I se definiše kao zbir drugog izvoda po horizontalnom i vertikalnom pravcu:

$$\nabla^2 I = \frac{\partial^2 I}{\partial x^2} + \frac{\partial^2 I}{\partial y^2}$$

$$\nabla^2 I \approx I(x+1,y) + I(x-1,y) + I(x,y+1) + I(x,y-1) - 4I(x,y) \tag{5.32}$$

a odgovarajući *Laplacian* kernel je:

$$K_{Laplacian} = \begin{vmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{vmatrix}. \tag{5.33}$$

Za razliku od *Sobel*-a i *Prewitt*-a gde je neophodno računati magnitudu gradijenta kombinovanjem horizontalnog i vertikalnog filtra, *Laplacian* filter detektuje promene u svim pravcima istovremeno i to u jednom koraku tj. samo primenom kernela datim u relaciji (5.33). *Laplacian* filter se često primenjuje na sliku koja je prethodno filtrirana *Gaussian* filtrom kako bi se izbegla preterana segmentacija ivica.

5.2.2. Nelinearni prostorni filtri

Jedan od najčešće korišćenih nelinearnih filtera je *Median* filter. Ovaj filter za svaku podsliku određenu kernelom računa medijanu[28] i tu vrednost

[28] Na podslici određenoj kernelom se najpre sortiraju elementi u rastućem poretku, a potom se određuje vrednost piksela koja sortiranu listu elemenata deli na dva jednaka dela.

dodeljuje centralnom pikselu (*x*,*y*). Osnovna namena *Median* filtra je smanjivanje šuma (posebno impulsnog šuma, eng. *salt and pepper*[29]).

Na Pr. 5.4A je prikazan programski kôd koji primenjuje *Median* i *Gaussian* filter na 61. snimku dinamske renografije *drsbru_001_POST.dcm* opisane i korišćene u Poglavlju 3. Nakon učitavanja 61. snimka, primenjene su funkcije *ndimage* biblioteke `median_filter()` i `gaussian_filter()` sa veličinom kernela 3x3 i σ=1 za *Gaussian* filter. Na Pr. 5.4B je prikazan rezultat izvršavanja programskog kôda.

Pr. 5.4. Primer *median_gaussian_filter.py* – primer primene *Median* i *Gaussian* filtra

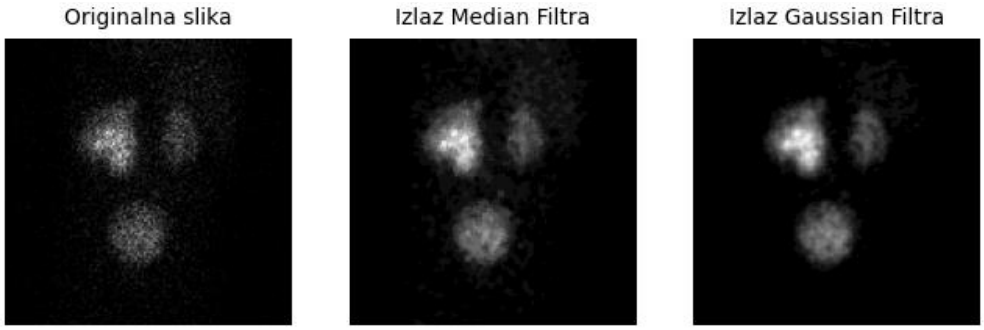
A. Programski kôd:

```
import SimpleITK as sitk
import matplotlib.pyplot as plt
from scipy.ndimage import (median_filter, gaussian_filter)

plt.close("all")

# Ucitavanje DICOM sekvence slika
# Izdvajanje 61. snimka
dicom_sekvenca = sitk.ReadImage("drsbru_001_POST.dcm",
                                sitk.sitkInt16)
dicom_sekvenca = sitk.GetArrayFromImage(dicom_sekvenca)
snimak=dicom_sekvenca[60,:,:]

median_filtrirano = median_filter(snimak, size=3)
gaussian_filtrirano = gaussian_filter(snimak, sigma=1)

plt.subplot(1, 3, 1)
plt.title("Originalna slika", fontsize=10)
plt.imshow(snimak, cmap='gray')
```

[29] Šum u vidu svetlih (so, eng. *salt*) i tamnih (biber, eng. *pepper*) tačaka na slici. Svetle tačke su obično beli pikseli (pikseli sa intenzitetom maksimalne vrednosti), a tamne tačke su obično crni pikseli (pikseli sa intenzitetom minimalne vrednosti).

```
plt.axis('off')

plt.subplot(1, 3, 2)
plt.title("Izlaz Median Filtra", fontsize=10)
plt.imshow(median_filtrirano, cmap='gray')
plt.axis('off')

plt.subplot(1, 3, 3)
plt.title("Izlaz Gaussian Filtra", fontsize=10)
plt.imshow(gaussian_filtrirano, cmap='gray')
plt.axis('off')
```

B. Rezultat izvršavanja:



Max filter za svaku podsliku određenu kernelom računa maksimalnu vrednost piksela i tu vrednost dodeljuje centralnom pikselu podslike. Osnovna namena *Max* filtra je isticanje svetlih piksela slike i uklanjanje tamnih mrlja.

Min filter za svaku podsliku određenu kernelom računa minimalnu vrednost piksela i tu vrednost dodeljuje centralnom pikselu podslike. Osnovna namena *Min* filtra je isticanje tamnih piksela slike i uklanjanje svetlih mrlja.

Na Pr. 5.5A je prikazan programski kôd koji primenjuje *Min* i *Max* filter na snimku „Cancer (1835).jpg“ iz baze podataka *Brain Tumor Dataset*[30].

[30] Baza CT slika pacijenata sa tumorom mozga i zdravih pacijenata dostupna na linku https://www.kaggle.com/datasets/preetviradiya/brian-tumor-dataset (poslednji pregled Decembar 2024) i podeljena pod GPL 2 licencom (https://www.gnu.org/licenses/old-licenses/gpl-2.0.en.html).

Nakon učitavanja snimka i konverzije u *grayscale* sliku, primenjene su funkcije `minimum_filter()` i `maximum_filter()` iz *ndimage* biblioteke sa veličinom kernela 3x3. Na Pr. 5.5B je prikazan rezultat izvršavanja programskog kôda. Može se primetiti da je *Min* filter istakao tamne detalje uključujući i crni okvir, a da je *Max* filter istakao svetle i potisnuo tamne detalje kao što je crni okvir.

Pr. 5.5. Primer *min_max_filter.py* – implementacija i primer primene *Min* i *Max* filtra

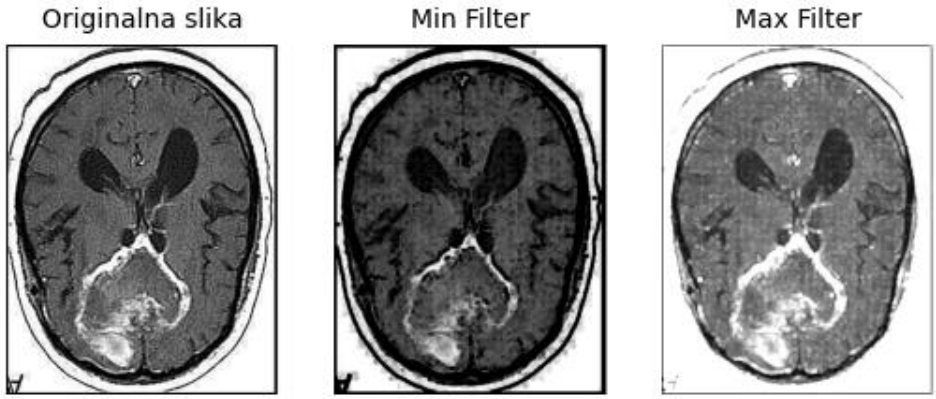
---

A. Programski kôd:

```
import cv2
from scipy.ndimage import (minimum_filter, maximum_filter)
import matplotlib.pyplot as plt

# Čitanje podataka iz datoteke
slika_RGB=cv2.imread('Cancer (1835).jpg')
# Konverzija iz RGB sistema u grayscale
slika_grayscale=cv2.cvtColor(slika_RGB, cv2.COLOR_BGR2GRAY)
# Min i Max filter
min_filtrirano = minimum_filter(slika_grayscale, size=3)
max_filtrirano = maximum_filter(slika_grayscale, size=3)

plt.subplot(1, 3, 1)
plt.title("Originalna slika", fontsize=10)
plt.imshow(slika_grayscale, cmap='gray')
plt.axis('off')

plt.subplot(1, 3, 2)
plt.title("Min Filter", fontsize=10)
plt.imshow(min_filtrirano, cmap='gray')
plt.axis('off')

plt.subplot(1, 3, 3)
plt.title("Max Filter", fontsize=10)
```

```
plt.imshow(max_filtrirano, cmap='gray')
plt.axis('off')
```

B. Rezultat izvršavanja:



Jedan od najčešće korišćenih filtera za detekciju ivica na slici je *Canny* filter čija implementacija obuhvata i linearne i nelinearne operacije:

- filtriranje primenom *Gaussian* filtra
- određivanje magnitude gradijenta i ugla gradijenta na osnovu formula (5.28) i (5.29) respektivno
- zadržavanje samo lokalnih maksimuma u pravcu gradijenta kako bi se izdvojile „tanke“ ivice (eng. *non-maximum suppression*): svaki piksel duž pravca gradijenta se poredi sa svojim susedima i ako je intenzitet posmatranog piksela manji od njegovih suseda njegova vrednost se postavlja na nulu, tj. eliminiše se iz „ivice“
- primena dva praga za jačinu gradijenta (gornji prag T_G i donji prag T_D):
    - pikseli sa intenzitetom većim od T_G se proglašavaju delom „sigurne“ ivice
    - pikseli sa intenzitetom manjim od T_D se odbacuju

- pikseli sa intenzitetom između T_D i T_G se uključuju kao deo ivice samo ako su povezani[31] sa „sigurnom“ ivicom.

Na Pr. 5.6A je prikazan programski kôd koji primenjuje *Laplacian*, *Canny*, *Sobel* i *Prewitt* filtre na snimku limfoblasta[32]„Im037_0.jpg“ iz javno dostupne baze slika ALL-IDB2[33] [Piuri et al. 2004], [Scotti 2005], [Scotti 2006], [Labati et al. 2011].

Nakon učitavanja snimka i konverzije u *grayscale* sliku, primenjene su funkcije *ndimage* biblioteke `laplace()`, `sobel()` i `prewitt()` radi primene *Laplacian, Sobel* i *Prewitt* filtera respektivno i funkcija *OpenCV* biblioteke `cv2.Canny()` radi primene *Canny* filtra. Predefinisana veličina kernela za sve pomenute filtre je 3x3. Ilustracije radi, za *Canny* filter su određeni izlazi za tri različita izbora vrednosti donjeg (`treshold1`) i gornjeg praga (`treshold2`).

Dodatno, pored direktne primene na *grayscale* sliku, *Laplacian* je primenjen i na sliku dobijenu primenom *Gaussian* filtra (σ=1) pomoću funkcije *ndimage* biblioteke `gaussian_filter()` kako bi se izbegao efekat preterane segmentacije ivica.

Za *Sobel* i *Prewitt* filtre su određeni najpre horizontalan filter (argument *axis*=0) i vertikalan filter (argument *axis*=1), a potom i gradijent magnitude pri čemu su vrednosti gradijenta magnitude konvertovane u celobrojne vrednosti opsega 0-255 radi prikaza (konverzija je izvršena pomoću funkcija `exposure.rescale_intensity()` i `astype(np.uint8)`).

Konačno, izlazi svih primenjenih filtera su smešteni u listu `filter_outputs` zajedno sa svojim odgovarajućim naslovima kako bi se na optimalan način realizovalo njihovo prikazivanje pomoću *for* petlje u narednom koraku.

Na Pr. 5.5B je prikazan rezultat izvršavanja programskog kôda. Vizuelnom inspekcijom se može uočiti da su izlazi *Canny* filtra (za donji prag

[31] Povezanost između piksela može biti *4-connected* (piksel je povezan sa drugim pikselom ako su oni susedi horizontalno ili vertikalno) ili *8-connected* (piksel je povezan sa drugim pikselom ako su oni susedi horizontalno, vertikalno ili dijagonalno). Za *Canny* filter se češće uzima u obzir *8-connected* povezanost.

[32] Limfoblasti su nezrele ćelije imunološkog sistema. U normalnim uslovima, limfoblasti su privremena faza u razvoju limfocita, ali pri akutnoj limfoblastnoj leukemiji (ALL) dolazi do nagomilavanja limfoblasta što ometa funkcionisanje imunološkog sistema.

[33] ALL-IDB2 - *Acute Lymphoblastic Leukemia Image Database for Image Processing*, Fabio Scotti, Departman za informacione tehnologije Univerziteta u Milanu, https://scotti.di.unimi.it/all/ (poslednji pregled Decembar 2024).

20 i gornji prag 90) i *Laplacian* filter primenjen na sliku filtriranu *Gaussian* filtrom dali superiornije rezultate u detekciji ivica u odnosu na ostale filtre.

Pr. 5.6. Primer *detekcija_ivica.py* – primer primene *Laplacian, Canny, Sobel* i *Prewitt* filtra

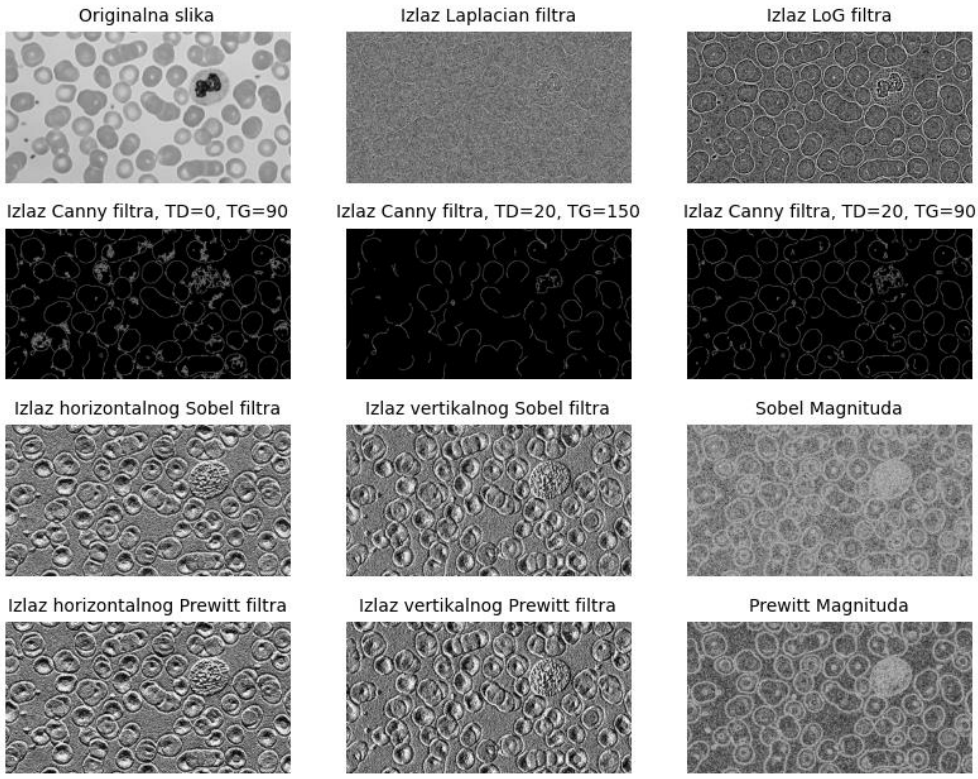
---

A. Programski kôd:

```
import numpy as np
import matplotlib.pyplot as plt
from skimage import exposure
from scipy.ndimage import (sobel, prewitt,
                           laplace, gaussian_filter)
import cv2

plt.close("all")

# Čitanje podataka iz datoteke
slika_RGB=cv2.imread('Im037_0.jpg')
# Konverzija iz RGB sistema u grayscale
slika_grayscale=cv2.cvtColor(slika_RGB, cv2.COLOR_BGR2GRAY)

laplacian_filt = laplace(slika_grayscale)
gaussian_filt = gaussian_filter(
    slika_grayscale, sigma=1)
LoG_filt = laplace(gaussian_filt)
canny_filt_p1 = cv2.Canny(
    slika_grayscale, threshold1=0, threshold2=90)
canny_filt_p2 = cv2.Canny(
    slika_grayscale, threshold1=20, threshold2=150)
canny_filt_p3 = cv2.Canny(
    slika_grayscale, threshold1=20, threshold2=90)
sobel_filt_h = sobel(slika_grayscale, axis=0)
sobel_filt_v = sobel(slika_grayscale, axis=1)
prewitt_filt_h = prewitt(slika_grayscale, axis=0)
prewitt_filt_v = prewitt(slika_grayscale, axis=1)
```

```
# Izračunavanje gradijenta magnitude za Sobel i Prewitt filter
sobel_magnituda = np.sqrt(sobel_filt_h**2 +
                          sobel_filt_v**2)
sobel_magnituda = exposure.rescale_intensity(
    sobel_magnituda, out_range=(0, 255)).astype(np.uint8)
prewitt_magnituda = np.sqrt(prewitt_filt_h**2 +
                            prewitt_filt_v**2)
prewitt_magnituda = exposure.rescale_intensity(
    prewitt_magnituda, out_range=(0, 255)).astype(np.uint8)

filter_outputs = [
    ("Originalna slika", slika_grayscale),
    ("Izlaz Laplacian filtra", laplacian_filt),
    ("Izlaz LoG filtra", LoG_filt),
    ("Izlaz Canny filtra, TD=0, TG=90", canny_filt_p1),
    ("Izlaz Canny filtra, TD=20, TG=150", canny_filt_p2),
    ("Izlaz Canny filtra, TD=20, TG=90", canny_filt_p3),
    ("Izlaz horizontalnog Sobel filtra", sobel_filt_h),
    ("Izlaz vertikalnog Sobel filtra", sobel_filt_v),
    ("Sobel Magnituda", sobel_magnituda),
    ("Izlaz horizontalnog Prewitt filtra", prewitt_filt_h),
    ("Izlaz vertikalnog Prewitt filtra", prewitt_filt_v),
    ("Prewitt Magnituda", prewitt_magnituda),
]

for i, (title, image) in enumerate(filter_outputs, start=1):
    plt.subplot(4, 3, i)
    plt.title(title, fontsize=10)
    plt.imshow(image, cmap='gray')
    plt.axis('off')
```

B. Rezultat izvršavanja:



5.3. Procena kvaliteta slike

Jedan od osnovnih ciljeva filtriranja je poboljšanje kvaliteta slike uklanjanjem šuma. Standardna kvantitativna mera za evaluaciju efikasnosti filtriranja je odnos signal/šum (eng. *Signal-to-Noise Ratio, SNR*). *SNR* se obično definiše kao odnos snage signala i snage šuma:

- u frekvencijskom domenu:

$$SNR = \frac{\sum_{u=0}^{m-1}\sum_{v=0}^{n-1}|F(u,v)|^2}{\sum_{u=0}^{m-1}\sum_{v=0}^{n-1}|N(u,v)|^2} \tag{5.34}$$

gde su $|F(u,v)|^2$ i $|N(u,v)|^2$ spektri snage slike i šuma na slici, respektivno, pri čemu je dimenzija slike m x n.

- u prostornom domenu:

$$SNR = \frac{\mu^2}{\sigma_N^2} \tag{5.35}$$

gde je μ srednja vrednost intenziteta piksela slike bez šuma, a σ standardna devijacija intenziteta šuma.

U praktičnoj primeni, šum obično nije dostupan kao posebna slika, pa se mogu koristiti sledeće formule za *SNR*:

- u frekvencijskom domenu:

$$SNR = \frac{\sum_{u=0}^{m-1}\sum_{v=0}^{n-1}|F(u,v)|^2}{m \cdot n \cdot \sigma_N^2} \tag{5.36}$$

gde je $|F(u,v)|^2$ spektar snage slike sa šumom, a σ_N je standardna devijacija šuma procenjena iz pozadinskog dela slike, pri čemu je dimenzija slike *m* x *n* (jednačina važi pod pretpostavkom da je šum homogen)

- u prostornom domenu:

$$SNR = \frac{\sum_{x=0}^{m-1}\sum_{y=0}^{n-1}\rho(x,y)^2}{\sum_{x=0}^{m-1}\sum_{y=0}^{n-1}(\rho(x,y)-\rho_{filt}(x,y))^2} \tag{5.37}$$

gde su $\rho(x,y)$ i $\rho_{filt}(x,y)$ intenziteti piksela originalne i filtrirane slike na poziciji (x,y)

$$ili \quad SNR = \frac{\mu^2}{\sigma^2} \tag{5.38}$$

gde je μ srednja vrednost intenziteta piksela slike sa šumom, a σ standardna devijacija intenziteta slike sa šumom.

SNR se često izražava u decibelima (dB) kao:

$$SNR\ [dB] = 10 \cdot log\ SNR \tag{5.39}$$

5.4. Rezime poglavlja

- Cilj filtriranja biomedicinskih slika je poboljšanje kvaliteta slike, uklanjanje artefakata i šumova, poboljšanje kontrasta, isticanje određenih karakteristika slike (morfoloških ili teksturalnih) i priprema slike za dalje vrste analize.
- Osnovna podela načina filtriranja obuhvata frekvencijsko filtriranje i prostorno filtriranje.
- Frekvencijsko filtriranje obuhvata sledeće korake: 1) diskretna Furijeova transformacija slike, 2) konvolucija transformisane slike i prenosne funkcije filtra u frekvencijskom domenu, 3) inverzna diskretna Furijeova transformacija rezultata konvolucije iz prethodnog koraka.
- Prostorno filtriranje slike je tehnika digitalne obrade slike koja primenjuje prostorne filtre direktno na piksele slike u prostornom domenu. Prostorno filtriranje rezultuje novim vrednostima piksela slike

usled primene linearne ili nelinearne operacije između male matrice (kernela) i lokalnog „susedstva“ piksela.

- Prostorni filtri se prema linearnosti operacije dele na linearne (*Mean, Gaussian, Sobel, Prewitt, Laplacian*) i nelinearne (*Min, Max, Median*). *Canny* filter kombinuje linearne i nelinearne operacije.
- Prema nameni, prostorni filtri se dele na filtre za propuštanje niskih učestanosti radi „glađenja“, „zamućenja“ ili uklanjanja šuma (*Mean, Median, Min, Max, Gaussian*) i filtre za propuštanje visokih učestanosti radi izoštravanja ili detektovanja ivica (*Sobel, Prewitt, Laplacian, Canny*).
- Standardna kvantitativna mera za evaluaciju efikasnosti filtriranja je odnos signal/šum (odnos snage signala i snage šuma).

6. SEGMENTACIJA

Segmentacija obuhvata metode analize digitalne slike koje omogućavaju izdvajanje značajnih delova ili regiona od interesa slike (eng. *region of interest, ROI*) na osnovu kriterijuma sličnosti susednih piksela, a sa ciljem njihove detaljnije analize ili ekstrakcije informacija. Kriterijumi sličnosti mogu biti intenzitet, ivice, tekstura itd. Segmentacija predstavlja važan korak za kompjuterski podržanu dijagnostiku (eng. *Computer-Aided Diagnosis, CAD*) jer omogućava: lokalizaciju i identifikaciju abnormalnosti, merenje površine ili zapremine *ROI* što omogućava praćenje progresa bolesti, personalizaciju tretmana i praćenje efekta terapije. Segmentacija se vrlo često primenjuje na poboljšanim (vidi Poglavlje 4) i filtriranim (vidi Poglavlje 5) slikama. Automatska segmentacija bilo primenom konvencionalnih metoda koje će biti opisane u ovom poglavlju, bilo primenama metoda mašinskog učenja (vidi Poglavlje 10), omogućava standardizaciju tj. eliminiše subjektivne razlike u interpretaciji slika između različitih observera. Dodatno, automatska segmentacija značajno može da ubrza proces donošenja dijagnostičkih odluka.

Automatske metode segmentacije se mogu podeliti na:

- Metode zasnovane na statistici histograma
- Metode zasnovane na detekciji ivica
- Metode zasnovane na regionima
- Metode zasnovane na aktivnim konturama
- Metode zasnovane na klasterovanju
- Metode zasnovane na dubokom učenju
- Metode zasnovane na grafovima
- Hibridne metode.

U ovom poglavlju će biti objašnjeni primeri metoda zasnovanih na statistici histograma (Potpoglavlje 6.6), metoda zasnovanih na regionima (Potpoglavlje 6.7), metoda zasnovanih na aktivnim konturama (Potpoglavlje 6.8) i primer hibridnog pristupa (Potpoglavlje 6.9). Osnovni koncepti duboke radiomike su predstavljeni u Poglavlju 10.

Metode zasnovane na detekciji ivica su opisane detaljno u Poglavlju 5. Ivica slike predstavlja granicu između regiona različitih intenziteta piksela. Ivica je oštra za nagli prelaz i zamućena za postepeni prelaz između regiona

različitih intenziteta piksela. Povezivanjem ivica u okviru jednog *ROI* se formira kontura tj. linija koja predstavlja spoljašnju granicu *ROI*. Dobijene konture se koriste za segmentaciju i analizu oblika objekata na slici.

Za metode zasnovane na klasterovanju, dubokom učenju i grafovima se može konsultovati dodatna literatura [Dhawan 2011], [Rangaraj 2005], [Bankman 2009], [Gonzalez and Woods 2008].

6.1. Manuelna segmentacija

Manuelna segmentacija predstavlja proces ručnog označavanja *ROI*. Manuelna segmentacija je vrlo precizna jer se oslanja na ekspertizu korisnika, ali može biti izuzetno vremenski zahtevna. Rezultati ovakve segmentacije mogu varirati između korisnika u zavisnosti od njihovog iskustva u interpretiranju biomedicinskih slika (intervarijabilnost). Intervarijabilnost rezultata segmentacije se može smanjiti standardizacijom protokola segmentacije, adekvatnom obukom korisnika i primenom asistivnih alata za poluautomatsku ili automatsku segmentaciju (uz kontinuiranu superviziju i korekciju od strane korisnika). Za složenije primere segmentacije, neophodno je postići konsenzus eksperata radi definisanja zlatnog standarda rezultata segmentacije. Takođe, rezultati segmentacije se mogu razlikovati i ako su urađeni od strane iste osobe u različitim vremenskim trenucima (intravarijabilnost). Intravarijabilnost rezultata segmentacije se može smanjiti obezbeđivanjem ujednačenih uslova rada (osvetljenje, veličina i rezolucija monitora) i koncentracije korisnika, kao i automatizacijom repetitivnih zadataka.

Često korišćeni besplatni alati za manuelnu segmentaciju su *ImageJ* [34], *ITK-SNAP*[35], *3D Slicer*[36]. Ovi alati pružaju i podršku za poluatomatsku ili automatsku segmentaciju.

Na Pr. 6.1. je prikazan programski kôd koji omogućava interaktivno zaokruživanje *ROI* na usrednjenom[37] renogramu „usrednjen_renogram.jpg“ (sa povećanom rezolucijom sa 128x128 na 1024x1024, [Mladenović 2024]) dobijenog na osnovu *drsbru_001_POST.dcm* snimka opisanog i korišćenog u Poglavlju 3.

[34] *ImageJ website*: https://imagej.net (poslednji pregled Decembar 2024)
[35] *ITK-SNAP website*: http://www.itksnap.org/pmwiki/pmwiki.php (poslednji pregled Decembar 2024)
[36] *3D Slicer website*: https://www.slicer.org (poslednji pregled Decembar 2024)
[37] Usrednjen renogram predstavlja zbirnu sliku svih frejmova (ili određenog broja frejmova) dinamske scintigrafije bubrega podeljenu sa brojem frejmova

Pr. 6.1. Primer *manuelna_segmentacija.py* – interaktivno zaokruživanje i čuvanje regiona od interesa

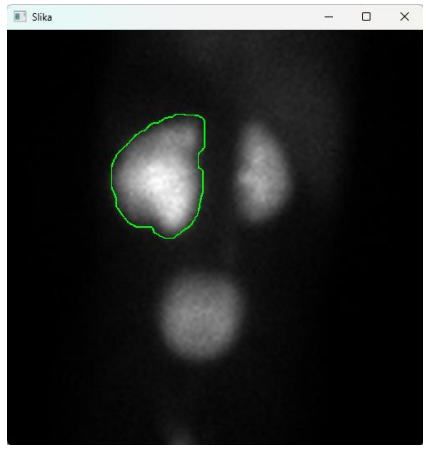
---

A. Programski kôd:

```
import matplotlib.pyplot as plt
import numpy as np
import cv2

plt.close('all')

def crtaj(event, x, y, flags, param):
    global crtanje, poslednja_x, poslednja_y, roi

    if event == cv2.EVENT_LBUTTONDOWN:
        crtanje = True
        poslednja_x, poslednja_y = x, y
        roi[x, y] = 1

    elif event == cv2.EVENT_MOUSEMOVE:
        if crtanje:
            cv2.line(slika, (poslednja_x, poslednja_y),
                     (x, y), (0, 255, 0), 2)
            roi[x, y] = 1
            poslednja_x, poslednja_y = x, y

    elif event == cv2.EVENT_LBUTTONUP:
        crtanje = False
        cv2.line(slika, (poslednja_x, poslednja_y),
                 (x, y), (0, 255, 0), 2)
        roi[x, y] = 1

# Ucitavanje slike
slika = cv2.imread('usrednjen_renogram.jpg')
visina, sirina, kanal = slika.shape
```

```
# Inicijalizacija globalnih promenljivih
crtanje = False
poslednja_x, poslednja_y = 0, 0
roi = np.zeros((visina, sirina))

# Prikaz interaktivnog prozora za crtanje
cv2.namedWindow('Slika', cv2.WINDOW_NORMAL)
cv2.resizeWindow('Slika', 500, 500)
cv2.setMouseCallback('Slika', crtaj)

# Crtanje ROI
while True:
    cv2.imshow('Slika', slika)
    key = cv2.waitKey(1) & 0xFF
    if key == 27:  # Pritisnuti ESC za izlaz
        break

# Zatvaranje prozora
cv2.destroyAllWindows()

# Čuvanje ROI matrice u fajl
np.save('roi.npy', roi)
```

B. Rezultat izvršavanja:



Nakon učitavanja slike „usrednjen_renogram.jpg“, inicijalizuju se vrednosti globalnih promenljivih modula `crtaj`: fleg `crtanje` se postavlja inicijalno na *False* vrednost koja odgovara môdu u kome nema crtanja, poslednja pozicija miša (`poslednja_x`, `poslednja_y`) se postavlja na (0,0), a matrica `roi` koja je namenjena pamćenju *ROI* koordinata na vrednosti 0. Prikaz interaktivnog prozora sa imenom „Slika“ u meniju i učitanom slikom, podešavanje veličine prozora na 500 x 500 piksela i pozivanje modula `crtaj` je omogućeno redom pomoću OpenCV funkcija: `cv2.namedWindow()`, `cv2.resizeWindow()` i `cv2.setMouseCallback()`, respektivno. Potom se unutar *while* petlje na svakih 1 ms (vrednost argumenta funkcije `cv2.waitKey` je 1) proverava da li je pritisnut taster ESC (tj. da li se u nižih 8 bita vrednosti koju vraća funkcija `cv2.waitKey(1)` nalazi vrednost 0x1B, što je broj 27 u decimalnom sistemu). Operator AND (cv2.waitKey(1) & 0xFF) omogućava da se posmatra samo najnižih 8 bita u kojima je sadržana informacija o pritisnutom tasteru. Kada je pritisnut ESC taster, proces crtanja se smatra završenim, interaktivni prozor se zatvara pomoću funkcije `cv2.destroyAllWindows()`, a matrica `roi` koja na (*x*,*y*) pozicijama nacrtane *ROI* linije ima vrednosti „1“ (na pozicijama van *ROI* linije vrednost „0“) se čuva u binarnoj datoteci *roi.npy* pomoću funkcije `np.save()`. Modul `crtaj` obuhvata sledeće korake:

- označavanje promenljivih `crtanje`, `poslednja_x`, `poslednja_y`, `roi` za globalne promenljive čije vrednosti će se menjati unutar modula
- kada je levi taster miša pritisnut na dole (događaj: `event == cv2.EVENT_LBUTTONDOWN`), fleg `crtanje` menja vrednost u *True*, poslednja pozicija miša (`poslednja_x`, `poslednja_y`) se postavlja na trenutnu (*x*,*y*) koordinatu, a odgovarajući element matrice `roi[x,y]` menja vrednost na 1.
- sve dok je levi taster miša pritisnut i zaokružuje *ROI* (događaj: `event == cv2.EVENT_MOUSEMOVE`), na interaktivnom prozoru se u zelenoj boji crta linija koja spaja prethodnu tačku sa trenutnom primenom funkcije `cv2.line()`. Argument (0,255,0) se odnosi na (R,G,B) sistem pa je zato ispisivanje *ROI* u prozoru zelenom bojom (R=0, G=255, B=0). Debljina linije je 2 piksela (argument 2). Odgovarajući element matrice `roi[x,y]` menja vrednost na 1.
- kada se levi taster miša otpusti (događaj: `event == cv2.EVENT_LBUTTONUP`) fleg `crtanje` menja vrednost u *False*, prethodna i trenutna pozicija se spajaju linijom zelene boje debljine 2

piksela, a odgovarajući element matrice `roi[x,y]` menja vrednost na 1.

Zadatak za samostalni rad 6.1. Kreirati program koji može da pročita sačuvanu `roi` matricu u datoteci *roi.npy* iz Pr. 6.1 i prikaže je na usrednjenom renogramu.

Zadatak za samostalni rad 6.2. Dopuniti Pr. 6.1 tako da se nakon iscrtavanja granica *ROI* formira tzv. maska za *ROI*: svi pikseli unutar *ROI* maske dobijaju vrednost „1“, a svi pikseli van *ROI* maske imaju vrednost „0“. Primer rezultata *ROI* maske (za *ROI* iz rezultata izvršavanja u Pr. 6.1) je prikazan na Sl. 6.1. Pikseli unutar *ROI* maske koji imaju vrednost „1“ su bele boje, a pikseli van *ROI* maske koji imaju vrednost „0“ su crne boje.

Slika 6.1. Maska za *ROI*

6.2. Maskiranje

Maskiranje je proces selekcije piksela slike na kojima će se vršiti dalja obrada ili ekstrakcija informacija. Maskiranje se vrši pomoću maske koja je obično binarna ili logička: pikseli maske koji imaju vrednost „1“ili *True* odgovaraju pozicijama piksela slike na kojima će se vršiti dalja analiza, a pikseli maske koji imaju vrednost „0“ ili *False* odgovaraju pozicijama piksela na kojima se neće vršiti dalja analiza. Maske mogu da budu i sive, i tada intenziteti piksela maske imaju vrednosti u opsegu [0,1] i predstavljaju nivo transparentnosti tj. težine za dalju analizu.

Na Pr. 6.2A je ilustrovan proces maskiranja na jednostavnom primeru matrice dimenzija 3x3.

Pr. 6.2. Primer *maskiranje.py* – kreiranje maskirane slike

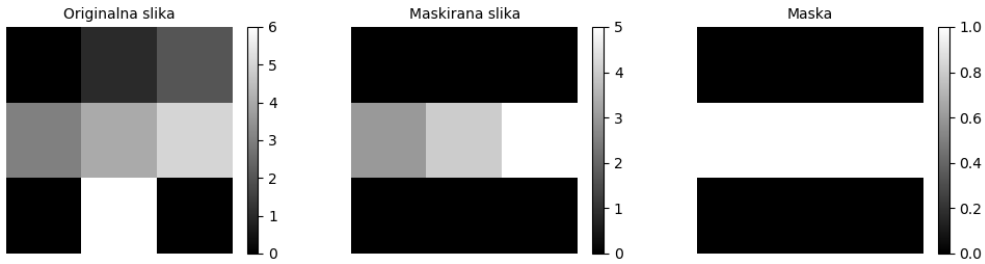
A. Programski kôd:

```
import numpy as np
import matplotlib.pyplot as plt

plt.close('all')

originalna_slika=np.array([[0,1,2],[3,4,5],[0,6,0]])
print('originalna slika=')
print(originalna_slika)

maska=(originalna_slika>2)&(originalna_slika<6)
print('maska=')
print(maska)

maskirana_slika=np.where(maska,originalna_slika,0)
print('maskirana slika=')
print(maskirana_slika)

plt.subplot(1, 3, 1)
plt.title("Originalna slika", fontsize=10)
plt.imshow(originalna_slika, cmap='gray')
plt.colorbar()
plt.axis('off')

plt.subplot(1, 3, 2)
plt.title("Maskirana slika", fontsize=10)
plt.imshow(maskirana_slika, cmap='gray')
plt.colorbar()
plt.axis('off')

plt.subplot(1, 3, 3)
plt.title("Maska", fontsize=10)
plt.imshow(maska, cmap='gray')
```

```
plt.colorbar()
plt.axis('off')
```

B. Rezultat izvršavanja:

```
originalna slika=
[[0 1 2]
 [3 4 5]
 [0 6 0]]
maska=
[[False False False]
 [ True  True  True]
 [False False False]]
maskirana slika=
[[0 0 0]
 [3 4 5]
 [0 0 0]]
```

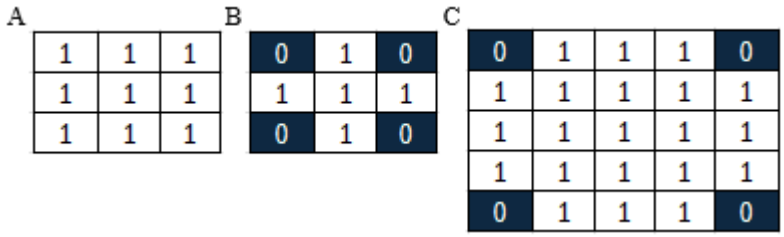


U Pr. 6.2. je kreirana `maska` čiji elementi imaju vrednost *True* ako intenzitet piksela na odgovarajućoj (x,y) poziciji na originalnoj slici zadovoljava uslov da je veći od 2 i manji od 6 (`originalna_slika>2)&(originalna_slika<6`), i čiji elementi imaju vrednost *False* ako navedeni uslov nije zadovoljen. Maskirana slika (`maskirana_slika`) se formira pomoću funkcije `np.where(maska,originalna_slika,0)` gde `maska` definiše logički uslov koji se primenjuje na `originalna_slika` tako što će intenzitet piksela biti postavljen na 0 tamo gde je element maske *False*. Na Pr.2B je prikazan rezultat izvršavanja: originalna slika, maskirana slika kao i sama logička tj. binarna maska.

6.3. Morfološke operacije

Morfološke operacije predstavljaju metode obrade slike koje se koriste za modifikaciju geometrijskih struktura na slici. Najčešće se primenjuju na binarne slike tj. maske. Morfološke operacije se zasnivaju na matematičkim operacijama morfološke obrade: erozija, dilatacija, otvaranje i zatvaranje. Tokom ovih operacija mala matrica koja se naziva strukturni element se pomera tako da se svojim centrom pozicionira iznad svakog elementa maske

koji ima vrednost „1“. Potom se primenjuje odgovarajuća matematička operacija između strukturnog elementa i piksela maske sa susedstvom (veličina susedstva je određena veličinom i oblikom strukturnog elementa). Primeri strukturnih elemenata su prikazani na Sl. 6.2.

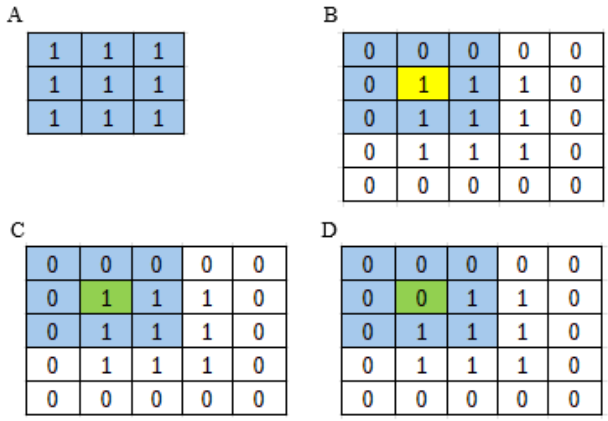


Slika 6.1. Primeri strukturnih elemenata

Na Sl. 6.2 je ilustrovana primena operacija dilatacije i erozije za strukturni element prikazan na Sl. 6.2A za piksel označen žutom bojom na Sl. 6.2B.

U slučaju primene operacije dilatacije, novi pikseli na dilatiranoj maski postavljaju se na „1“ ako je bar jedan od suseda piksela (prema strukturnom elementu) jednak „1“, Sl. 6.2C. Ako su svi susedi „0“, novi piksel će biti „0“. Dilatacija se koristi za proširivanje objekata ili ivica. Dilatacija se može primeniti u više iteracija, tj. više puta se može primeniti opisani postupak.

U slučaju primene operacije erozije, novi pikseli na erodiranoj maski se postavljaju na „1“ samo ako su svi susedni pikseli maske (prema strukturnom elementu) takođe jednaki „1“. Ako postoji bar jedan susedni piksel koji je „0“, onda će novi piksel biti „0“, Sl. 6.2D. Erozija se koristi za sužavanje objekata ili ivica. Erozija se može primeniti u više iteracija, tj. više puta se može primeniti opisani postupak.



Slika 6.2. A) Strukturni element, B) Originalna maska, C) Rezultat dilatacije za žuti piksel iz B, D) Rezultat erozije za žuti piksel iz B

Moguće je kombinovati eroziju i dilataciju i u zavisnosti od namene se radi prvo erozija pa dilatacija (ova operacija se naziva „otvaranje“) ili dilatacija pa erozija (ova operacija se naziva „zatvaranje“). Operacija otvaranja se koristi za uklanjanje sitnih detalja i šuma, a operacija zatvaranja se koristi za popunjavanje rupa u maski i za povezivanje bliskih objekata.

Na Pr. 6.3A je programski kôd koji, za pojednostavljeni primer maske sa jednim sitnim detaljem van osnovnog kvadratnog *ROI*, prikazuje efekat erozije i otvaranja. Za strukturni element je odabrana matrica kao na Sl. 6.1A. Operacije erozija i otvaranje su realizovane pomoću ugrađenih funkcija `ndi.binary_erosion()` i `ndi.binary_opening()` u jednoj iteraciji. Na Pr. 6.3B su prikazane redom: 1) originalna maska, 2) erodirana maska na kojoj je uklonjen sitan detalj koji je postojao na originalnoj maski van osnovnog kvadratnog *ROI* (pri čemu je i osnovna *ROI* sužena), i 3) maska nakon otvaranja, tj. nakon primene dilatacije na erodiranu masku. Može se uočiti da je maska nakon otvaranja identična originalnoj maski bez sitnog detalja.

Pr. 6.3. Primer *morfoloske_operacije.py* – primena operacija erozije i otvaranja

---

A. Programski kôd:

```
import numpy as np
import scipy.ndimage as ndi
import matplotlib.pyplot as plt

plt.close('all')
# Kreiranje 9x9 maske
maska=np.array([
    [0,0,0,0,0,0,0,0,0],
    [0,1,0,0,0,0,0,0,0],
    [0,0,1,1,1,1,1,0,0],
    [0,0,1,1,1,1,1,0,0],
    [0,0,1,1,1,1,1,0,0],
    [0,0,1,1,1,1,1,0,0],
    [0,0,1,1,1,1,1,0,0],
    [0,0,0,0,0,0,0,0,0],
    [0,0,0,0,0,0,0,0,0]
    ])
```

```
# Definisanje strukturnog elementa 3x3
strukturni_element = np.array([
    [1,1,1],
    [1,1,1],
    [1,1,1]
    ])

erod_maska = ndi.binary_erosion(maska,
                                structure=strukturni_element,
                                iterations=1)
otv_maska = ndi.binary_opening(maska,
                               structure=strukturni_element,
                               iterations=1)

plt.figure(2)
plt.subplot(1, 3, 1)
plt.title("maska", fontsize=10)
plt.imshow(maska, cmap='gray')
plt.axis('off')

plt.subplot(1, 3, 2)
plt.title("Erodirana maska", fontsize=10)
plt.imshow(erod_maska, cmap='gray')
plt.axis('off')

plt.subplot(1, 3, 3)
plt.title("Otvaranje", fontsize=10)
plt.imshow(otv_maska, cmap='gray')
plt.axis('off')
```

B. Rezultat izvršavanja:

Zadatak za samostalni rad 6.2. Modifikovati program Pr. 6.3 tako da:

1) primenjuje u jednoj iteraciji strukturni element sa Sl. 6.1B.
2) primenjuje dilataciju (funkcija `ndi.binary_dilation()`) i zatvaranje (`ndi.binary_closing()`) u jednoj iteraciji na modifikovanoj maski koja nema sitan detalj, ali ima „rupu“ (crni piksel unutar kvadratne *ROI*), a za strukturni element koristiti Sl. 6.1A.

Rešenje. Rešenje zadatka pod 1) je prikazano na Sl. 6.3, a rešenje zadatka pod 2) je prikazano na Sl. 6.4.

Slika 6.3. Ilustracija erozije i otvaranja pomoću strukturnog elementa u obliku krsta

Slika 6.4. Ilustracija dilatacije i zatvaranja

6.4. Hafova transformacija

Hafova transformacija (eng. *Hough Transform*) je algoritam koji se primenjuje za prepoznavanje specifičnih geometrijskih oblika npr. linija, krugova, elipse i sl. Najčešće se primenjuje na binarnu sliku tj. masku nastalu ili metodama detekcije ivica (pogledati Poglavlje 5) ili drugim metodama segmentacije (Podpoglavlja 6.6-6.9), ali se može primenjivati i na slike na skali sivog.

Hafova transformacija umesto u prostoru piksela radi u tzv. parametarskom prostoru u koji transformiše originalnu sliku. Na primer, ako je potrebno detektovati linije, parametarski prostor će biti polarni koordinatni sistem. Na Sl. 6.5 je dat prikaz dve prave u Dekartovom (x,y) koordinatnom sistemu i polarnom (ρ,θ) koordinatnom sistemu. Prava 1 (crvena prava) je u polarnom koordinatnom sistemu jednoznačno opisana parametrima ρ_1, (udaljenost početka koordinatnog sistema do prave) i θ_1 (ugao između normale na pravu povučene iz koordinatnog početka i x ose). Slično je prava 2 (plava prava) određena parametrima ρ_2 i θ_2. U opštem slučaju, prava u Dekartovom (x,y) prostoru se može predstaviti u polarnim koordinatama (ρ,θ) kao:

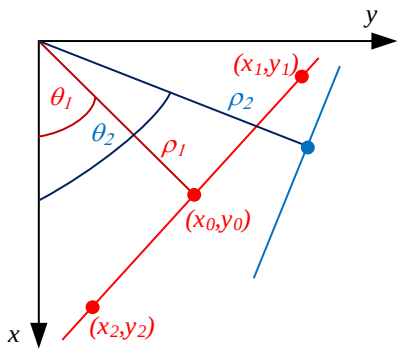
$$\rho = x \cdot cos\theta + y \cdot sin\theta \tag{6.1}$$

gde je ρ udaljenost od koordinatnog početka duž normale, θ ugao između normale na pravu i x ose, a (x,y) koordinata tačke na pravoj. Ukoliko je poznata tačka prave (x_0,y_0) (tačka preseka normale povučene iz koordinatnog početka i prave) i ugao $\theta=\theta_1$, moguće je odrediti i ostale tačke prave. Na primer, za tačke (x_1,y_1) i (x_2,y_2) sa različitih strana tačke (x_0,y_0) (vidi Sl. 6.5) važe sledeće relacije:

$$x_1 = x_0 + k \cdot cos\theta, \quad x_2 = x_0 - k \cdot cos\theta \tag{6.2}$$

$$y_1 = y_0 + k \cdot sin\theta, \quad y_2 = y_0 - k \cdot sin\theta \tag{6.3}$$

gde je k parametar koji se menja da bi se izračunale različite tačke prave.

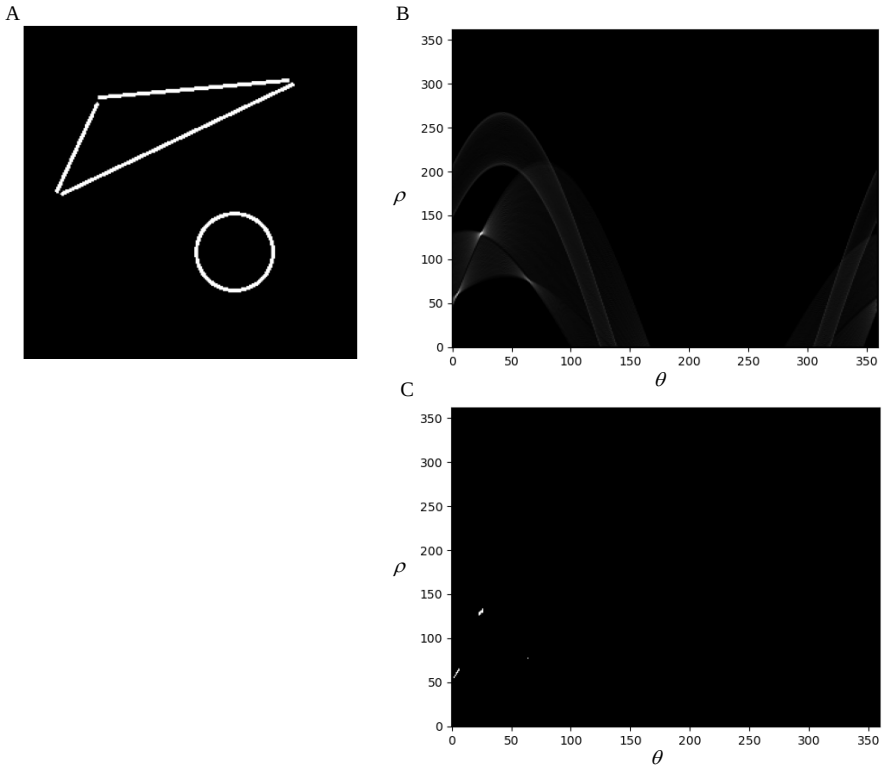


Slika 6.5. Prikaz prave u Dekartovom i polarnom koordinatnom sistemu

Hafova transformacija za detekciju linija obuhvata sledeće korake:

- primena algoritma za detekciju ivica (npr. *Canny* filtra, pogledati Poglavlje 5.2.2) – ovaj korak nije uvek neophodno primeniti
- transformacija slike u parametarski prostor primenom relacije (6.1): sve tačke koje u Dekartovom koordinatnom sistemu pripadaju istoj liniji će imati iste vrednosti za ρ i θ, a odgovarajući element parametarske matrice će imati vrednost koja je jednaka broju tačaka jedne linije
- određivanje lokalnih maksimuma u parametarskoj matrici i selekcija onih maksimuma koji su veći od praga koji definiše korisnik
- povratak u Dekartov koordinatni sistem, tj. izračunavanje (x_0,y_0), (x_1,y_1) i (x_2,y_2) pomoću relacija (6.2) i (6.3) radi crtanja detektovanih linija.

Na Sl. 6.6A je prikazana slika koja sadrži 3 linije i krug, a na Sl. 6.6B je prikazana njena (ρ,θ) parametarska matrica na kojoj se mogu uočiti tri svetle tačke koje odgovaraju trima linijama. Na Sl. 6.6C je prikazana parametarska matrica nakon primene praga 80, tj. samo oni elementi parametarske matrice koji imaju više od 80 tačaka su kandidati za linije.



Slika 6.6. A) Originalna slika, B) Parametarska matrica (ρ,θ), C) Parametarska matrica nakon primene praga 80

Na Pr. 6.4A je prikazan programski kôd koji koristi `cv2.HoughLines()` funkciju za određivanje linija. Kao ulazna slika je korišćena maska prikazana na Sl. 6.6A. Pri proračunu parametarske matrice su uzeti koraci: 1 piksel za ρ (`rho_korak`) i 1 stepen za θ (`theta_korak`). Da bi na jednoj slici bili prikazani i originalna `slika` i detektovane linije, pravi se kopija ulazne slike `slika_sa_linijama` u RGB sistemu pomoću funkcije: `cv2.cvtColor(slika, cv2.COLOR_GRAY2BGR)`. Kada se za svaku liniju detektovanu Hafovom transformacijom odrede (x_0,y_0), (x_1,y_1) i (x_2,y_2) pomoću relacija (6.2) i (6.3), onda se pomoću funkcije `cv2.line()` iscrtavaju linije zelene boje (argument (R,G,B)=`(0,255,0)`) i debljine 2 piksela (argument `2`). Parametar k je izabran da bude dovoljno veliki (`k=1000`) da nacrtane linije sigurno pokrivaju celu sliku. Rezultat detekcije linija je prikazan na Pr. 6.4B.

Pr. 6.4. Primer *Hafova_transformacija_linija.py* – primena Hafove transformacije za detekciju linija

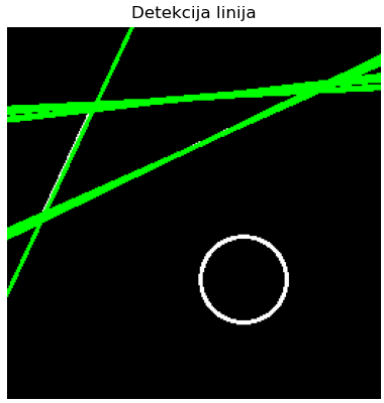
---

A. Programski kôd:

```
import cv2
import numpy as np
import matplotlib.pyplot as plt

plt.close("all")

def Haf_linije(slika):
    rho_korak = 1
    theta_korak = np.pi / 180
    prag = 80
    linije = cv2.HoughLines(slika,
                            rho_korak,
                            theta_korak,
                            prag)

    # Kopija slike za crtanje linija
    slika_sa_linijama = cv2.cvtColor(slika,
                                     cv2.COLOR_GRAY2BGR)

    k=1000
```

```
    for i in linije:
        rho, theta = i[0]
        x0 = rho * np.cos(theta)
        y0 = rho * np.sin(theta)
        x1 = int(x0 - k * np.sin(theta))
        x2 = int(x0 + k * np.sin(theta))
        y1 = int(y0 + k * np.cos(theta))
        y2 = int(y0 - k * np.cos(theta))
        cv2.line(slika_sa_linijama,
                 (x1, y1), (x2, y2),
                 (0, 255, 0), 2)
    return slika_sa_linijama

# Ucitavanja slike
slika_RGB=cv2.imread('linije_krug.png')
slika_grayscale=cv2.cvtColor(slika_RGB, cv2.COLOR_BGR2GRAY)

# Detekcija i prikaz linija
slika_linije = Haf_linije(slika_grayscale)
plt.imshow(slika_linije, cmap='gray')
plt.title("Detekcija linija")
plt.axis("off")
```

B. Rezultat izvršavanja:



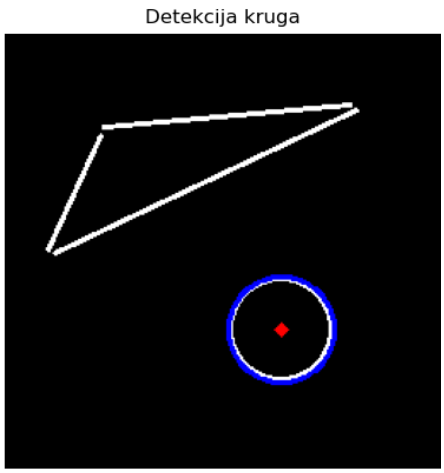
Zadatak za samostalni rad 6.3. Implementirati opisane korake Hafove transformacije i prikazati međukorake prikazane na Sl. 6.6B i Sl. 6.6C za ulaznu sliku Sl. 6.6A.

Zadatak za samostalni rad 6.4. Hafova transformacija za detekciju kružnice koristi sledeću parametarsku jednačinu:

$$r^2 = (x-a)^2 + (y-b)^2 \tag{6.4}$$

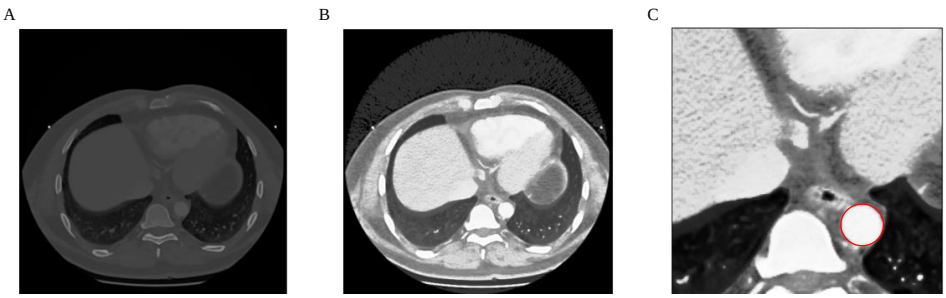
gde su (a,b) koordinate centra kružnica, a r radijus kružnice. Primenom ove jednačine i funkcije `cv2.HoughCircles()` detektovati kružnicu na ulaznoj slici Sl. 6.6A i odrediti centar detektovane kružnice.

Rešenje. Rezultat detekcije kružnice je prikazan na Sl. 6.7.



Slika 6.7. Detekcija kružnice Hafovom transformacijom

Na Sl. 6.8 je prikazan primer detekcije nishodnog dela aorte na ekvalizovanom *CT* snimku abdomena [Đulinac et al. 2024] pomoću Hafove transformacije za detekciju kružnice.



Slika 6.8. A) Originalan *CT* snimak, B) Ekvalizovan *CT* snimak, C) Deo slike od interesa sa izdvojenom aortom kružnog oblika (slika preuzeta iz [Đulinac et al. 2024] i modifikovana)

6.5. Transformacija rastojanja

Transformacija rastojanja (eng. *Distance Transform*) se primenjuje na binarnim slikama i predstavlja izračunavanje rastojanja svakog piksela sa vrednošću „1“ (beli pikseli) na slici od najbližeg piksela sa vrednošću „0“ (najbliži crni piksel, tj. pozadinski piksel). Rezultat transformacije rastojanja se obično prikazuje u vidu *grayscale* mape rastojanja, tzv. topografske mape, koja se dobija tako što se vrednost svakog belog piksela zameni vrednošću rastojanja.

Rastojanje belih piksela od najbližeg crnog piksela se može računati primenom različitih metoda, tj. razlikuju se Euklidsko, *Manhattan*[38] i *Chebyshev*[39] rastojanje. Za analizu slike se standardno koristi Euklidsko rastojanje koje precizno računa udaljenost između tačaka (x_1, y_1) i (x_2, y_2) prema sledećoj formuli:

$$d_E = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}. \tag{6.5}$$

Geometrijski centar (x_C, y_C) tj. centroid slike ili regiona se definiše na sledeći način:

$$x_C = \frac{\sum_{i=1}^{N} x_i \cdot \rho_i}{\sum_{i=1}^{N} \rho_i} \tag{6.6}$$

$$y_C = \frac{\sum_{i=1}^{N} y_i \cdot \rho_i}{\sum_{i=1}^{N} \rho_i} \tag{6.7}$$

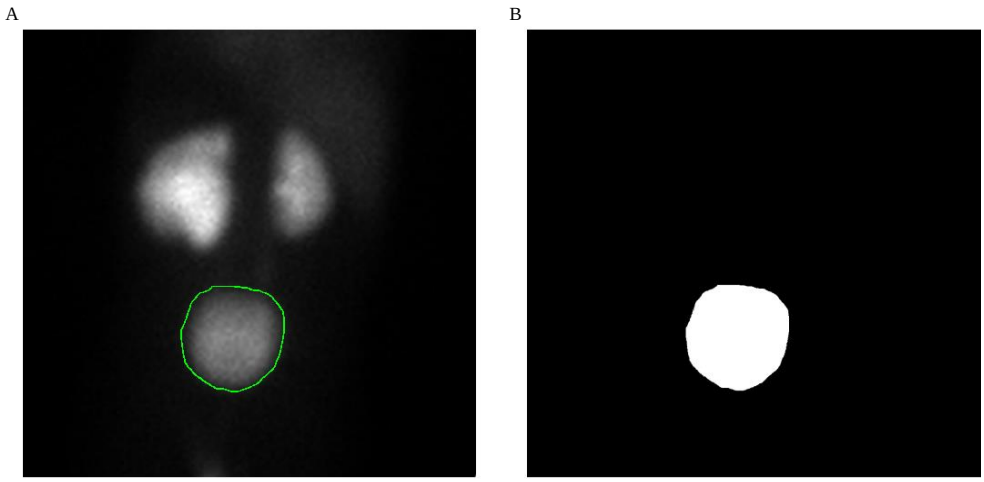
gde je N broj piksela binarne slike, (x_i, y_i) je pozicija i-tog piksela, a ρ_i je intenzitet i-tog piksela.

Na Sl. 6.9 je prikazan rezultat manuelne segmentacije bešike na usrednjenom renogramu korišćenom u Pr. 6.1 kao i izgled maske koja odgovara zaokruženom regionu bešike. Na Pr. 6.5 je prikazan programski kôd koji primenjuje transformaciju rastojanja na kreiranu masku „maska_ROI_besika.png“ i prikazuje je u *grayscale* paleti boja uz određivanje maksimalnog rastojanja i centroida. Nakon učitavanja RGB maske i konverzije u sliku sa vrednostima intenziteta piksela „0“ i „1“, za određivanje mape Euklidskih rastojanja (`mapa_rastojanja`) primenjena je funkcija `ndi.distance_transform_edt()` za sve piksele čija je vrednost „1“ (argument `maska == 1`). Pri tome je za svaki piksel slike određen i `indeks` najbližeg „pozadinskog“ piksela (argument `return_indices=True`). Maksimalno rastojanje (`rastojanje_max`) je određeno funkcijom

[38] *Manhattan* rastojanje između između tačaka (x_1, y_1) i (x_2, y_2) se definiše kao:
$$d_M = |x_2 - x_1| + |y_2 - y_1|$$

[39] *Chebyshev* rastojanje između tačaka (x_1, y_1) i (x_2, y_2) se definiše kao:
$$d_C = \max(|x_2 - x_1|, |y_2 - y_1|)$$

`ndi.maximum()` koja je primenjena na mapu rastojanja, a pozicija piksela pozadine za maksimalno rastojanje u mapi rastojanja (`pozicija_max`) je određena funkcijom `ndi.maximum_position()`. Centroid je određen funkcijom `ndi.center_of_mass()` koja je primenjena na mapi rastojanja, a prema relacijama (6.6) i (6.7).



Slika 6.9. A) Manuelna segmentacija *ROI* bešike, B) Maska za *ROI* pod A)

Pr. 6.5. Primer *transformacija_rastojanja.py* – vizuelizacija mape rastojanja, maksimalnog rastojanja i centroida

---

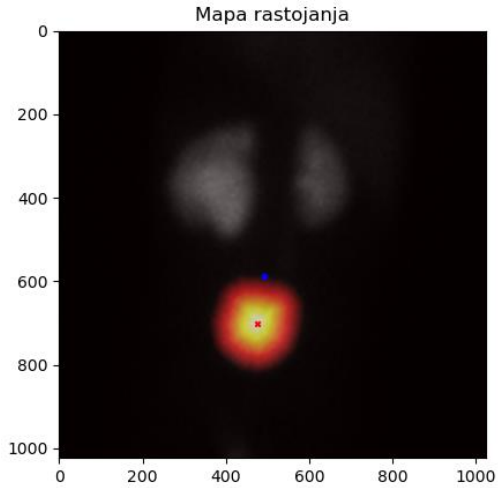
A. Programski kôd:

```
import cv2
import scipy.ndimage as ndi
import matplotlib.pyplot as plt

plt.close('all')

# Učitavanja slike
slika = cv2.imread('usrednjen_renogram.jpg')
# Učitavanje maske
maska_RGB=cv2.imread('maska_ROI_besika.png')
maska=cv2.cvtColor(maska_RGB, cv2.COLOR_BGR2GRAY)
maska[maska == 255] = 1
```

```
# Primena transformacije rastojanja
mapa_rastojanja, indeks = ndi.distance_transform_edt(
    maska == 1, return_indices=True)

rastojanje_max = ndi.maximum(mapa_rastojanja)
print('Maksimalno rastojanje: ', rastojanje_max)
pozicija_max = ndi.maximum_position(maska)
print('Pozicija max rastojanja: ', pozicija_max)
centar = ndi.center_of_mass(mapa_rastojanja)
print('Pozicija centra', centar)

# Mapa rastojanja
plt.imshow(slika, cmap='gray')
plt.imshow(mapa_rastojanja, cmap='hot', alpha=0.6)
plt.title('Mapa rastojanja')
plt.scatter(pozicija_max[1], pozicija_max[0], s=10,
            c='blue', marker='o')
plt.scatter(centar[1], centar[0], s=10, c='red', marker='x')
```

B. Rezultat izvršavanja:

```
Maksimalno rastojanje:  111.00450441310929
Pozicija max rastojanja:  (587, 491)
Pozicija centra (700.389662453972, 474.85213280631547)
```



Slika maske je prekrivena mapom rastojanja u paleti boja `hot` sa nivoom transparencije od 60% (`alpha=0.6`), a primenom funkcije `plt.scatter()` su označeni piksel pozadine na maksimalnom rastojanju (plavi kružić veličine 10 kvadratnih tačaka) i centroid (crveni krstić veličine 10 kvadratnih tačaka). Rezultat izvršavanja je dat na Pr. 6.5B.

Transformacija rastojanja u biomedicinskom slikanju se primenjuje za segmentaciju u slučajevima kada postoje objekti koji se preklapaju (npr. segmentacija primenom vododelnica, Potpoglavlje 6.9), za merenje geometrijskih karakteristika objekata (npr. prečnika krvnih sudova, debljine tkiva i sl.) kao i za analizu oblika (npr. detekcija anomalija).

6.6. Metode segmentacije zasnovane na statistici histograma

Metode segmentacije zasnovane na statistici histograma koriste informaciju o raspodeli intenziteta piksela na slici da bi podelile sliku na regije. Ovakve metode za svoju efikasnu primenu podrazumevaju da unutar svake regije pikseli imaju slične intenzitete, a da se intenziteti piksela između regija razlikuju. Pod ovakvom pretpostavkom, izborom odgovarajućeg globalnog praga/ova se na osnovu intenziteta piksela mogu izdvojiti regioni od interesa.

Zadatak za samostalni rad 6.5. Za *CT* sliku („0.dcm“) je u zadatku za samostalni rad 4.1 prikazan normalizovani histogram. Napraviti modifikaciju tako da bude prikazan histogram bez normalizacije, a potom formirati tri maske:

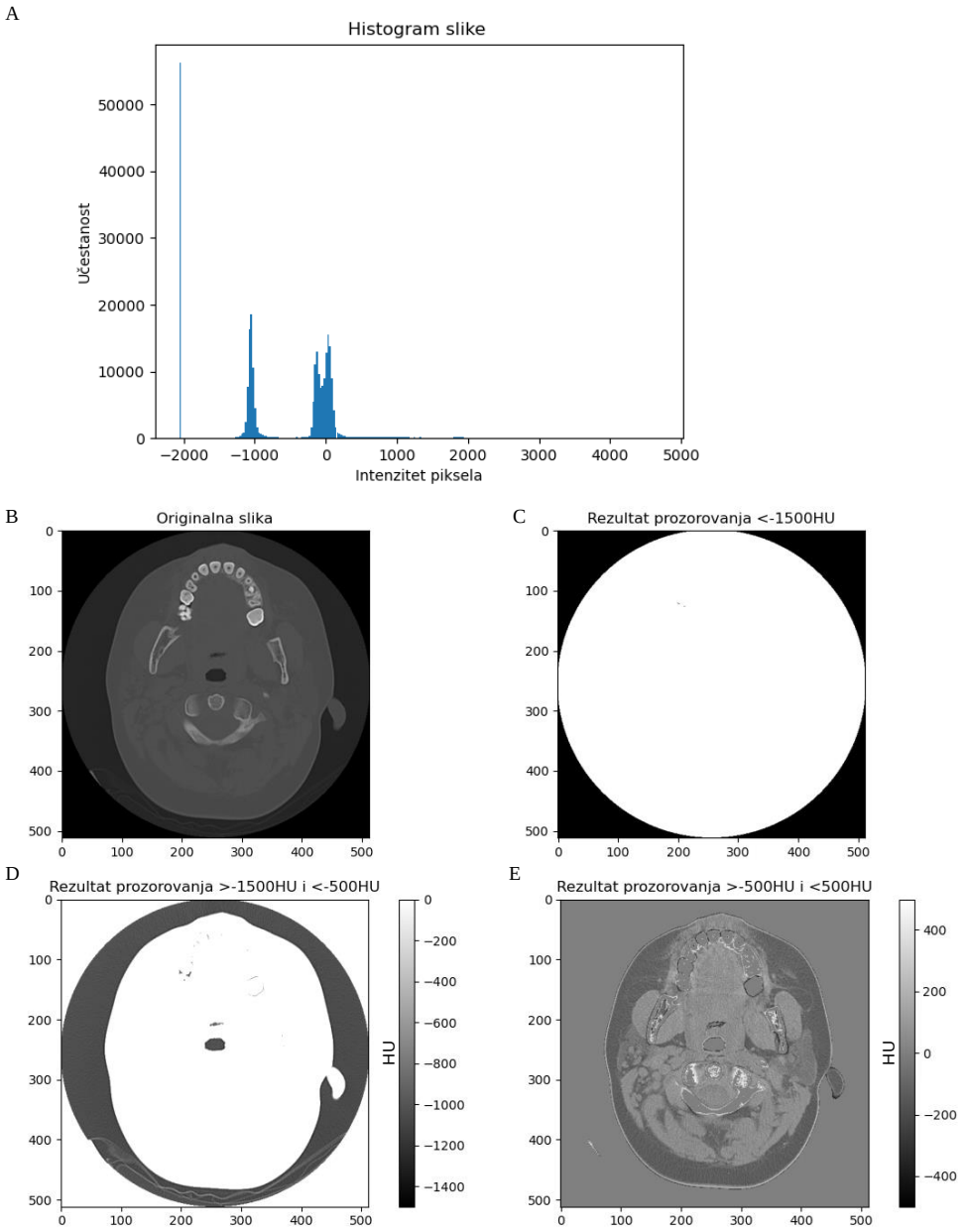
- maska1 – ima piksele sa vrednošću „1“ na pozicijama onih piksela *CT* slike koji imaju vrednost manju od -1500 HU
- maska2 – ima piksele sa vrednošću „1“ na pozicijama onih piksela *CT* slike koji imaju vrednost između -1500 HU i -500 HU
- maska3 – ima piksele sa vrednošću „1“ na pozicijama onih piksela *CT* slike koji imaju vrednost između -500 HU i 500 HU.

Prikazati slike dobijene prozorovanjem, tj. množenjem originalne *CT* slike sa svakom od opisanih maski.

Rešenje:

Na Sl. 6.10A je prikazan histogram *CT* slike sa tri pika, a na Sl. 6.10B-E su prikazane redom originalna *CT* slika, i rezultati njenog množenja sa maskama koje odgovaraju prozorima <-1500 HU, [-1500, -500] HU, [-500,500] HU, respektivno. Izabrani prozori odgovaraju opsezima tri pika prikazanog histograma na Sl. 6.10A. Može se uočiti da Sl. 6.10C odgovara vazduhu, a da

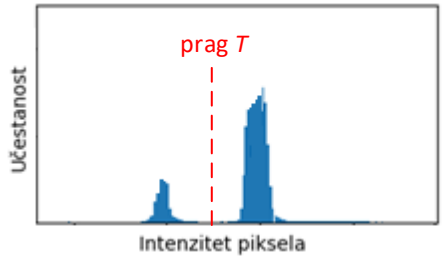
Sl. 6.10E izdvaja glavu pacijenta. Na ovako dobijene slike prozorovanjem je moguće izvršiti poboljšanje kontrasta dodatnom primenom metoda transformacija intenziteta opisanih u Poglavlju 4.



Slika 6.10 A) Histogram sa tri pika, B) Originalna *CT* slika, C) Efekat primene maske1 (< -1500 HU), D) Efekat primene maske2 (prozor [-1500, -500] HU), E) Efekat primene maske3 (prozor [-500, 500] HU)

6.6.1. Otsu metoda

Otsu metoda predstavlja algoritam koji se standardno koristi za automatsko određivanje globalnog praga za histogram s ciljem segmentacije slike. Ovaj algoritam je najefikasniji za tzv. bimodalni histogram, tj. histogram koji ima dva razdvojena pika od kojih jedan pik odgovara svetlijim pikselima (eng. *foreground*), dok drugi pik odgovara tamnijim pikselima (eng. *background*). Cilj Otsu metode je nalaženje praga T, Sl. 6.11. koji maksimizuje varijansu između klase svetlih i klase tamnih piksela (interklasna varijansa) omogućavajući da se na taj način segmentiraju tj. izdvoje svetli pikseli.



Slika 6.11 Bimodalni histogram i Otsu prag *T*

Koraci Otsu metode za optimalno određivanje praga T su sledeći:

- Određivanje histograma slike
- Izračunavanje verovatnoće za intenzitet ρ prema jednačini:

$$P_\rho = \frac{N_\rho}{N} \tag{6.8}$$

  gde je N_ρ broj piksela sa intenzitetom ρ, a N je ukupan broj piksela slike.

- Za različite vrednosti praga T, pikseli slike se dele u dve klase: prva klasa (tamniji pikseli) kojoj pripadaju pikseli sa intenzitetom $\rho<T$ i druga klasa (svetliji pikseli) kojoj pripadaju pikseli sa intenzitetom $\rho \geq T$.
- Određivanje verovatnoće P_1 za prvu klasu i P_2 za drugu klasu:

$$P_1 = \sum_{\rho<T} P_\rho \tag{6.9}$$

$$P_2 = \sum_{\rho\geq T} P_\rho \tag{6.10}$$

- Određivanje srednje vrednosti intenziteta piksela celog histograma, μ:

$$\mu = P_1 \cdot \mu_1 + P_2 \cdot \mu_2 \tag{6.11}$$

  gde su μ_1 i μ_2 srednje vrednosti za prvu i drugu klasu, respektivno.

- Određivanje interklasne varijanse σ^2_{inter} za prvu i drugu klasu prema relaciji:

$$\sigma^2_{inter} = P_1 \cdot P_2 \cdot (\mu_1 - \mu_2)^2 \quad (6.12)$$

- Određivanje praga T tako da interklasna varijansa σ^2_{inter} bude maksimalna, tj. za koje će biti najmanje preklapanje između prve i druge klase.

Na Pr. 6.6A je prikazan programski kôd koji segmentira tumor glave primenom Otsu metode i morfoloških operacija na „Cancer (33).tif“ datoteci iz direktorijuma `CT_tumor` dostupnog u okviru baze podataka *Computed Tomography (CT) of the Brain* koja je opisana u Poglavlju 3.3.2. Rezultat izvršavanja kôda je prikazan na Pr. 6.6B.

Pr. 6.6. Primer *Otsu_metoda.py* – segmentacija tumora na *CT* snimku glave

A. Programski kôd:

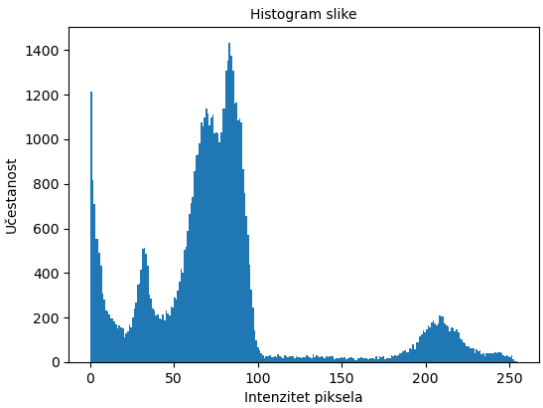
```
import cv2
from skimage.filters.thresholding import threshold_otsu
import matplotlib.pyplot as plt
import numpy as np
import scipy.ndimage as ndi

plt.close('all')
# Ucitavanja slike
slika_RGB=cv2.imread('Cancer (33).tif')
slika_grayscale=cv2.cvtColor(slika_RGB, cv2.COLOR_BGR2GRAY)
# Prikaz histograma
plt.figure()
histogram=plt.hist(slika_grayscale.ravel(),
         bins=256, range=(0, 255))
plt.xlabel('Intenzitet piksela', fontsize=10)
plt.ylabel('Učestanost', fontsize=10)
plt.title('Histogram slike', fontsize=10)

# Primena Otsu metode
prag=threshold_otsu(slika_grayscale)
```

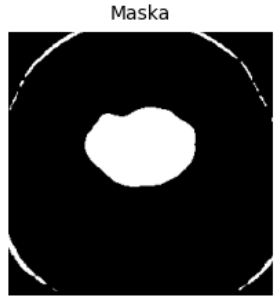
```
print('Vrednost praga T=', prag)

# Kreiranje maske na osnovu Otsu praga
maska = np.where(slika_grayscale>prag, 1, 0)

# Primena morfoloških operacija
erod_maska = ndi.binary_erosion(maska, iterations=3)
dil_maska = ndi.binary_dilation(erod_maska, iterations=3)

# Labeliranje objekata
labele, br_labela = ndi.label(dil_maska)
print('Broj objekata: ', br_labela)

# Izdvajanje objekta koji je tumor
tumor = labele == 1
# Kreiranje overlay-a
overlay = np.where(labele == 1, labele, np.nan)

# Prikaz koraka obrade
outputs = [
    ("Originalna slika", slika_grayscale),
    ("Maska", maska),
    ("Maska nakon erozije", erod_maska),
    ("Maska nakon erozije i dilatacije", dil_maska),
    ("Labele", labele),
    ("Tumor", tumor),
]
plt.figure()
for i, (title, image) in enumerate(outputs, start=1):
    plt.subplot(2, 3, i)
    plt.title(title, fontsize=10)
    plt.imshow(image, cmap='gray')
    plt.axis('off')
```

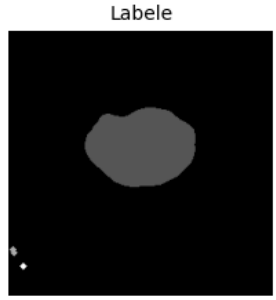
```
# Prikaz segmentiranog tumora na slici
plt.figure()
plt.imshow(slika_grayscale, cmap='gray')
plt.imshow(overlay, cmap='rainbow', alpha=0.3)
plt.axis('off')
plt.title('Originalna slika sa izdvojenim tumorom')
```

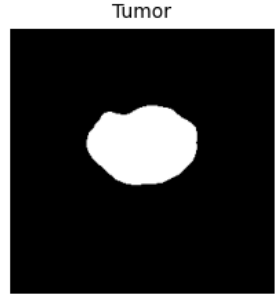
B. Rezultat izvršavanja:

```
Vrednost praga T= 133
Broj objekata:  3
```









Nakon učitavanja slike i konverzije u *grayscale* paletu boja, i iscrtavanja histograma, primenjena je funkcija `threshold_otsu()` za određivanje Otsu praga za segmentaciju. Dobijena vrednost praga je T=133 koja na histogramu predstavlja vrednost intenziteta piksela između dva odvojena pika. Pikseli sa intenzitetom većim od praga T su kandidati za region tumora. Pomoću funkcije `np.where()` je formirana `maska` koja sadrži sve piksele koji su kandidati za region tumora. S obzirom na to da maska sadrži i piksele koji ne pripadaju regionu tumora, potrebno je primeniti morfološke operacije, najpre eroziju primenom funkcije `ndi.binary_erosion()` (u 3 iteracije sa *default*-nim strukturnim elementom kao na Sl. 6.2A), a potom dilataciju pomoću funkcije `ndi.binary_dilation()` (u 3 iteracije sa takođe *default*-nim strukturnim elementom). Na erodiranu i dilatiranu masku je primenjena funkcija za labeliranje objekata `ndi.label()`. Broj detektovanih objekata (`broj_labela`) je ovom slučaju 3 jer osim regiona tumora postoje još dva manja detektovana objekta (razmisliti koje morfološke operacije bi mogle da se primene tako da ovih malih objekata ne bude na obrađenoj maski). Matrica `labele` sadrži broj labele za svaki piksel maske na koju je labeliranje primenjeno. Na primer, u slučaju 3 detektovana objekta, svaki element matrice `labele` je jednak 1,2 ili 3 u zavisnosti od toga kojem objektu pripada. Objekat koji sadrži piksele tumora u ovom slučaju ima labelu „1“ i od ovog objekta je formiran `overlay` koji je prikazan sa transparencijom 30% (`alpha=0.3`) preko originalne *grayscale* slike. *Overlay* je maska koja ima vrednost NaN za sve piksele koji ne pripadaju posmatranoj regiji tumora, a za piksele koji pripadaju posmatranoj regiji tumora vrednost je jednaka vrednosti labele (u ovom slučaju „1“).

6.7. Metode segmentacije zasnovane na regionima

Metoda rasta regiona (eng. *Region growing*) je jedna od najčešće korišćenih metoda segmentacija zasnovanih na regionima. Ovaj metod izdvaja *ROI* kao region koji je homogen po određenom kriterijumu. Metoda rasta regiona se sastoji iz sledećih koraka:

- zadavanje početne tačke (eng. *seed*) od strane korisnika
- širenje regiona od zadate početne tačke prema susednom pikselu:
    - po kriterijumu sličnosti intenziteta piksela (razlika u intenzitetu piksela posmatranog piksela i susednog piksela je manja od definisanog praga)
    - po kriterijumu sličnosti teksture (vidi Poglavlje 7)
    - po kriterijumu prostorne povezanosti sa susedstvom (4-*connected* ili 8-*connected*) i sl.

- zaustavljanje širenja regiona ako ne postoji susedni piksel za koji je kriterijum širenja iz prethodne tačke zadovoljen ili ako je dostignuta unapred zadata veličina regiona.

Metoda rasta regiona je jednostavna za implementaciju i efikasna za izdvajanje homogenih regiona. Osnovni nedostaci leže u osetljivosti rezultata segmentacije na izbor početne tačke i osetljivosti rezultata na šum na slici. Takođe, zbog lošeg kontrasta na slici može nastati efekat „curenja“ regiona zbog čega se preporučuje primena metoda poboljšanje kontrasta (Poglavlje 4) ili isticanje anatomskih ivica (Poglavlje 5) pre primene ove metode segmentacije.

Na Pr. 6.7A je prikazan programski kôd koji segmentira strukturu *corpus callosum* (lat.) na izabranom *MRI* snimku glave „Caltech\0051456\session_1\abide_msp.nii“ iz baze ABIDE[40] [Kucharsky et al. 2015]. Nakon učitavanja slike i konvertovanja iz *SimpleITK.Image* tipa podataka u odgovarajuću matricu što se postiže pomoću funkcije `sitk.GetArrayViewFromImage()`, proverena je dimenzija originalne slike (u ovom slučaju originalni podaci su organizovani kao 3D matrica, ali imaju samo jedan slajs). Opštosti radi, naredni korak izdvaja željeni slajs pomoću funkcije `sitk.Extract()`. Potom se izdvojeni `slajs_slika` tipa *SimpleITK.Image* konvertuje u matricu `slajs_matrica`. Za primenu metode rasta regiona je neophodno zadati koordinate početnog semena (`seme = [(235, 200)]`) i kriterijum širenja. Za širenje regiona je u ovom primeru iskorišćena funkcija `sitk.ConnectedThreshold()` sa kriterijumom širenja zasnovanom na 4-*connected* povezanosti i sličnosti intenziteta piksela zadatoj donjim i gornjim pragom (`donji_prag = 340`, `gornji_prag = 490`). Rezultat izvršavanja kôda je prikazan na Pr. 6.7B: originalni slajs sa zadatim semenom (crvena tačka) i segmentirani *corpus callosum*.

Pr. 6.7. Primer *metoda_rasta_regiona.py* – segmentacija *corpus callosum*-a na *MRI* snimku glave metodom rasta regiona

---

A. Programski kôd:

```
import SimpleITK as sitk
import matplotlib.pyplot as plt


plt.close('all')
```

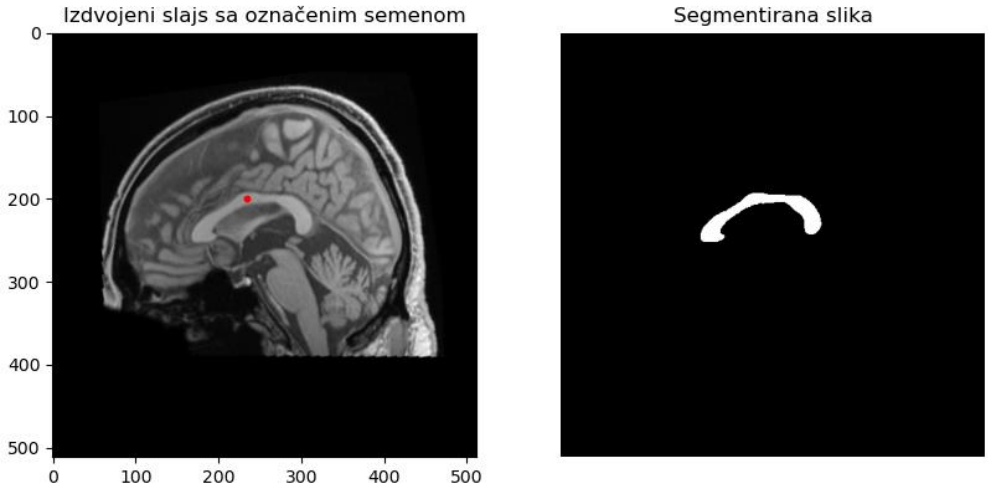
[40] Baza podataka *ABIDE corpus callosum and brain segmentation data* je dostupna na sledećem linku: https://sites.google.com/site/hpardoe/cc_abide (poslednji pregled Decembar 2024).

```
# Učitavanje slike
slika = sitk.ReadImage("abide_msp.nii", sitk.sitkFloat32)

# Konvertovanje slike u matricu
slika_matrica = sitk.GetArrayViewFromImage(slika)

# Provera dimenzija slike
print("Dimenzije slike:", slika_matrica.shape)

# Izdvajanje slajsa kao SimpleITK.Image
slajs_indeks = 0
velicina = [slika.GetSize()[0], slika.GetSize()[1], 0]
indeks = [0, 0, slajs_indeks]  # Početna tačka za slice
slajs_slika = sitk.Extract(slika, size=velicina, index=indeks)

# Konvertovanje slajsa u matricu
slajs_matrica = sitk.GetArrayFromImage(slajs_slika)

# Zadavanje semena
seme = [(235, 200)]
# Zadavanje pragova intenziteta piksela
donji_prag = 340
gornji_prag = 490

# Metoda rasta regiona
segmentacija = sitk.ConnectedThreshold(
    image1=slajs_slika,
    seedList=seme,
    lower=donji_prag,
    upper=gornji_prag
)

# Konvertovanje u matricu
segmentacija_matrica                                    =
sitk.GetArrayViewFromImage(segmentacija)
```

```
plt.subplot(1,2,1)
plt.imshow(slajs_matrica, cmap='gray')
plt.title("Izdvojeni slajs sa označenim semenom")
plt.scatter(seme[0][0], seme[0][1], c='red', s=10)

plt.subplot(1,2,2)
plt.imshow(segmentacija_matrica, cmap='gray')
plt.title('Segmentirana slika')
plt.axis('off')
plt.show()
```

B. Rezultat izvršavanja:

```
Dimenzije slike: (1, 512, 512)
```



Zadatak za samostalni rad 6.6. Za Pr. 6.7 varirati poziciju semena i vrednosti donjeg i gornjeg praga intenziteta piksela i ispitati efekat ovih varijacija na rezultat segmentacije primenom metode rasta regiona.

Zadatak za samostalni rad 6.7. Isprobati primenu metode rasta regiona na *CT* snimku „Cancer (270).jpg“ iz direktorijuma `CT_tumor` dostupnog u okviru baze podataka *Computed Tomography (CT) of the Brain* koja je opisana u Poglavlju 3.3.2. Prokomentarisati (ne)uspešnost metode rasta regiona na ovom primeru.

U slučaju heterogenih slika, može se primeniti tzv. metoda deljenja regiona (eng. *Region splitting*) kod koje se cela slika deli na manje regione sve dok ne bude zadovoljen uslov homogenosti u regionima. Ovakav pristup može dovesti do preterane segmentacije. Stoga se metod može korigovati u tzv. metod deljenja i spajanja (eng. *Region splitting and merging*) koji nakon deobe na regione proverava da li postoji sličnost između nastalih regiona (i između onih koji mogu biti i udaljeni međusobno). Potom se ukoliko postoji sličnost, region spaja (eng. *merge*) sa najsličnijim.

6.8. Metode segmentacije zasnovane na aktivnim konturama

Metode zasnovane na aktivnim konturama (eng. *snakes*) omogućavaju segmentaciju slike korišćenjem deformabilnih krivih koje se u iterativnom procesu prilagođavaju ivicama *ROI* koji se detektuje. Na početku procesa segmentacije, neophodno je označiti na slici inicijalnu konturu. Potom se u iterativnom procesu ova inicijalna kontura deformiše pod uticajem unutrašnjih i spoljašnjih sila sve dok se ne pronađe stabilna i glatka kontura sa minimalnom energijom koja oivičava *ROI*. Za detaljnije informisanje o modelu aktivnih kontura preporučena literatura je [Kass et al. 1988], a ovde će u nastavku poglavlja biti iznete matematičke osnove neophodne za razumevanje podešavanja parametara ove metode.

Aktivna kontura se matematički predstavlja kao kriva $K(s)=[x(s), y(s)]$, pri čemu je $s\in[0,1]$ parametar koji opisuje položaj tačke na krivoj. Energija aktivne konture E_{AK} se predstavlja na sledeći način:

$$E_{AK} = E_{unutrašnja} + E_{slika} + E_{spoljašnja} \tag{6.13}$$

gde je $E_{unutrašnja}$ unutrašnja energija, E_{slika} energija slike i $E_{spoljašnja}$ spoljašnja energija (dodatne sile). Unutrašnja energija kontroliše glatkoću i kontinuitet aktivne konture i definiše se na sledeći način:

$$E_{unutrašnja} = alpha \cdot \left|\frac{\partial K}{\partial s}\right|^2 + beta \cdot \left|\frac{\partial^2 K}{\partial s^2}\right|^2 \tag{6.14}$$

gde parametar *alpha* kontroliše dužinu aktivne konture (veće *alpha* utiče na to da kontura bude kraća, tj. *alpha* penalizuje dužinu aktivne konture), a parametar *beta* kontroliše glatkoću aktivne konture (veće *beta* utiče na to da kontura bude glatkija).

Energija slike usmerava aktivnu konturu ka važnim karakteristikama slike kao što su ivice slike tj. ka regiji sa jakim gradijentima i definiše se na sledeći način:

$$E_{slika} = -|\nabla I(x,y)|^2 \tag{6.15}$$

gde je $I(x,y)$ intenzitet slike, a ∇I predstavlja gradijent slike.

Osim parametara *alpha* i *beta*, algoritam aktivnih kontura ima još jedan ulazni parametar *gamma* koji predstavlja korak optimizacije i služi za kontrolisanje brzine deformacije aktivne konture (ako je *gamma* malo kontura se sporije menja, a konvergencija je stabilnija, ali je ovakav algoritam računarski zahtevniji).

Na Pr. 6.8A je prikazan programski kôd koji omogućava detekciju *ROI* tumora na *CT* snimku „Cancer (270).jpg“ iz direktorijuma `CT_tumor` dostupnog u okviru baze podataka *Computed Tomography (CT) of the Brain* koja je opisana u Poglavlju 3.3.2. Nakon učitavanja snimka i konvertovanja u *grayscale* paletu boja, formira se `inicijalna_kontura` u obliku kružnice sa centrom u tački `(270,175)` i radijusom `40` piksela. Potom se pomoću funkcije `active_contour()` iz *skimage* biblioteke pokreće proces deformacije inicijalne konture tako da se minimizuje energija konture date relacijom (6.13). Pri tome se za ulaznu sliku uzima *grayscale* slika na koju je primenjen Gaussian filter (`sigma=2`). Parametri *alpha* i *beta* iz relacije (6.14) su podešeni na `0.01` i `0.3`, respektivno, a korak optimizacije *gamma* na `0.001`. Maksimalan broj iteracija je podešen na `300` iteracija. Rezultat izvršavanja kôda je prikazan na Sl. 6.8B gde je plavom linijom označena inicijalna kontura, a crvenom linijom *ROI* tumora segmentiran metodom aktivnih kontura.

Pr. 6.8. Primer *aktivna_kontura.py* – segmentacija tumora na *CT* snimku glave metodom aktivnih kontura

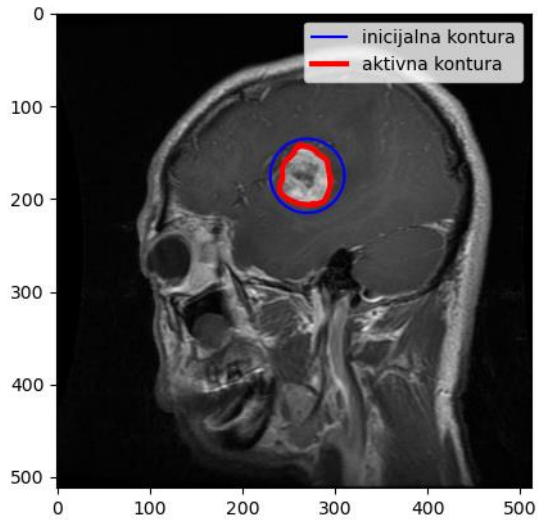
---

A. Programski kôd:

```
import cv2
import numpy as np
from skimage.filters import gaussian
from skimage.segmentation import active_contour
import matplotlib.pyplot as plt

plt.close('all')
slika_RGB=cv2.imread('Cancer (270).jpg')
slika_grayscale=cv2.cvtColor(slika_RGB, cv2.COLOR_BGR2GRAY)

ugao = np.linspace(0, 2 * np.pi, 400)
centar = (270, 175)
radijus = 40
```

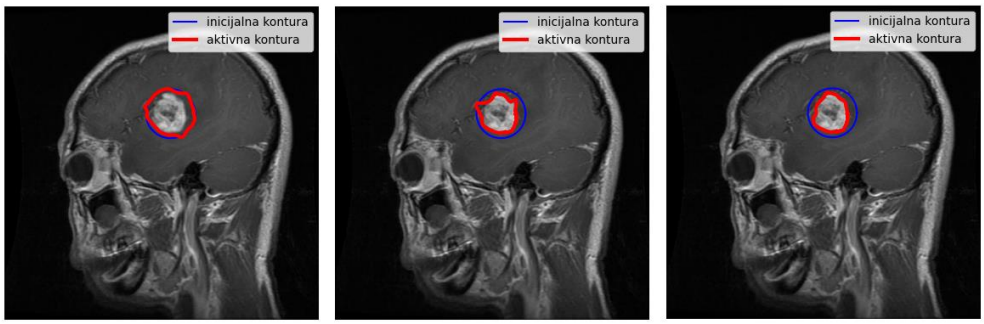
```
x = centar[0] + radijus * np.cos(ugao)
y = centar[1] + radijus * np.sin(ugao)
inicijalna_kontura = np.array([y,x]).T

aktivna_kontura = active_contour(
    gaussian(slika_grayscale, sigma=2),
    inicijalna_kontura,
    alpha=0.01,
    beta=0.3,
    gamma=0.001,
    max_iterations=300,
)

plt.imshow(slika_grayscale, cmap='gray')
plt.plot(inicijalna_kontura[:, 1], inicijalna_kontura[:, 0],
         'b', label="inicijalna kontura")
plt.plot(aktivna_kontura[:, 1], aktivna_kontura[:, 0],
         'r', lw=3, label="aktivna kontura")
plt.legend()
plt.show()
```

B. Rezultat izvršavanja:



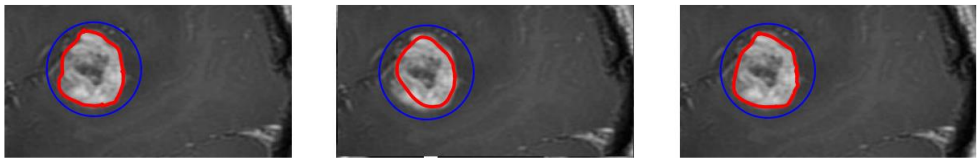
Zadatak za samostalni rad 6.8. Za Pr. 6.8 varirati vrednosti broja iteracija i vrednosti parametara *alpha* i *beta*, pa ispitati efekat ovih varijacija na rezultat segmentacije primenom metode aktivnih kontura.

Rešenje: Na Sl. 6.12. je prikazan rezultat segmentacije metodom aktivnih kontura redom za 5, 50 i 100 iteracija, pri čemu je vizuelno vidljiv progres deformacija aktivne konture počev od inicijalno zadate kružne konture ka ivicama ROI tumora. Na Sl. 6.13 je ilustrovan efekat promene *alpha* parametra (sa njegovim povećanjem kontura postaje kraća) i *beta* parametra (sa njegovim povećanjem kontura postaje glatkija).



broj iteracija = 5 broj iteracija = 50 broj iteracija = 100

Slika 6.12. Poređenje rezutata segmentacije metodom aktivnih kontura kroz iteracije



alpha=0.01, *beta*=0.3 *alpha*=1, *beta*=0.3 *alpha*=0.01, *beta*=3

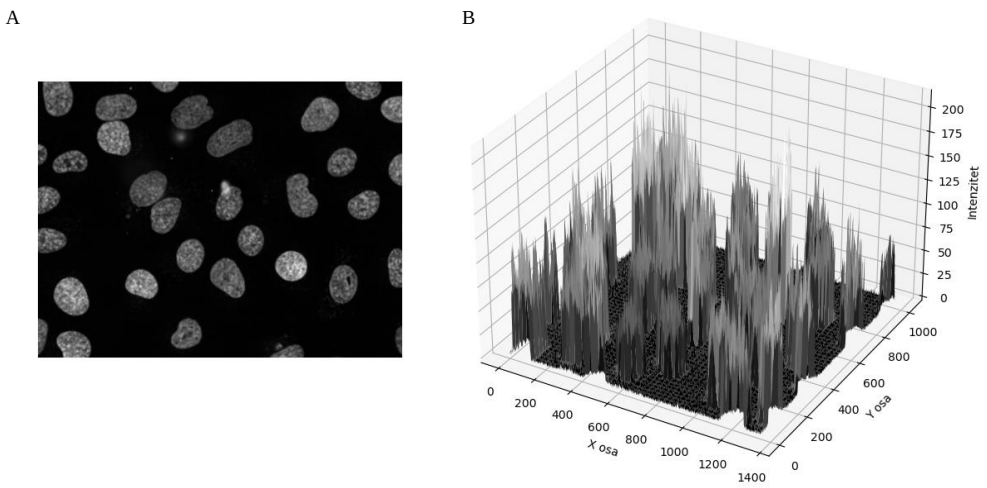
Slika 6.13. Poređenje rezutata segmentacije metodom aktivnih kontura za različite parametre *alpha* i *beta*

6.9. Metoda vododelnica - hibridni pristup segmentacije

Metoda vododelnica (eng. *Watershed*) je metoda segmentacije objekata na osnovu njihove geometrije. Sam naziv potiče od vododelnica koje postoje u prirodi i koje razdvajaju različite slivove, tj. vododelnica je granica na kojoj se određuje ka kom slivu će kap kiše koja na nju padne da se kreće: voda koja padne sa jedne strane vododelnice teče u jedan sliv, a voda koja padne sa druge strane teče u drugi sliv. Vododelnice se obično nalaze na većim visinama. Kod segmentacije slika, vododelnice prestavljaju granice objekata na slici. Koraci metode vododelnica za segmentaciju su sledeći:

- Kreiranje topografskog prikaza slike - slika se posmatra kao funkcija intenziteta piksela gde svaki piksel ima svoju visinu, tj. slika se predstavlja kao teren sa dolinama (pikseli niskog intenziteta) i

uzvišenjima (pikseli visokog intenziteta), Sl. 6.14B. Topološka prezentacija se može formirati direktno na osnovu vrednosti ulazne *grayscale* slike ili na osnovu transformacije rastojanja binarizovane ulazne slike. Transformacija rastojanja se standardno koristi jer pomaže u boljem definisanju granica objekata.



Slika 6.14 A) Originalna slika, B) Topografski prikaz slike pod A)

- Određivanje markera tj. početnih tačaka odakle „vodeni tok“ u slučaju poplave počinje da se širi – markeri su obično mesta lokalnih minimuma („dolina“). Markeri se mogu postaviti manuelno, ili se postavljaju automatski na pozicije centroida ako se koristi transformacija rastojanja.
- Formiranje granice (ivica) – granice između objekata se formiraju na mestima gde se „vodeni tokovi“ od različitih markera susreću.

Metoda vododelnica predstavlja hibridni pristup jer kombinuje metodologiju zasnovanu na rastu regiona (Podpoglavlje 6.7) i metodologiju detekcije ivica (Poglavlje 5): region se širi počev od početnih markera, ali se granice između regija formiranju na mestima sa visokim gradijentom intenziteta. Ova metoda je od posebnog značaja za segmentaciju objekata koji su povezani ili se preklapaju.

Na Pr. 6.9 je prikazan programski kôd za segmentaciju jezgra ćelija na mikroskopskom snimku „dna-48.png“ iz baze slika *2009_ISBI_2DNuclei_code_data*[41] [Coelho et al. 2009].

[41] *2009_ISBI_2DNuclei_code_data* baza mikroskopskih slika jezgra ćelija je dostupna na linku: https://murphylab.web.cmu.edu/data/2009_ISBI_Nuclei.html (poslednji pristup Decembar 2024).

Pr. 6.9. Primer *vododelnice.py* – segmentacija jezgara na mikroskopskim snimcima ćelija metodom vododelnica

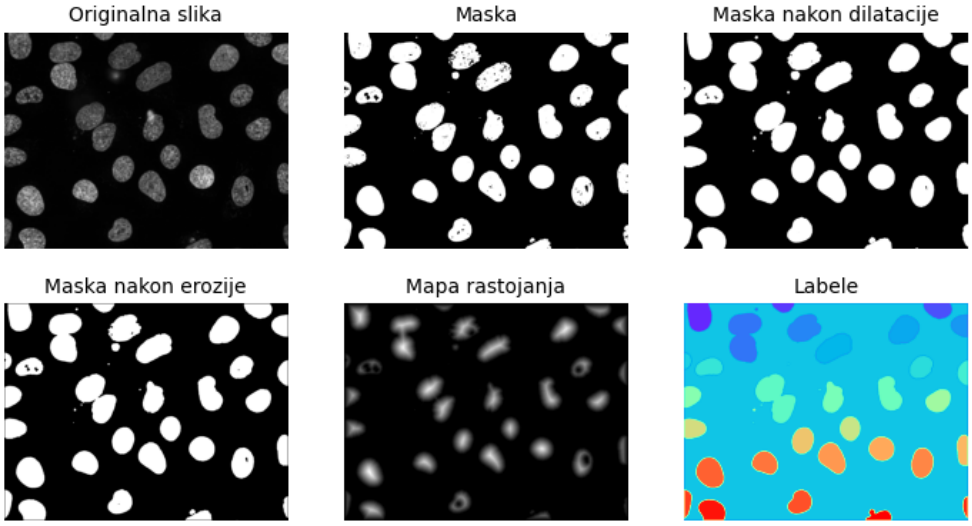
---

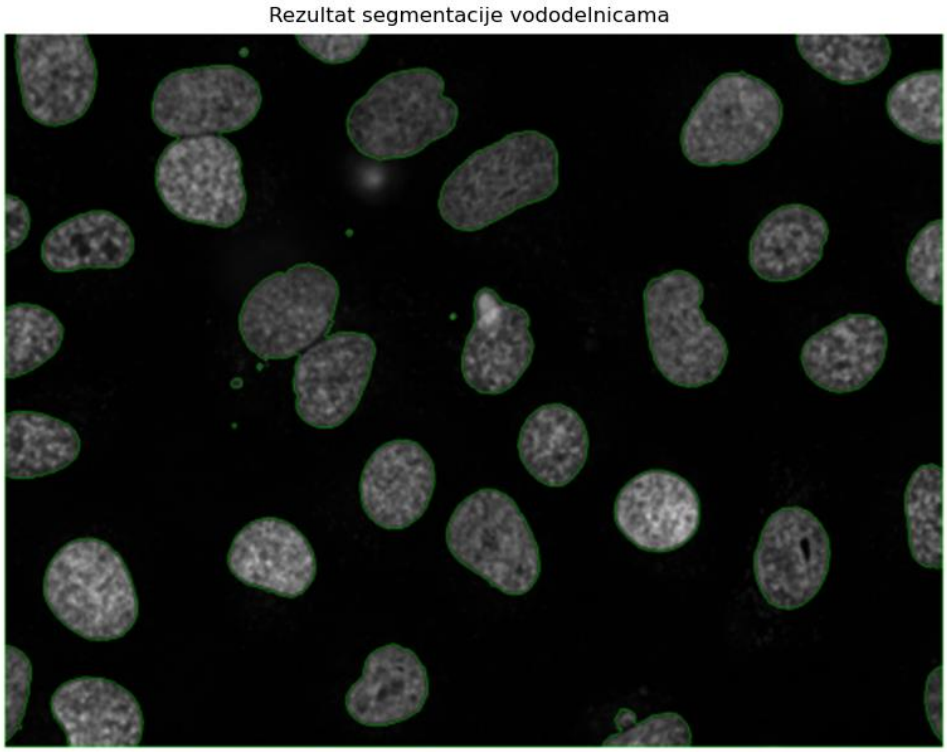
A. Programski kôd:

```
import cv2
from skimage.filters.thresholding import threshold_otsu
import numpy as np
import scipy.ndimage as ndi
import matplotlib.pyplot as plt

plt.close('all')
# Ucitavanja slike
slika_RGB=cv2.imread('dna-48.png')
slika_grayscale=cv2.cvtColor(slika_RGB, cv2.COLOR_BGR2GRAY)

# Primena Otsu metode
prag=threshold_otsu(slika_grayscale)
print('Vrednost praga T=', prag)

# Kreiranje maske na osnovu Otsu praga
maska = np.where(slika_grayscale>prag, 1, 0)

# Primena morfoloških operacija
dil_maska = ndi.binary_dilation(maska, iterations=4)
erod_maska = ndi.binary_erosion(dil_maska, iterations=2)

# Primena transformacije rastojanja
mapa_rastojanja, indeks = ndi.distance_transform_edt(
    erod_maska == 1, return_indices=True)

# Kreiranje maske mape rastojanja
maska_mape = np.where(mapa_rastojanja>1, 1, 0)

# Labeliranje objekata
labele, br_labela = ndi.label(maska_mape)
```

```
# Primena vododelnica
rezultat=cv2.watershed(slika_RGB, labele)

# Označavanje ivica onjekata
slika_RGB[rezultat == -1] = [0, 255, 0]

# Prikaz koraka obrade
outputs = [
    ("Originalna slika", slika_grayscale),
    ("Maska", maska),
    ("Maska nakon dilatacije", dil_maska),
    ("Maska nakon erozije", erod_maska),
    ("Mapa rastojanja", mapa_rastojanja),
    ("Labele", labele)
]
plt.figure(1)
for i, (title, image) in enumerate(outputs, start=1):
    plt.subplot(2, 3, i)
    plt.title(title, fontsize=10)
    if i==6:
        plt.imshow(image, cmap='rainbow')
    else:
        plt.imshow(image, cmap='gray')
    plt.axis('off')
    plt.show()

plt.figure(2)
plt.imshow(cv2.cvtColor(slika_RGB, cv2.COLOR_BGR2RGB))
plt.title('Rezultat segmentacije vododelnicama')
plt.show()
plt.axis('off')
```

B. Rezultat izvršavanja:





U Pr. 6.9 su nakon učitavanja slike i konvertovanja u *grayscale* paletu primenjeni sledeći koraci:

- kreiranje maske primenom Otsu metode (videti Potpoglavlje 6.6.1)

- primena morfoloških operacija dilatacije i erozije na masku (videti Potpoglavlje 6.3)
- primena transformacije rastojanja na dilatiranu i erodiranu masku (videti Potpoglavlje 6.5)
- kreiranje maske mape rastojanja (pikseli iz mape rastojanja koji imaju veću vrednost od praga jednakog 1 će imati vrednost „1“ u novoj masci, a ostali pikseli nove maske će imati vrednost „0“)
- labeliranje objekata (videti Pr. 6.6)
- primena funkcije `cv2.watershed()` koja primenjuje metodu vododelnica na dobijene labele iz prethodnog koraka.

Na Pr. 6.9B je prikazan rezultati svih koraka izvršavanja kôda. Rezultat metode vododelnica su regioni zaokruženi zelenom bojom na originalnoj slici.

6.10. Metrike za evaluaciju segmentacije

Nakon automatskog ili poluautomatskog procesa segmentacije, tačnost rezultata segmentacije se može opisati poređenjem sa zlatnim standardom pomoću metrika za evaluaciju segmentacije. Zlatni standard segmentacije je obično rezultat manuelne segmentacije (vidi Potpoglavlje 6.1). U ovom poglavlju će biti opisane dve takve metrike, *Dice* koeficijent (*D*) i *Hausdorff*-ova udaljenost (*H*).

Dice koeficijent se definiše na sledeći način:

$$D = \frac{2|A \cap B|}{|A| + |B|} \tag{6.16}$$

gde su A i B skupovi piksela za rezultat segmentacije i zlatni standard segmentacije, respektivno. Opseg vrednosti *Dice* koeficijenta je $D \in [0,1]$.

Hausdorff-ova udaljenost predstavlja maksimalnu udaljenost između bilo koje tačke u rezultatu segmentacije A i najbliže tačke zlatnog standarda B, tj. definiše se na sledeći način:

$$H = \max\left(H_{forward}, H_{backward}\right) \tag{6.17}$$

$$H_{forward} = \sup_{a \in A} \inf_{b \in B} d(a,b) \tag{6.18}$$

$$H_{backward} = \sup_{b \in B} \inf_{a \in A} d(b,a) \tag{6.19}$$

gde je $d(a,b)$ rastojanje između tačke a koja pripada rezultatu segmentacija A i tačke b koja pripada zlatnom standardu segmentacije B. Da bi se izračunalo H, najpre se za svako a računa minimalno rastojanje do B, a potom se za ceo skup A traži maksimum takvih rastojanja ($H_{forward}$). Potom se za svako b računa minimalno rastojanje do A, a potom se za ceo skup B traži maksimum

takvih rastojanja ($H_{backward}$). Konačno, H se određuje kao maksimum od $H_{forward}$ i $H_{backward}$.

Na Pr. 6.10 je prikazana implementacija relacije (6.16) za proračun *Dice* koeficijenta i relacija (6.17)-(6.19) za proračun *Hausdorff*-ove udaljenosti. Ovim programskim kôdom treba dopuniti Pr. 6.7 kako bi se odredila sličnost rezultata segmentacije metodom rasta regiona i zlatnog standarda segmentacije (datog u datoteci „abide_cc.nii“ u okviru *ABIDE* baze).

Pr. 6.10. Primer *metoda_rasta_regiona_metrike.py* – dopuna Pr. 6.7 metrikama za evaluaciju rezultata segmentacije

---

A. Programski kôd (dopuna na Pr. 6.7):

```
# Učitavanje zlatnog standarda
zlatni = sitk.ReadImage("abide_cc.nii", sitk.sitkFloat32)

# Konvertovanje zlatnog standarda u matricu
zlatni_standard = sitk.GetArrayViewFromImage(zlatni)
zlatni_standard = zlatni_standard[slajs_indeks,:,:]

import numpy as np
def dice_coefficient(segmentation1, segmentation2):
    # Računanje Dice koeficijenta
    return (2. * np.sum(segmentation1*segmentation2)/
            (np.sum(segmentation1) + np.sum(segmentation2)))

D=dice_coefficient(segmentacija_matrica, zlatni_standard)
print(f"Dice koeficijent: {D}")

from scipy.spatial.distance import cdist
def hausdorff_udaljenost(segmentation1, segmentation2):
    # Računanje rastojanja d
    d = cdist(segmentation1, segmentation2,
              metric='euclidean')
    H_forward = np.max(np.min(d, axis=1))
    H_backward = np.max(np.min(d, axis=0))
```

```
    return max(H_forward, H_backward)

H=hausdorff_udaljenost(segmentacija_matrica, zlatni_standard)
print(f"Hausdorff-ova udaljenost: {H}")
```

B. Rezultat izvršavanja:

```
Dice koeficijent: 0.9666857034075766
Hausdorff-ova udaljenost: 3.1622776601683795
```

6.11. Rezime poglavlja

- Segmentacija omogućava izdvajanje regiona od interesa slike (ROI) na osnovu kriterijuma sličnosti i/ili povezanosti susednih piksela.
- Metode segmentacije se dele na metode zasnovane na statistici histograma, detekciji ivica, na rastu regiona, aktivnim konturama, klasterovanju, dubokom učenju i grafovima. Hibridni pristupi mogu kombinovati barem dva od navedenih pristupa segmentacije.
- Manuelnu (ručnu) segmentaciju karakterišu intervarijabilnost i intravarijabilnost eksperta.
- Rezultat maskiranja su binarne slike na kojima pikseli sa vrednošću „True“ predstavljaju kandidate za dalju analizu.
- Morfološke operacije su: erozija, dilatacija, otvaranje i zatvaranje. Erozija se koristi za sužavanje objekta, a dilatacija za proširivanje objekta. Otvaranje se primenjuje za otklanjanje malih objekata, a zatvaranje za popunjavanje rupica u objektima.
- Hafova transformacija koristi parametarski prostor umesto Dekartovog koordinatnog sistema i primenjuje se za detekciju geometrijskih oblika.
- Transformacija rastojanja izračunava rastojanje svakog belog piksela binarne slike do najbližeg pozadinskog piksela. Jedna od najvažnijih primena prostorne transformacije je u metodi segmentacije vododelnicama.
- Otsu metoda je standardna metoda za segmentaciju slike sa bimodalnim histogramom na osnovu globalnog praga koji maksimizuje interklasnu varijansu.

- Metoda rasta regiona za segmentaciju se bazira na pretpostavljanju početnog semena (tačke unutar regiona) i širenju semena prema kriterijumima sličnosti i/ili povezanosti.
- Metoda aktivnih kontura za segmentaciju se zasniva na pretpostavljanju početne konture (oko regiona ili unutar regiona za segmentaciju) i deformacijama početne konture pod dejstvom spoljašnjih i unutrašnjih sila radi optimizacije energije konture.
- Metoda vododelnica za segmentaciju se bazira na topografiji intenziteta piksela slike ili njene transformacione mape rastojanja, a sa ciljem nalaženja granica između objekata kao tačaka gde se nalaze „vododelnice“.
- *Dice* koeficijent i *Hausdorff*-ova udaljenost su metrike za evaluaciju rezultata segmentacije koje omogućavaju određivanje sličnosti rezultata segmentacije za zlatnim standardom.

7. STATISTIČKA ANALIZA TEKSTURE

Tekstura predstavlja osobinu slike koja opisuje prostornu raspodelu intenziteta piksela u regionu slike od interesa ili na celoj slici. Fina tekstura odgovara malim promenama u intenzitetu piksela, a gruba tekstura odgovara većim promenama u intenzitetu piksela. Na osnovu teksture se mogu detektovati abnormalnosti koje mogu biti nevidljive golim okom. Osnovna podela teksturalnih obeležja je na: statistička, strukturna i spektralna (transformaciona) obeležja. Primeri strukturnih (ivice) i spektralnih obeležja (Furijeovi koeficijenti) su razmatrana u Poglavlju 5, a u nastavku ovog poglavlja će biti objašnjeni primeri statističkih obeležja, tj. obeležja koja se izdvajaju na osnovu statističke analize intenziteta piksela. Dalja primena ovih obeležja je pre svega u algoritmima mašinskog učenja što će biti objašnjeno u Poglavlju 10.

7.1.1. Statistička obeležja prvog reda

Statistička obeležja prvog reda su:

- srednja vrednost μ:

$$\mu = \frac{1}{m \cdot n} \sum_{x=0}^{m-1} \sum_{y=0}^{n-1} \rho_{xy} \tag{7.1}$$

- varijansa σ^2:

$$\sigma^2 = \frac{1}{m \cdot n} \sum_{x=0}^{m-1} \sum_{y=0}^{n-1} (\rho_{xy} - \mu)^2 \tag{7.2}$$

- standardna devijacija σ:

$$\sigma = \sqrt{\frac{1}{m \cdot n} \sum_{x=0}^{m-1} \sum_{y=0}^{n-1} (\rho_{xy} - \mu)^2} \tag{7.3}$$

- entropija H (mera složenosti i nepredvidljivosti):

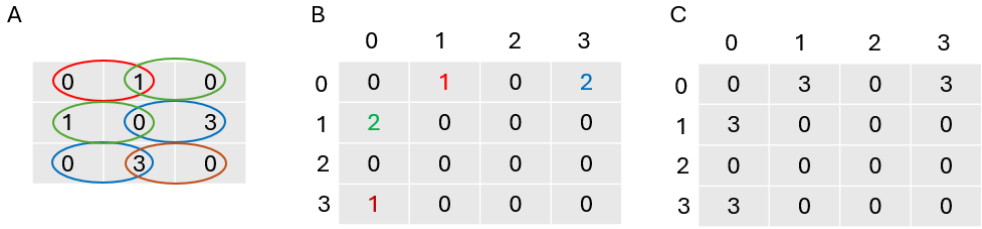
$$H = -\sum_{x=0}^{m-1} \sum_{y=0}^{n-1} p(\rho_{xy}) log_2(p(\rho_{xy})) \tag{7.4}$$

gde je ρ_{xy} intenzitet piksela na poziciji (x,y) u slici I dimenzije m x n, a $p(\rho_{xy})$ verovatnoća pojavljivanja piksela intenziteta ρ_{xy} (odnos broja piksela koji ima intenzitet ρ_{xy} prema ukupnom broju piksela slike $m \cdot n$).

7.1.2. Statistička obeležja drugog reda

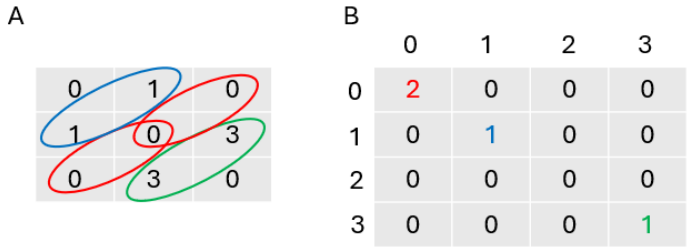
Statistička obeležja drugog reda su obeležja koja se izračunavaju na osnovu matrice ko-pojavljivanja nivoa sivog (eng. *Gray Level Co-occurrence*

Matrix, *GLCM*). Za sliku koja ima L nivoa sivog, *GLCM* matrica ima dimenzije L x L i pokazuje koliko često se piksel intenziteta i nalazi zajedno sa pikselom intenziteta j pod uglom δ i na rastojanju d. Ugao može biti 0°, 45°, 90° ili 135° stepeni, a rastojanje se meri u pikselima (najčešće je rastojanje d=1). Na Sl. 7.1A je dat primer slike dimenzije m x n = 3 x 3 sa L=4 nivoa sivog, a na 7.1B je data njena *GLCM* matrica dimenzije L x L= 4 x 4 za ugao δ=0° i rastojanje d=1.



Slika 7.1. A) Primer slike 3 x 3, B) *GLCM* matrica za ugao δ=0° i rastojanje d=1, C) Simetrična *GLCM* matrica za ugao δ=0° i rastojanje d=1

Na Sl. 7.1C je data simetrična *GLCM* matrica dimenzije L x L= 4 x 4 za ugao δ=0° i rastojanje d=1 u kojoj se parovi intenziteta piksela (i,j) i (j,i) smatraju istim i broje se zajedno. Na Sl. 7.2A su identifikovani parovi intenziteta piksela za ugao δ =45° i rastojanje d=1, a na Sl. 7.2B je prikazana odgovarajuća *GLCM* matrica.



Slika 7.2. A) Primer slike 3 x 3, B) *GLCM* matrica za ugao δ=45° i rastojanje d=1

Na Pr. 7.1 je data implementacija koja omogućava proračun *GLCM* matrica za primer prikazan na Sl. 7.2. Normalizovana *GLCM* matrica se dobija deljenjem svakog elementa *GLCM* matrice zbirom svih njenih elemenata.

Pr. 7.1. Primer *GLCM_matrica.py* – proračun *GLCM* matrice

A. Programski kôd:

```
import numpy as np
from skimage.feature import greycomatrix
```

```
# Kreiranje 3x3 matrice sa 4 nivoa sivog
matrix=np.array([
    [0,1,0],
    [1,0,3],
    [0,3,0]
    ])

# Definisanje parametara (za rastojanje i za ugao) za GLCM
distances = [1, 2]    # rastojanja
angles = [0, np.pi/4, np.pi/2, 3*np.pi/4]   # uglovi
levels = 4          # broj nivoa sivog

# Izračunavanje GLCM matrice
glcm = greycomatrix(matrix, distances=distances,
                    angles=angles, levels=levels,
                    symmetric=False, normed=False)
glcm_norm = greycomatrix(matrix, distances=distances,
                         angles=angles, levels=levels,
                         symmetric=False, normed=True)
glcm_sym = greycomatrix(matrix, distances=distances,
                        angles=angles, levels=levels,
                        symmetric=True, normed=False)

distance_i = distances.index(1)  # indeks rastojanja 1
angle_i = angles.index(0)        # indeks ugla 0

# GLCM matrica za rastojanje 1 piksel i ugao 0
glcm_1_0 = glcm[:, :, distance_i, angle_i]
glcm_1_0_norm = glcm_norm[:, :, distance_i, angle_i]
glcm_1_0_sym = glcm_sym[:, :, distance_i, angle_i]

print('GLCM matrica za rastojanje 1, ugao 0deg:')
print(glcm_1_0)
print('Normalizovana GLCM matrica:')
```

```
print(glcm_1_0_norm)
print('Simetricna GLCM matrica:')
print(glcm_1_0_sym)
```

B. Rezultat izvršavanja:

```
GLCM matrica za rastojanje 1, ugao 0deg:
[[0 1 0 2]
 [2 0 0 0]
 [0 0 0 0]
 [1 0 0 0]]
Normalizovana GLCM matrica za rastojanje 1, ugao 0deg:
[[0.         0.16666667 0.         0.33333333]
 [0.33333333 0.         0.         0.        ]
 [0.         0.         0.         0.        ]
 [0.16666667 0.         0.         0.        ]]
Simetricna GLCM matrica za rastojanje 1, ugao 0deg:
[[0 3 0 3]
 [3 0 0 0]
 [0 0 0 0]
 [3 0 0 0]]
```

GLCM obeležja (statistička obeležja drugog reda koja se izračunavaju na osnovu *GLCM* matrice slike) su:

- kontrast *C*:

$$C = \sum_{i=0}^{L-1} \sum_{j=0}^{L-1} p_{ij}(i-j)^2 \tag{7.5}$$

- nesličnost (eng. *dissimilarity*) *D*:

$$D = \sum_{i=0}^{L-1} \sum_{j=0}^{L-1} p_{ij}|i-j| \tag{7.6}$$

- homogenost *h*:

$$h = \sum_{i=0}^{L-1} \sum_{j=0}^{L-1} \frac{p_{ij}}{1+(i-j)^2} \tag{7.7}$$

- korelacija *Corr*:

$$Corr = \frac{\sum_{i=0}^{L-1} \sum_{j=0}^{L-1} p_{ij}(i-\mu_i)(j-\mu_j)}{\sigma_i \sigma_j} \tag{7.8}$$

- ugaoni moment drugog reda (eng. *Angular Second Moment*, *ASM*), *ASM*:

$$ASM = \sum_{i=0}^{L-1} \sum_{j=0}^{L-1} {p_{ij}}^2 \tag{7.9}$$

- energija *E*:

$$E = \sqrt{ASM} \tag{7.10}$$

- entropija iz *GLCM*, H_{GLCM}:

$$H_{GLCM} = -\sum_{i=0}^{L-1}\sum_{j=0}^{L-1} p_{ij} log_2 p_{ij} \quad (7.11)$$

gde su *i, j* indeksi u *GLCM* matrici slike koji predstavljaju intenzitete piksela slike; μ_i, μ_j srednje vrednosti *i*-tog reda i *j*-te kolone *GLCM* matrice; σ_i, σ_j standardne devijacije *i*-tog reda i *j*-te kolone *GLCM* matrice; *L* broj nivoa sivog na slici; p_{ij} verovatnoća da se (*i*,*j*) pojave zajedno u slici (tj. da se piksel intenziteta *i* i piksel intenziteta *j* nađu zajedno u slici pod uglom δ i na rastojanju *d*). Verovatnoća p_{ij} se dobija kao odnos g_{ij} elementa *GLCM* matrice i sume svih elemenata *GLCM* matrice:

$$p_{ij} = \frac{g_{ij}}{\sum_{i=0}^{L-1}\sum_{j=0}^{L-1} g_{ij}}. \quad (7.12)$$

Primer primene *GLCM* obeležja u biomedicinskom slikanju je dat u okviru Poglavlja 10 (Radiomika), Potpoglavlje 10.3.3. (Primer primene tradicionalne radiomike) gde je ilustrovano kako se na osnovu *GLCM* obeležja limfoblasta i limfocita može formirati sistem za prepoznavanje limfoblasta baziran na mašinskom učenju.

7.1.3. Lokalni binarni obrazac

Lokalni binarni obrazac (eng. *Local Binary Pattern*, *LBP*) je primer obeležja koje se kategoriše kao strukturno i statističko obeležje, jep sa jedne strane analizira lokalne obrasce strukture u slici (tj. prelaze intenziteta između centralnog piksela i njegovih suseda), a sa druge strane *LBP* obeležje je obično vektor koji sadrži histograme lokalnih obrazaca.

Koraci računanja *LBP* obeležja su:

1. Podela slike na podslike tj. ćelije (npr. dimenzija svake podslike je 16 x 16 piksela).
2. Za svaki piksel intenziteta ρ_{xy} na poziciji (*x*,*y*) na podslici se proračunava *LBP* vrednost prema sledećoj formuli:

$$LBP = \sum_{i=0}^{broj\_suseda-1} s(\rho_i - \rho_{xy}) \cdot 2^i \quad (7.13)$$

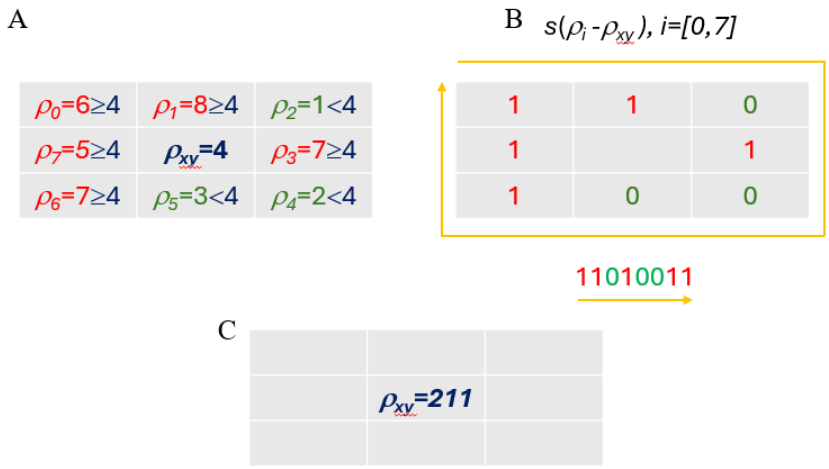
gde je ρ_i intenzitet piksela koji su susedni u odnosu na centralni (*x*,*y*) piksel (nalaze se na radijusu *r* u odnosu na centralni piksel i ima ih 8·*r*), a funkcija *s* je definisana na sledeći način:

$$s(k) = \begin{cases} 1, & k \geq 0 \\ 0, & k < 0 \end{cases} \quad (7.14)$$

Primer susedstva za radijus *r*=1 i proračun *LBP* vrednosti centralnog piksela je dat na Sl. 7.3. Broj suseda centralnog piksela (*x*,*y*) je u ovom slučaju 8·*r*=8·1=8, a njihovi intenziteti su ρ_i, $i \in [0,7]$. Intenzitet ρ_i svakog susednog piksela se poredi sa ρ_{xy}, i ako je manji od njega odgovarajuća

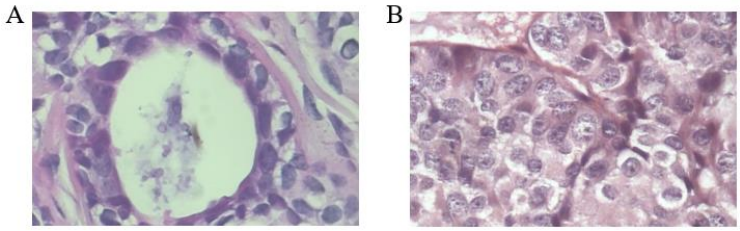
vrednost funkcije $s(\rho_i - \rho_{xy})$ će biti 0, u suprotnom će biti 1, Sl. 7.3B. Konačno, vrednost *LBP*=211 dobijena relacijama (7.13) i (7.14) se dodeljuje centralnom pikselu (x,y), Sl. 7.3C. Procedura se primenjuje dalje na sve piksele svake od podslika.

3. Za svaku podsliku se određuje normalizovani histogram *LBP* vrednosti izračunatih u prethodnom koraku.
4. Spajaju se u jedan *LBP* vektor obeležja histogrami svih podslika.



Slika 7.3. Primer proračuna *LBP* vrednosti: A) podslika dimenzije 3 x 3, B) proračun vrednosti *s* funkcije, C) *LBP* vrednost se dodeljuje centralnom pikselu

Na Pr. 7.2A je prikazan programski kôd koji na histopatološkom mikroskopskom snimku „SOB_M_MC-14-13418DE-400-002.png“ malignog tumora dojke iz baze BreaKHis[42][Spanhol et al. 2016] određuje *LBP* vektor obeležja. Na Sl. 7.4A i Sl. 7.4.B su dati primeri benignog i malignog snimka BreaKHis baze.



Slika 7.4. Primeri histopatoloških mikroskopskih snimaka tumora dojke iz BreaKHis baze: A) benigni tumor „SOB_B_A-14-22549AB-400-006.png“, B) maligni tumor „SOB_M_MC-14-13418DE-400-002.png“

[42] Baza podataka je dostupna na sledećem linku (poslednji pregled Decembar 2024.): https://www.kaggle.com/datasets/forderation/breakhis-400x pod licencom CCO: Public Domain (https://creativecommons.org/publicdomain/zero/1.0/)

Nakon učitavanja snimka i konverzije u *grayscale* sliku, definisana je dimenzija svake podslike (`podslika_dim=16`, tj. dimenzija podslike je 16 x 16) za koju će se određivati *LBP* vrednosti prema relacijama (7.13) i (7.14). Rastojanje do suseda koje će se uzimati u obzir pri proračunu *LBP* vrednosti je `radijus=1` što daje broj suseda jednak 8. Za svaku podsliku se primenjuje funkcija `local_binary_pattern()` iz *skimage* biblioteke u kojoj je implementirano računanje *LBP* vrednosti (`default` metod proračuna je prema gore opisanim koracima). Potom se za svaku podsliku kreira normalizovani histogram sa brojem binova jednakim maksimumu *LBP* vrednosti podslike uvećanoj za 1 (ako je maksimalna *LBP* vrednost 255, onda je broj binova histograma 256). Normalizovani histogram *LBP* vrednosti svake podslike se dodaje u LBP vektor obeležja. Na Pr. 7.2B je prikazan rezultat izvršavanja programskog kôda. Dimenzija učitanog histopatološkog snimka je 460 x 700, pa je broj podslika za koje se računaju normalizovani histogrami *LBP* vrednosti 28 x 43 = 1204 (za dimenziju podslike 16 x 16), a dimenzija formiranog *LBP* vektora obeležja je 1204 x 256 = 308224 (dimenzija normalizovanog histograma *LBP* vrednosti za svaku podsliku je 256).

Pr. 7.2. Primer *LBP.py* – proračun *LBP* vektora obeležja

A. Programski kôd:

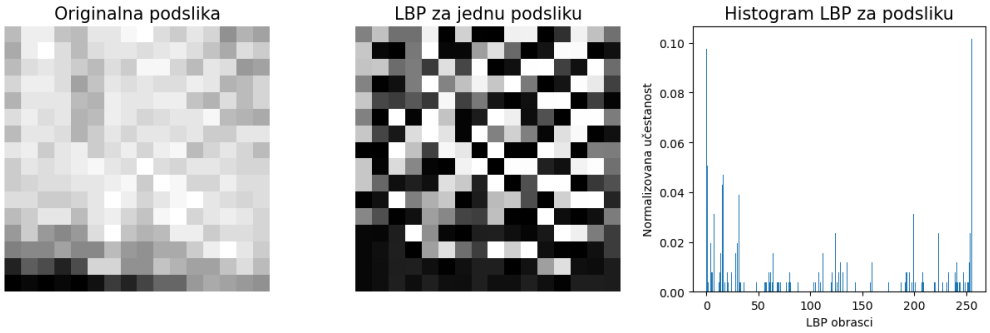
```
import cv2
import numpy as np
from skimage.feature import local_binary_pattern
import matplotlib.pyplot as plt

plt.close("all")

slika = cv2.imread('SOB_M_MC-14-13418DE-400-002.png')
slika_grayscale = cv2.cvtColor(slika, cv2.COLOR_RGB2GRAY)

# Dimenzije podslike i broj podslika
podslika_dim = 16
height, width = slika_grayscale.shape
broj_podslika_x = width // podslika_dim
broj_podslika_y = height // podslika_dim
```

```
# Parametri susedstva
radijus = 1  # Radijus susedstva
broj_suseda = 8 * radijus  # Broj suseda

vektor_obelezja = []

# Prolazak kroz podslike
for i in range(broj_podslika_y):
    for j in range(broj_podslika_x):
        # Izrezivanje trenutne podslike
        podslika = slika_grayscale[i * podslika_dim:(i + 1) *
                                   podslika_dim, j *
                                   podslika_dim:(j + 1) *
                                   podslika_dim]

        # Izračunavanje LBP za podsliku
        lbp_podslika = local_binary_pattern(podslika,
                                            broj_suseda,
                                            radijus,
                                            method='default')
        broj_bin=int(lbp_podslika.max() + 1)

        # Kreiranje histograma za podsliku
        hist, bin_granice = np.histogram(lbp_podslika.ravel(),
                                bins=broj_bin, range=(0, 256))

        # Normalizacija histograma
        hist = hist.astype("float")
        hist /= hist.sum()

        # Dodavanje histograma u LBP vektor obeležja
        vektor_obelezja.extend(hist)
```

```
# Prikaz dimenzije vektora obeležja
print("Dimenzija vektora obeležja:", len(vektor_obelezja))

# Prikaz jedne podslike i njenog histograma
primer_podslike = slika_grayscale[0:podslika_dim,
                                  0:podslika_dim]
primer_lbp = local_binary_pattern(primer_podslike,
                                  broj_suseda,
                                  radijus,
                                  method='default')
primer_broj_bin=int(primer_lbp.max() + 1)
primer_hist, bin_granice = np.histogram(
    primer_lbp.ravel(), bins=primer_broj_bin, range=(0, 256))
primer_hist = primer_hist.astype("float") / primer_hist.sum()

plt.subplot(1, 3, 1)
plt.title("Originalna podslika", fontsize=15)
plt.imshow(primer_podslike, cmap="gray")
plt.axis("off")

plt.subplot(1, 3, 2)
plt.title("LBP za jednu podsliku", fontsize=15)
plt.imshow(primer_lbp, cmap="gray")
plt.axis("off")

plt.subplot(1, 3, 3)
plt.title("Histogram LBP za podsliku", fontsize=15)
plt.bar(bin_granice[:-1], primer_hist)
plt.xlabel("LBP obrasci")
plt.ylabel("Normalizovana učestanost")
plt.show()
```

B. Rezultat izvršavanja:

```
Dimenzija vektora obeležja: 308224
```



Uniformnost *LBP*-a. Dimenzija *LBP* vektora obeležja se može redukovati računanjem uniformnosti *U* tj. broja prelaza „0 u 1“ ili „1 u 0“ svake *LBP* vrednosti. *LBP* vrednost se smatra uniformnom ako je broj prelaza manji ili jednak 2 ($U \leq 2$), u suprotnom nije uniformna. Sve neuniformne *LBP* vrednosti se grupišu u poseban bin za neuniformne obrasce prilikom određivanja histograma, čime je moguće značajno redukovati dimenziju vektora obeležja. Na primer, ako je *LBP* vrednost u binarnom obliku kao na Sl. 7.3 11010011, broj prelaza „0 u 1“ ili „1 u 0“ je $U=4$ i obrazac spada u neuniformne obrasce ($U>2$). Ako je na primer *LBP* vrednost u binarnom obliku 11100000, može se uočiti samo jedan prelaz „1 u 0“, tj. obrazac je uniforman ($U \leq 2$).

Zadatak za samostalni rad 7.1. Modifikovati Pr. 7.2 tako da se za proračun *LBP* vrednosti podslika koristi uniformni metod (`method='uniform'` za `local_binary_pattern()`). Kolika će sada biti dimenzija *LBP* vektora obeležja?

7.2. Rezime poglavlja

- Statistička obeležja se izdvajaju na osnovu statističke analize intenziteta piksela.
- Statistička obeležja prvog reda su npr. srednja vrednost, varijansa, standardna devijacija, entropija
- *GLCM obeležja* (npr. kontrast, nesličnost, homogenost, korelacija, ugaoni moment drugog reda, energija, entropija) su statistička obeležja drugog reda.

- Lokalni binarni obrasci predstavljaju kombinovano, strukturno i statističko obeležje koje se bazira na praćenju lokalnih promena intenziteta i na histogramu lokalnih obrazaca.
- Pristup uniformnosti (određivanje broja prelaza između binarnih vrednosti 0 i 1 i grupisanje u poseban bin svih neuniformnih obrazaca) može značajno redukovati dužinu *LBP* vektora obeležja.
- Statistička teksturalna obeležja određena za region slike ili za celu sliku imaju dalju primenu u metodama mašinskog učenja sa ciljem klasifikacije ili prepoznavanja karakterističnih promena.

8. REGISTRACIJA

Registracija biomedicinskih slika je proces „poravnavanja“ dve ili više slika/volumena snimljenih na različitim modalitetima biomedicinskog slikanja (istovremeno ili u različitim vremenskim trenucima) ili istim modalitetom biomedicinskog slikanja u različitim vremenskim trenucima. Registracija je moguća za slike/volumene istog pacijenta, ali i za slike/volumene različitih pacijenata (npr. radi poređenja informacija jedne grupe pacijenata u odnosu na drugu grupu pacijenata). Često se registracija vrši za slike/volumene nastale modalitetom slikanja koji nosi funkcionalnu informaciju (npr. *PET* snimanje) i strukturnu informaciju (npr. *CT* ili *MRI*) gde se potom „poravnate“ (registrovane) slike/volumeni koriste za tzv. fuzioni prikaz. Fuzioni prikaz omogućava ili odvojeno posmatranje registrovanih snimaka, Sl. 8.1A,B ili posmatranje preklopljenih registrovanih snimaka u različitim paletama boja (npr. snimak sa jednog modaliteta se prikaže u *grayscale* paleti boja, a snimak sa drugog modaliteta se prikaže u *rainbow* ili *hot* paleti boja), Sl. 8.1C.

Slika 8.1 Snimak karcinoma desne dojke načinjen hibridnim *PET/CT* modalitetom: A) *PET* snimak, B) *CT* snimak, C) fuzioni preklopljeni *PET* i *CT* snimak prikazani u različitim paletama boja. Slika je preuzeta i modifikovana iz [Ming et al. 2020]. Slika je podeljena pod CC BY 4.0 licencom (https://creativecommons.org/licenses/by/4.0/), poslednji pregled Decembar 2024

U procesu registracije, jedan snimak se postavlja kao fiksan, a drugi snimak koji se registruje se postavlja kao pokretni. Cilj registracije je da se pronađe transformaciona funkcija koja pomera piksele/voksele pokretnog snimka tako da se oni „usklade“ sa pikselima fiksnog snimka, tj. da njihova nova pozicija odgovara istim fizičkim delovima biološkog sistema na fiksnom snimku. Koraci registracije obuhvataju:

- traženje obeležja za uparivanje
- uparivanje obeležja
- proračun transformacione funkcije

- primenu transformacione funkcije na svaki piksel/voksel.

Traženje obeležja za uparivanje. Obeležja za uparivanje mogu biti: pozicije i karakteristike invazivno postavljenih markera (npr. markeri implantirani u kost), pozicije i karakteristike neinvazivno postavljenih markera (npr. markeri postavljeni na kožu pacijenta), anatomski markeri i vrednosti piksela/voksela ili izvedene veličine (npr. obeležja teksture, videti Poglavlje 7).

Metode registracije za koje je neophodno invazivno ili neinvazivno postaviti markere pre snimanja nazivaju se prospektivne metode, a metode registracije za koje se ne postavljaju markeri nazivaju se retrospektivne metode (posmatranje anatomskih markera ili vrednosti intenziteta/obeležja teksture).

Uparivanje obeležja. U zavisnosti od izbora obeležja, načini uparivanja se dele na:

- Metode zasnovane na uparivanju tačaka (eng. *Point-based*) - primenjuju se na invazivno/neinvazivno postavljene markere. Primer ovakve metode je opisan u Potpoglavlju 8.2.
- Metode zasnovane na uparivanju površina (eng. *Surface-based*) - primenjuju se na anatomske markere. Kompleksnost ovakvih algoritama prevazilazi nivo ovog udžbenika, preporučena literatura za detaljnije upoznavanje sa primerom implementacije ovakve metode je [Besl and McKay 1992].
- Metode zasnovane na intenzitetima/teksturi (eng. *Intensity/ Texture-based*) - primenjuju se metrike sličnosti i traži se transformacija za koju je sličnost između snimaka maksimalna. Primer ovakve metode je opisan u Potpoglavlju 8.3.

Proračun transformacione funkcije. Postoje tri tipa transformacionih funkcija koje se mogu izračunavati za potrebe registracije:

- rigidna transformacija koja kombinuje samo translaciju i rotaciju (pogledati Potpoglavlje 8.1)
- afina transformacija koja kombinuje translaciju, rotaciju, skaliranje i smicanje (pogledati Potpoglavlje 8.1)
- nerigidna transformacija tj. nelinearna transformacija koja koristi deformaciju oblika - kompleksnost ovakvih algoritama prevazilazi nivo ovog udžbenika, preporučena literatura za detaljnije upoznavanje sa ovakvim transformacijama je [Crum et al. 2004]

Za metode zasnovane na uparivanju tačaka se transformaciona funkcija računa na osnovu kriterijuma minimalnog rastojanja korespondentnih tačaka na snimcima. Za metode zasnovane na uparivanju površina i metode zasnovane

na intenzitetima/teksturi je za proračun transformacione funkcije neophodno primeniti iterativni proces.

8.1. Afina transformacije

Afina transformacije su geometrijske transformacije koje se primenjuju na svaki piksel sa koordinatama (x,y) tako što se odgovarajući vektor homogenih koordinata [x y 1] množi sa transformacionom matricom T:

$$|x_{trans} \quad y_{trans} \quad 1| = |x \quad y \quad 1| \cdot T \tag{8.1}$$

gde su x_{trans} i y_{trans} transformisane koordinate.

Nakon afina transformacija se održava kolinearnost, paralelizam, konveksnost i odnos dužina paralelnih linija. Za metode registracije se koriste sledeće afina transformacije (i njihove kombinacije): translacija, rotacija, skaliranje i smicanje.

Translacija se definiše na sledeći način:

$$|x_{trans} \quad y_{trans} \quad 1| = |x \quad y \quad 1| \cdot T_{translacija} \tag{8.2}$$

$$T_{translacija} = \begin{vmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ t_x & t_y & 1 \end{vmatrix} \tag{8.3}$$

$$|x_{trans} \quad y_{trans} \quad 1| = |x + t_x \quad y + t_y \quad 1| \tag{8.4}$$

gde su t_x i t_y pozitivni ili negativni pomeraji duž x i duž y ose, respektivno.

Rotacija se definiše na sledeći način:

$$|x_{trans} \quad y_{trans} \quad 1| = |x \quad y \quad 1| \cdot T_{rotacija} \tag{8.5}$$

Na Sl. 8.2 je prikazana rotacija (x,y) tačke za ugao θ oko koordinatnog početka nakon čega se dobija koordinata (x_{trans},y_{trans}). Za odgovarajuće polarne koordinate za (x,y) i za (x_{trans},y_{trans}) važi:

$$x = \rho \cos\phi \tag{8.6}$$

$$y = \rho \sin\phi \tag{8.7}$$

$$x_{trans} = \rho \cos(\phi + \theta) = \rho \cos\phi \cos\theta - \rho \sin\phi \sin\theta \tag{8.8}$$

$$y_{trans} = \rho \sin(\phi + \theta) = \rho \cos\phi \sin\theta + \rho \sin\phi \cos\theta \tag{8.9}$$

odakle se dobija veza između (x,y) i (x_{trans},y_{trans}):

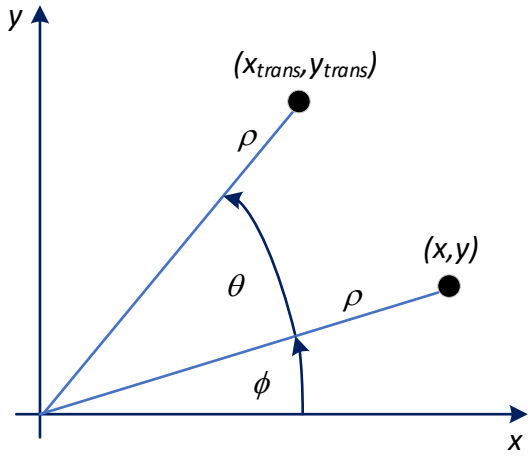
$$x_{trans} = x \cos\theta - y \sin\theta \tag{8.10}$$

$$y_{trans} = x \sin\theta + y \cos\theta \tag{8.11}$$

tj. rotaciona matrica transformacije je:

$$T_{rotacija} = \begin{vmatrix} cos\theta & sin\theta & 0 \\ -sin\theta & cos\theta & 0 \\ 0 & 0 & 1 \end{vmatrix} \quad (8.12)$$

$$|x_{trans} \quad y_{trans} \quad 1| = |xcos\theta - ysin\theta \quad xsin\theta + ycos\theta \quad 1| \quad (8.13)$$



Slika 8.2 Ilustracija rotacije tačke (*x*,*y*) za ugao θ

Skaliranje se definiše na sledeći način:

$$|x_{trans} \quad y_{trans} \quad 1| = |x \quad y \quad 1| \cdot T_{skaliranje} \quad (8.14)$$

$$T_{skaliranje} = \begin{vmatrix} k_x & 0 & 0 \\ 0 & k_y & 0 \\ 0 & 0 & 1 \end{vmatrix} \quad (8.15)$$

$$|x_{trans} \quad y_{trans} \quad 1| = |x \cdot k_x \quad y \cdot k_y \quad 1| \quad (8.16)$$

gde su k_x i k_y faktori skaliranja duž *x* i duž *y* ose, respektivno (ako su faktori skaliranja veći od 1 transformacija uvećava sliku, a ako su faktori manji od 1 transformacija smanjuje sliku). U slučaju transformacije skaliranja neophodno je primeniti metode interpolacije (videti Podpoglavlje 8.2).

Smicanje se definiše na sledeći način:

$$|x_{trans} \quad y_{trans} \quad 1| = |x \quad y \quad 1| \cdot T_{smicanje} \quad (8.17)$$

$$T_{smicanje} = \begin{vmatrix} 1 & s_x & 0 \\ s_y & 1 & 0 \\ 0 & 0 & 1 \end{vmatrix} \quad (8.18)$$

$$|x_{trans} \quad y_{trans} \quad 1| = |x + s_y \cdot y \quad y + s_x \cdot x \quad 1| \quad (8.19)$$

gde je s_x faktor smicanja duž x ose proporcionalno y koordinati i s_y faktor smicanja duž y ose proporcionalno x koordinati. U slučaju transformacije smicanja neophodno je primeniti metode interpolacije (videti Podpoglavlje 8.2).

Na Pr. 8.1 je prikazan programski kôd koji na jednostavnom primeru slike ilustruje sve gore opisane afina transformacije. Afina transformacije su definisane pomoću *skimage* funkcije `AffineTransform()` gde su redom definisane: translacija (t_x=`20`, t_y= `-10`), rotacija (θ=`10°`, funkcija `np.radians()` konvertuje stepene u radijane), skaliranje (k_x=`1.5`, k_y=`0.5`), smicanje (s_x=`0.5`, s_y=`0.3`). Funkcija `warp()` koristi inverznu transformaciju (`inverse`) da bi za svaku poziciju piksela na transformisanoj slici odredio koje koordinate tome odgovaraju na originalnoj slici. Rezultat izvršavanja kôda je prikazan na Pr. 8.1B.

Pr. 8.1. Primer *afina_transformacija.py* – ilustracija afina transformacija: translacije, rotacije, skaliranja i smicanja

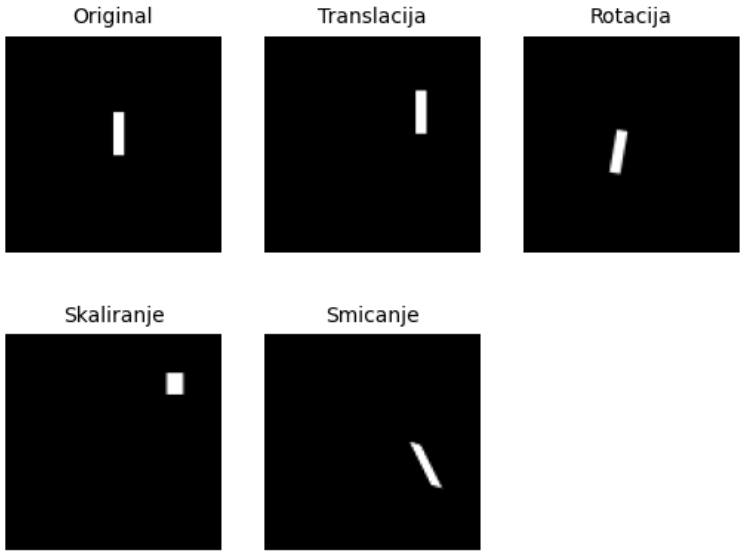
---

A. Programski kôd:

```
import numpy as np
import matplotlib.pyplot as plt
from skimage.transform import warp, AffineTransform

plt.close('all')
# Kreiranje slike
slika = np.zeros((100, 100))
slika[35:55, 50:55] = 1

# Translacija
translacija = AffineTransform(translation=(20, -10))
# Rotacija
rotacija = AffineTransform(rotation=np.radians(10))
# Skaliranje
skaliranje = AffineTransform(scale=(1.5, 0.5))
# Smicanje
s_x = 0.5
s_y = 0.3
```

```
matrica_smicanja = np.array([
    [1, s_x, 0],
    [s_y, 1, 0],
    [0, 0, 1]
])
smicanje = AffineTransform(matrix=matrica_smicanja)

# Primena transformacija
translirana_slika = warp(slika, translacija.inverse)
rotirana_slika = warp(slika, rotacija.inverse)
skalirana_slika = warp(slika, skaliranje.inverse)
smaknuta_slika = warp(slika, smicanje.inverse)

# Prikaz rezultata
outputs = [
    ('Original', slika),
    ('Translacija', translirana_slika),
    ('Rotacija', rotirana_slika),
    ('Skaliranje', skalirana_slika),
    ('Smicanje', smaknuta_slika),
     ]
slike = [slika, translirana_slika, rotirana_slika,
         skalirana_slika, smaknuta_slika]

plt.figure()
for i, (title, image) in enumerate(outputs, start=1):
    plt.subplot(2, 3, i)
    plt.title(title, fontsize=10)
    plt.imshow(image, cmap='gray')
    plt.axis('off')
```

B. Rezultat izvršavanja:



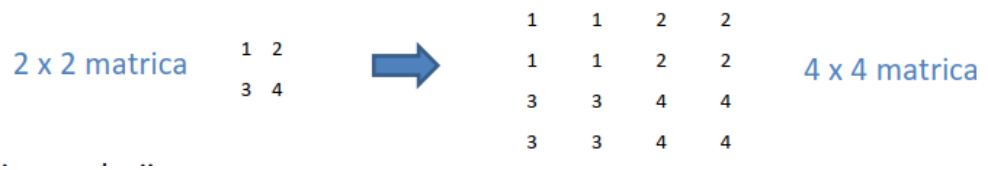
8.2. Interpolacija i ekstrapolacija

Interpolacija u biomedicinskom slikanju predstavlja procenu nepoznatih vrednosti intenziteta piksela na pozicijama koje se nalaze između poznatih vrednosti intenziteta piksela. Na primer, neka je data slika koja ima dimenzije 128x128, a potrebno je za bolju vizuelizaciju i analizu koristiti sliku dimenzije 1024x1024. Pomoću metoda interpolacije je moguće kreirati sliku većih dimenzija, a na osnovu poznatih vrednosti intenziteta piksela slike sa manjom dimenzijom. Osim za bolju vizuelizaciju, interpolacija se koristi i za otklanjanje artefakata koji nastaju usled primene prostornih transformacija. Pri proračunu prostornih transformacija, obično se primenjuje inverzna transformacija u kojoj se za svako (x_{trans}, y_{trans}) pronalazi (x, y) na originalnoj slici. Dobijeni (x, y) obično nisu celobrojni, pa se interpolacija koristi da izračuna vrednost piksela na toj poziciji.

Najčešće korišćene metode interpolacije su:

- interpolacija metodom najbližih suseda (eng. *Nearest-neighbor*)
- bilinearna interpolacija (eng. *Bi-linear*)
- bikubna interpolacija (eng. *Bi-cubic*).

Interpolacija metodom najbližeg suseda formira nove vrednosti ponavljanjem vrednosti najbližeg suseda. Jednostavan primer ovakve interpolacije za matricu 2x2 u veću matricu 4x4 je prikazan na Sl. 8.3.



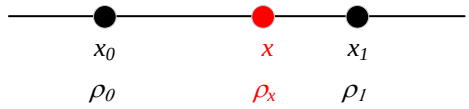
Slika 8.3 Primer interpolacije metodom najbližeg suseda

Za objašnjenje bilinearne interpolacije, neophodno je razumeti najpre koncept linearne interpolacije. Na Sl. 8.4 je prikazana tačka x sa intenzitetom ρ_x koja se nalazi na pravoj povučenoj kroz tačke x_0 i x_1 čije vrednosti intenziteta su ρ_0 i ρ_1, respektivno. Prema linearnoj interpolaciji, intenzitet ρ_x je definisan kao:

$$\rho_x = \rho_0 + \frac{\rho_1 - \rho_0}{x_1 - x_0}(x - x_0) \tag{8.21}$$

odakle se sređivanjem dobija:

$$\rho_x = \rho_0 \frac{x_1 - x}{x_1 - x_0} + \rho_1 \frac{x - x_0}{x_1 - x_0} \tag{8.22}$$



Slika 8.4 Koncept linearne interpolacije

Na Sl. 8.5 je koncept linearne interpolacije proširen u bilinearni koncept kod koga se određuje intenzitet piksela ρ_{xy} pozicioniranog između 4 piksela: (x_0, y_0), (x_0, y_1), (x_1, y_0) i (x_1, y_1) sa intenzitetima ρ_{00}, ρ_{01}, ρ_{10}, ρ_{11} respektivno. Primenom relacije (8.22) za:

- pravu određenu sa (x_0, y_0) i (x_1, y_0) dobija se:

$$\rho_{xy_0} = \rho_{00} \frac{x_1 - x}{x_1 - x_0} + \rho_{10} \frac{x - x_0}{x_1 - x_0} \tag{8.23}$$

- pravu određena sa (x_0, y_1) i (x_1, y_1) dobija se:

$$\rho_{xy_1} = \rho_{01} \frac{x_1 - x}{x_1 - x_0} + \rho_{11} \frac{x - x_0}{x_1 - x_0} \tag{8.24}$$

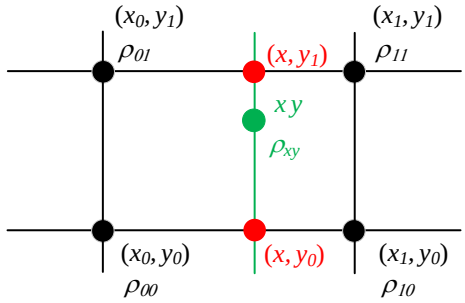
- pravu određenu sa (x, y_0) i (x, y_1)) dobija se:

$$\rho_{xy} = \rho_{xy_0} \frac{y_1 - y}{y_1 - y_0} + \rho_{xy_1} \frac{y - y_0}{y_1 - y_0} \tag{8.25}$$

Za x_1-x_0=1 i y_1-y_0=1 iz relacije (8.23), (8.24) i (8.25) se dobija finalni izraz za proračun bilinearne interpolacije za ρ_{xy}:

$$\rho_{xy} = \begin{pmatrix} 1-y \\ y \end{pmatrix} \begin{pmatrix} \rho_{00} & \rho_{10} \\ \rho_{01} & \rho_{11} \end{pmatrix} \begin{pmatrix} 1-x \\ x \end{pmatrix}^T \tag{8.26}$$

Po *default*-u, `warp()` funkcija korišćena u Pr. 8.1 koristi bilinearnu interpolaciju. U suprotnom, koristi se dodatni argument `order` čija vrednost određuje način interpolacije (npr. *order*=0 za interpolaciju metodom najbližih suseda, *order*=1 za bilinearnu interpolaciju, *order*=3 za bikubnu interpolaciju).



Slika 8.5 Koncept bilinearne interpolacije

Algoritam za bikubnu interpolaciju kao proširenje bilinearne interpolacije koje uzima u obzir 16 najbližih suseda detaljno je objašnjen u radu [Keys 1981], a ovde će ilustracije radi biti navedena samo jednačina koja se primenjuje za proračun interpoliranog intenziteta piksela $\rho_{x'y'}$:

$$\rho_{x'y'} = \sum_{i=-1}^{2} \sum_{j=-1}^{2} w(i, dx) w(j, dy) \rho_{x_i y_i} \qquad (8.27)$$

gde su (x_1, y_1) koordinate ciljne tačke za interpolaciju, a $x_i = x_1 + i$ i $y_j = y_1 + j$ su koordinate piksela u opsegu 4x4, dok se težine $w(i, dx)$ i $w(j, dy)$ za $dx = x' - x_i$ i $dy = y' - y_i$ računaju prema sledećoj relaciji:

$$w(i, dx) = \begin{cases} (a+2)|dx|^3 - (a+3)|dx|^2 + 1 & za\ |dx| \leq 1 \\ a|dx|^3 - 5a|dx|^2 + 8a|dx| - 4a & za\ 1 < |dx| < 2 \\ 0 & za\ |dx| \geq 2 \end{cases} \qquad (8.28)$$

$$w(i, dy) = \begin{cases} (a+2)|dy|^3 - (a+3)|dy|^2 + 1 & za\ |dy| \leq 1 \\ a|dy|^3 - 5a|dy|^2 + 8a|dy| - 4a & za\ 1 < |dy| < 2 \\ 0 & za\ |dy| \geq 2 \end{cases} \qquad (8.29)$$

gde je *a* konstanta (za *Catmull-Rom* splajn *a*= -0.5).

Ukoliko proračun prostorne transformacije zahteva vrednosti van granica originalne slike, onda se primenjuje neka od metoda ekstrapolacije, tj. dopunjavanja slično metodama opisanim u Potpoglavlju 5.2.1: dopunjavanje nulom ili nenultom konstantom, dopunjavanje vrednostima iz ivica, refleksijom vrednosti piksela u odnosu na ivicu slike i sl. Po *default*-u, `warp()` funkcija korišćena u Pr. 8.1 koristi ektrapolaciju vrednošću nula. U suprotnom, koristi se argument `mode` ove funkcije čija vrednost određuje tip ekstrapolacije (npr. ‘constant’, ‘edge’, ‘reflect’).

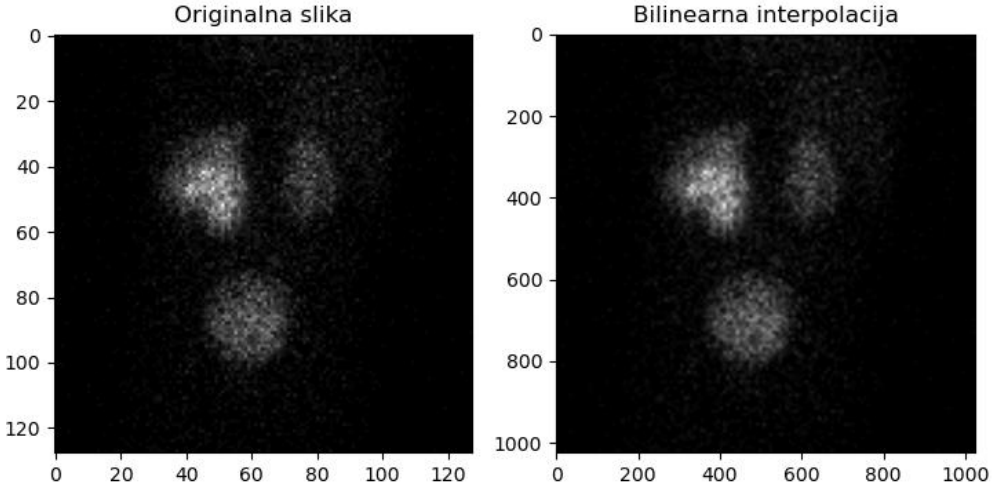
Zadatak za samostalni rad 8.1. Za 61. snimak dinamske renografije „*drsbru_001_POST.dcm*“ opisane i korišćene u Poglavlju 3 čija je dimenzija 128x128 kreirati snimak dimenzija 1024x1024 primenom metoda interpolacije. Koristiti *skimage* funkciju `resize()`:

```
interpoliran_snimak = resize(snimak_matrica,
                             (1024, 1024),
                             order=1)
```

gde su argumenti funkcije 61. snimak u obliku matrice 128x128 (`snimak_matrica`), nova dimenzija slike `(1024,1024)` i izbor metode interpolacije:

- interpolacija metodom najbližih suseda: `order`=0
- bilinearna interpolacija: `order`=1
- bikubna interpolacija: `order`=3.

Rešenje: Na Sl. 8.6 je prikazano poređenje originalnog 61. snimka dinamske renografije dimenzije 128x128 i slike dimenzija 1024x1024 nastale bilinearnom interpolacijom.



Slika 8.6 Ilustracija rezultata bilinearne interpolacije

8.3. Međusobna informacija

Međusobna informacija (eng. *Mutual information, MI*) je često korišćena metrika za izražavanje sličnosti dve registrovane slike. Nakon procesa registracije, porast međusobne informacije na dvema slikama je moguće proceniti na osnovu sledeće relacije:

$$MI = \sum_{\rho_{SL1}} \sum_{\rho_{SL2}} P(\rho_{SL1}, \rho_{SL2}) log \frac{P(\rho_{SL1}, \rho_{SL2})}{P(\rho_{SL1}) P(\rho_{SL2})} \tag{8.30}$$

gde je $P(\rho_{SL1},\rho_{SL2})$ verovatnoća pojavljivanja piksela sa vrednostima ρ_{SL1} u prvoj slici i ρ_{SL2} u drugoj slici na istim pozicijama, a $P(\rho_{SL1})$ je verovatnoća pojavljivanja piksela koje imaju vrednost ρ_{SL1} u prvoj slici i $P(\rho_{SL2})$ je verovatnoća pojavljivanja piksela koje imaju vrednost ρ_{SL2} u drugoj slici.

8.4. Metode zasnovane na uparivanju tačaka

Metode registracije zasnovane na uparivanju tačaka identifikuju i „usklađuju“ karakteristične tačke za dve slike da bi se proračunala transformacija koja najbolje poravnava te tačke. Karakteristične tačke mogu biti zadate manuelno ili automatski. U nastavku ovog poglavlja će biti prikazana primena homografije kao jedne od metoda registracije zasnovanih na uparivanju tačaka.

Homografija predstavlja transformaciju H koja od (x,y) koordinata karakterističnih tačaka na jednoj slici daje koordinate (x_{trans},y_{trans}) karakterističnih tačaka na drugoj slici:

$$|x_{trans} \quad y_{trans} \quad 1| = |x \quad y \quad 1| \cdot H \tag{8.31}$$

gde je H 3x3 matrica homografije:

$$H = \begin{vmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & h_{33} \end{vmatrix} \tag{8.32}$$

Elementi h_{ij} matrice homografije se određuju rešavanjem sistema linearnih jednačina koje se postavljaju na osnovu pozicija karakterističnih tačaka sa dve slike (npr. primenom metode najmanjih kvadrata). Nakon proračuna matrice homografije H, ona se primenjuje na sve piksele slike radi „usklađivanja“.

Nakon registracije, porast tzv. međusobne informacije (eng. *Mutual information, MI*), je moguće proceniti na osnovu relacije 8.30.

Na Pr. 8.2.A1-A2 je prikazan programski kôd koji na osnovu karakterističnih tačaka označenih manuelno na *MRI* i *CT* slici vrši registraciju primenom homografije [Badža Atanasijević et al. 2022]. Anonimizovana *MRI* i *CT* volumeni istog pacijenta („*MRI_anonimno.dcm*“ i „*CT_anonimno.dcm*“)

su dostupni zahvaljujući memorandumu o saradnji sa Univerzitetskom dečjom klinikom Tiršova. Na Pr. 8.2A1 je prikazan kôd koji učitava *MRI* i *CT* volumene i izdvaja 0-ti slajs na kome se vrši registracija. Manuelno izabranih 8 karakterističnih tačaka (`MRI_tacke` i `CT_tacke`) je zadato u samom kôdu i one se u rezultatu izvršavanja prikazuju u različitim bojama na *MRI* tj. *CT* slajsu.

Pr. 8.2.A1. Primer *homografija.py* – registracija *MRI* i *CT* slike homografijom – zadavanje karakterističnih tačaka

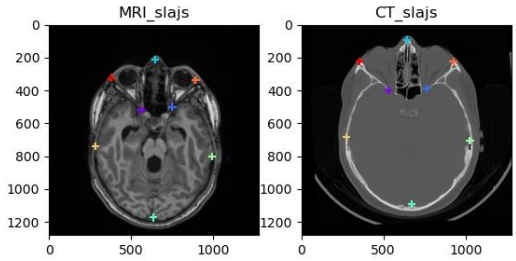
---

A. Programski kôd (karakteristične tačke):

```
import SimpleITK as sitk
import numpy as np
import matplotlib.pyplot as plt
import cv2
import matplotlib.cm as cm
import sklearn.metrics

plt.close('all')
# Ucitavanje slika
MRI_slika = sitk.ReadImage('MRI_anonimno.dcm')
MRI_slika_matrica = sitk.GetArrayFromImage(MRI_slika)
MRI_slajs = MRI_slika_matrica[0,:,:]

CT_slika = sitk.ReadImage('CT_anonimno.dcm')
CT_slika_matrica = sitk.GetArrayFromImage(CT_slika)
CT_slajs = CT_slika_matrica[0,:,:]

# Manuelno zadavanje karakterističnih tačaka
MRI_tacke = np.array([[ 563.71206048,  517.58968587],
                      [ 751.09982302,  498.85090961],
                      [ 646.162676  ,  210.2737553 ],
                      [ 634.91941025, 1173.44685475],
                      [ 994.70391432,  802.41908492],
                      [ 282.63041668,  738.70724566],
                      [ 893.51452255,  337.69743383],
```

```
                        [ 380.0720532 ,  322.70641283]])
CT_tacke = np.array([[ 531.64793223,  397.66151784],
                     [ 764.00875778,  386.41825209],
                     [ 644.08058975,   90.34558728],
                     [ 670.31487651, 1090.99623923],
                     [1026.35162533,  704.9774484 ],
                     [ 273.05281993,  682.4909169 ],
                     [ 925.16223356,  225.26477631],
                     [ 355.50343544,  221.51702106]])

fig, ax = plt.subplots(1,2)
ax[0].imshow(MRI_slajs, cmap='gray')
ax[0].set_title("MRI_slajs")
ax[1].imshow(CT_slajs, cmap='gray')
ax[1].set_title("CT_slajs")
colors = cm.rainbow(np.linspace(0, 1, 8))
ax[0].scatter(MRI_tacke[:,0],
              MRI_tacke[:,1],
              marker = '+', c = colors)
ax[1].scatter(CT_tacke[:,0],
              CT_tacke[:,1],
              marker = '+', c = colors)
```

B. Rezultat izvršavanja:



Na Pr. 8.2.A2 je prikazan kôd koji pomoću *OpenCV* funkcije `cv2.findHomography()` primenjuje relaciju (8.31) za traženje matrice homografije H (8.32) na osnovu karakterističnih tačaka. Promenljiva `status` daje „1“ za sve zadate karakteristične tačke, što znači da su sve tačke uspešno korišćene za izračunavanje H. Potom je na sve tačke *CT* slajsa primenjena

izračunata matrica homografije *H* tako što je primenjena funkcija `cv2.warpPerspective()`. Međusobna informacija *MI* pre i posle registracije homografijom je određena pomoću odgovarajuće funkcije `sklearn.metrics.mutual_info_score()` čiji argumenti su nizovi intenziteta piksela za *MRI* slajs i nizovi intenziteta piksela za *CT* slajs (ovi nizovi su dobijeni pomoću funkcije `flatten()`). Rezultat izvršavanja kôda je prikazan na Sl. 8.2.A2B. Prikazani su preklopljeni *MRI* i *CT* slajsevi (u različitim paletama boja i sa transparentnošću `alpha=0.7` za *CT* slajs) pre i posle registracije homografijom, kao i vrednosti međusobne informacije *MI* pre i posle registracije. Može se primetiti da je vrednosti *MI* porasla sa 0.71 na 0.88 nakon registracije.

Pr. 8.2.A2. Primer *homografija.py* – registracija *MRI* i *CT* slike homografijom – određivanje matrice homografije i primena transformacije

---

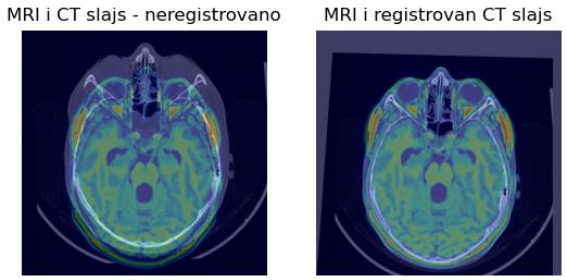
A. Programski kôd (matrica homografije):

```
H, status = cv2.findHomography(CT_tacke, MRI_tacke)
dimenzije_registrovan=(np.shape(MRI_slajs)[1],
                       np.shape(MRI_slajs)[0])
CT_registrovan = cv2.warpPerspective(CT_slajs, H,
                                     dimenzije_registrovan)


plt.figure()
plt.subplot(1,2,1)
plt.imshow(MRI_slajs,cmap = 'jet')
plt.imshow(CT_slajs, alpha = 0.7, cmap = 'gray')
plt.title("MRI i CT slajs - neregistrovano")
plt.axis('off')
plt.subplot(1,2,2)
plt.imshow(MRI_slajs,cmap = 'jet')
plt.imshow(CT_registrovan, alpha = 0.7, cmap = 'gray')
plt.title("MRI i registrovan CT slajs")
plt.axis('off')

MI_neregistrovano = sklearn.metrics.mutual_info_score(
    CT_slajs.flatten(), MRI_slajs.flatten())
```

```
print(f"Mutual information pre: {MI_neregistrovano}")
MI_registrovano = sklearn.metrics.mutual_info_score(
    CT_registrovan.flatten(), MRI_slajs.flatten())
print(f"Mutual information posle: {MI_registrovano}")
```

B. Rezultat izvršavanja:

```
Mutual information pre: 0.7072888436863478
Mutual information posle: 0.8876891383768318
```



8.5. Metode zasnovane na intenzitetima

Metode registracije zasnovane na intenzitetima definišu metriku sličnosti (npr. međusobnu informaciju *MI*, relacija 8.30) da bi kvantifikovale „usklađenost“ dve slike tokom procesa prostorne transformacije pokretne slike. Optimizacioni algoritam (npr. gradijent metrike sličnosti) podešava parametre prostorne transformacije tako da metrika sličnosti dostigne ekstremnu vrednost (u zavisnosti od prirode metrike to može biti minimalna ili maksimalna vrednost) i takva transformacija se onda proglašava optimalnom. Optimalna prostorna transformacija se finalno primenjuje na sve tačke pokretne slike kako bi je poravnala sa fiksnom slikom.

Na Pr. 8.3.A1-A3 je prikazan primer registracije *CT* snimka (datoteka „*training_001_ct.mha*“) i *MRI* snimka („*training_001_mr_T1.mha“*) istog pacijenta primenom metode zasnovane na intenzitetima. Slike i programski kôd su preuzeti i modifikovani iz primera *SimpleITK Notebooks*[43] [Yaniv et al. 2018]. Na početku Pr. 8.3.A1 je definisan modul `dodaj_metriku()` koji će omogućavati kreiranje liste vrednosti metrike sličnosti kroz iterativni proces registracije. Nakon učitavanja *CT* i *MRI* snimaka vrši se njihova konverzija u matrice da bi se vizuelizovao 11. slajs svake od njih (registracija će biti primenjena na ceo volumen, a 11. slajs se izdvaja samo radi prikaza).

[43] SimpleITK Notebooks je dostupan na linku (poslednji pregled Decembar 2024): https://insightsoftwareconsortium.github.io/SimpleITK-Notebooks/

Vizuelizovani slajsevi su prikazani preklopljeni u različitim paletama boja, sa transparencijom `alpha=0.5` za *CT* slajs. *CT* volumen će u ovom primeru biti fiksan, a *MRI* volumen će biti pokretan, tj. nad njim će se vršiti transformacija.

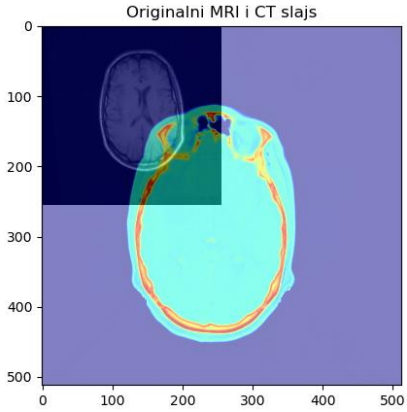
Pr. 8.3.A1. Primer *registracija_zasnovana_na_intenzitetu.py* – registracija *MRI* i *CT* slike metodom zasnovanom na intenzitetima – učitavanje volumena

---

A. Programski kôd (učitavanje volumena):

```
import SimpleITK as sitk
import matplotlib.pyplot as plt

plt.close('all')
# Kreiranje liste za čuvanje vrednosti metrika
metrika_lista = []
# Definisanje callback funkcije
# za dopunu vrednosti metrika sličnosti
def dodaj_metriku(metod_registracije):
    metrika = metod_registracije.GetMetricValue()
    metrika_lista.append(metrika)
# Učitavanje fiksne CT slike, pokretne MRI slike
fiksan_volumen = sitk.ReadImage("training_001_ct.mha",
                                sitk.sitkFloat32)
pokretni_volumen = sitk.ReadImage("training_001_mr_T1.mha",
                                  sitk.sitkFloat32)
fiksan_volumen_array=sitk.GetArrayFromImage(fiksan_volumen)
pokretni_volumen_array=sitk.GetArrayFromImage(pokretni_volumen)

plt.figure(1)
plt.imshow(pokretni_volumen_array[10,:,:], cmap = 'gray')
plt.imshow(fiksan_volumen_array[10,:,:], alpha = 0.5,
           cmap = 'jet')
plt.title("Originalni MRI i CT slajs")
```

B. Rezultat izvršavanja:



Na Pr. 8.3.A2. je prikazan programski kôd koji definiše transformacije koje će biti primenjene. Najpre se definiše pripremna transformacija koja za cilj ima centriranje slika (`sitk.CenteredTransformInitializer()`) i izjednačavanje njihovih dimenzija uz bilinearnu interpolaciju (`sitk.Resample()`). Radi vizuelne inspekcije, 11. slajsevi oba volumena su nakon pripremne transformacije prikazani preklopljeno. Pomoću funkcije `sitk.ImageRegistrationMethod()` je najpre inicijalizovan `metod_registracije` koji se u narednim koracima precizno definiše u pogledu: 1) izbora metrike sličnosti (međusobna informacija *MI*, `SetMetricAsMattesMutualInformation()`), 2) strategije uzorkovanja (nasumičan izbor 1% piksela za određivanje *MI* umesto da se koriste svi pikseli, `SetMetricSamplingStrategy()` i `SetMetricSamplingPercentage()`) i 3) optimizacionog algoritma (gradijent metrike sličnosti, `SetOptimizerAsGradientDescent())` koji u obzir uzima i razliku u fizičkim dimenzijama piksela za ova dva analizirana volumena (`SetOptimizerScalesFromPhysicalShift()`).

Pr. 8.3.A2. Primer *registracija_zasnovana_na_intenzitetu.py* – registracija *MRI* i *CT* slike metodom zasnovanom na intenzitetima – definisanje transformacija

---

A. Programski kôd (definisanje transformacija):

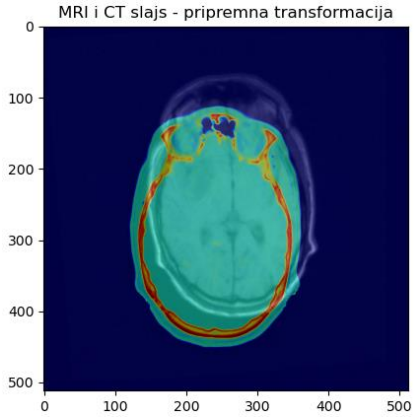
```
# Inicijalna transformacija za centriranje
inicijalna_trans = sitk.CenteredTransformInitializer(
    fiksan_volumen,
    pokretni_volumen,
```

```
    sitk.AffineTransform(fiksan_volumen.GetDimension()))
# Interpolacija
pokretna_interpolirana = sitk.Resample(
    pokretni_volumen,
    fiksan_volumen,
    inicijalna_trans,
    sitk.sitkLinear,
)

plt.figure(2)
pokretna_interpolirana_matrica=sitk.GetArrayFromImage(
    pokretna_interpolirana)
plt.imshow(pokretna_interpolirana_matrica[10,:,:],
           cmap = 'gray')
plt.imshow(fiksan_volumen_array[10,:,:], alpha = 0.5,
           cmap = 'jet')
plt.title("MRI i CT slajs - pripremna transformacija")

# Inicijalizacija metode registracije
metod_registracije = sitk.ImageRegistrationMethod()
# Postavljanje metrike sličnosti i optimizatora
metod_registracije.SetMetricAsMattesMutualInformation(
    numberOfHistogramBins=50)
# Slučajno uzorkovanje za MI
metod_registracije.SetMetricSamplingStrategy(
    metod_registracije.RANDOM)
metod_registracije.SetMetricSamplingPercentage(0.01)
# Postavljanje optimizatora
metod_registracije.SetOptimizerAsGradientDescent(
    learningRate=1.0,
    numberOfIterations=1000,
    convergenceMinimumValue=1e-6,
    convergenceWindowSize=10)
# Prilagoditi optimizator fizickim dimenzijama piksela
```

```
print("Dim. piksela (CT): ", fiksan_volumen.GetSpacing())
print("Dim. piksela (MRI): ", pokretni_volumen.GetSpacing())
metod_registracije.SetOptimizerScalesFromPhysicalShift()
```

B. Rezultat izvršavanja:

```
Dim. piksela (CT):  (0.653595, 0.653595, 4.0)
Dim. piksela (MRI):  (1.25, 1.25, 4.0)
```



Na Pr. 8.3.A3 je prikazan poslednji deo kôda koji poziva modul `dodaj_metriku()` i potom pokreće izvršavanje registracije koje sadrži tri faze: 1) pokretanje inicijalne transformacije za centriranje posmatranih volumena (`SetInitialTransform()`), 2) pokretanje odgovarajuće interpolacije (`SetInterpolator()`) i 3) pokretanje traženja optimalne transformacije (`Execute()`). Nakon nalaženja optimalne transformacije (`finalna_trans`), ona se primenjuje na sve piksele pokretnog volumena.

Pr. 8.3.A3. Primer *registracija_zasnovana_na_intenzitetu.py* – registracija *MRI* i *CT* slike metodom zasnovanom na intenzitetima – izvršavanje registracije

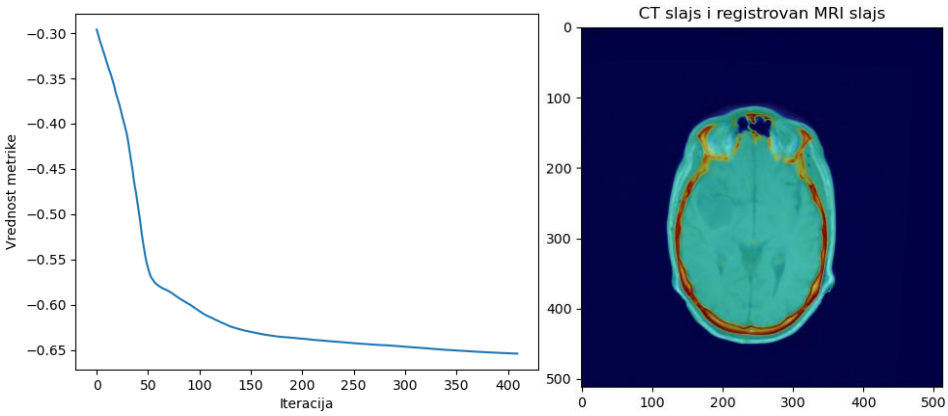
---

A. Programski kôd (izvršavanje registracije):

```
# Pozivanje callback funkcije
metod_registracije.AddCommand(
    sitk.sitkIterationEvent,
    lambda: dodaj_metriku(metod_registracije))
```

```
# Izvršavanje inicijalne transformacije
metod_registracije.SetInitialTransform(inicijalna_trans)

# Izvršavanje interpolacije
metod_registracije.SetInterpolator(sitk.sitkLinear)

# Nalaženje optimalne transformacije
finalna_trans = metod_registracije.Execute(
    fiksan_volumen, pokretni_volumen)

# Crtanje metrike
plt.figure(3)
plt.plot(metrika_lista)
plt.xlabel('Iteracija')
plt.ylabel('Vrednost metrike')

# Izvršavanje finalne transformacije na pokretnu sliku
pokretna_transformisana = sitk.Resample(
    pokretni_volumen,
    fiksan_volumen,
    finalna_trans,
    sitk.sitkLinear
)

# Prikaz rezultata registracije
pokretna_transformisana_matrica=sitk.GetArrayFromImage(
    pokretna_transformisana)
plt.figure(4)
plt.imshow(pokretna_transformisana_matrica[10,:,:],
           cmap = 'gray')
plt.imshow(fiksan_volumen_array[10,:,:], alpha = 0.5,
           cmap = 'jet')
plt.title("CT slajs i registrovan MRI slajs")
```

B. Rezultat izvršavanja:



8.6. Rezime poglavlja

- Registracija dve biomedicinske slike je proces transformacije piksela jedne slike tako da njihova nova pozicija odgovara istim fizičkim delovima biološkog sistema na drugoj slici.
- Koraci registracije obuhvataju: identifikaciju obeležja za uparivanje, sam proces uparivanja obeležja, proračun i primenu transformacione funkcije.
- Prema vrsti obeležja koja se uparuju, registracija može biti bazirana na uparivanju tačaka, uparivanju površina i uparivanju intenziteta/teksturalnih obeležja.
- Prema primenjenim matematičkim operacijama, registracija se deli na rigidnu (translacija i rotacija), afina (translacija, rotacija, skaliranje, smicanje) i nerigidnu transformaciju (nelinearne operacije za deformaciju oblika).
- Interpolacija se standardno primenjuje u predobradi biomedicinskih slika radi poboljšanja vizuelizacije i/ili otklanjanja artefakata prostornih transformacija. Često korišćene metode interpolacije su najbliži susedi, bilinearna i bikubna interpolacija.
- Međusobna informacija je metrika koja kvantifikuje efekat registracije na osnovu verovatnoća pojavljivanja parova intenziteta piksela na dve slike i verovatnoća pojavljivanja pojedinih intenziteta na svakoj od slika.
- Homografija je metoda registracije koja je bazirana na uparivanju karakterističnih tačaka i čiji je cilj pronalaženje transformacione matrice

koja na optimalan način poravnava karakteristične tačke kako bi se dalje primenila na sve piksele jedne slike.

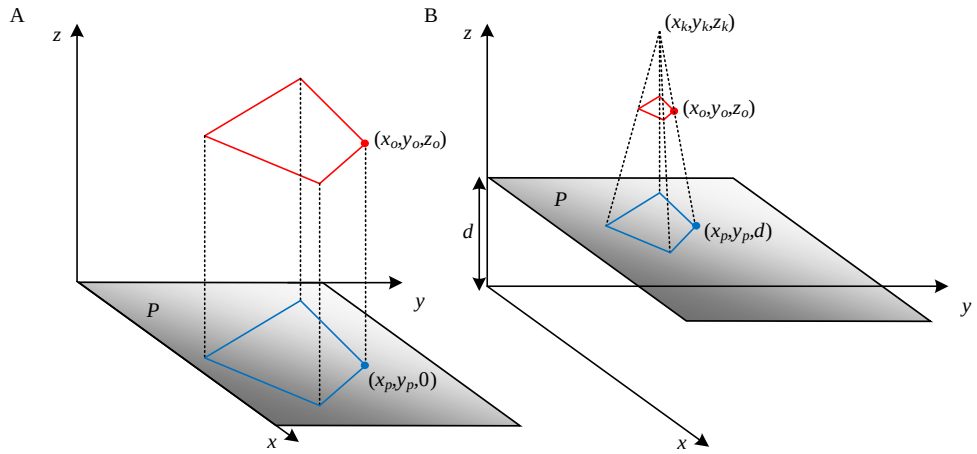
- Metode registracije zasnovane na intenzitetima obuhvataju sledeće korake: pripremnu transformaciju (centriranje, usklađivanje dimenzija i interpolacija), pronalaženje optimalne prostorne transformacije na osnovu metrike sličnosti i optimizacionog algoritma, primenu optimalne transformacije na pokretnu sliku.

9. OSNOVE VIZUELIZACIJE

Trodimenzioni podaci dobijeni biomedicinskim slikanjem se mogu vizuelizovati u obliku dvodimenzionih prikaza pomoću različitih metoda tzv. renderovanja. U ovom poglavlju će biti opisani primeri primene zapreminskog renderovanja i površinskog renderovanja.

9.1. Zapreminsko renderovanje

Zapreminsko renderovanje (eng. *volume rendering*) omogućava prikaz unutrašnjih slojeva trodimenzionih struktura. Pri zapreminskom renderovanju je potrebno odrediti vidljive površine tj. odbaciti skrivene površine kako bi se povećala efikasnost renderovanja. *Raycasting* renderovanje je jedna od najčešće primenjivanih metoda zapreminskog renderovanja. *Raycasting* metoda utvrđuje vidljivost površina praćenjem imaginarnog zraka svetlosti koji se kreće ka trodimenzionalnom objektu. Ovakav pristup baziran na imaginarnom zraku može biti ortogonalan (eng. *orthogonal*) ili perspektivni (eng. *perspective*). Sl. 9.1.



Slika 9.1. A) Ortogonalni *Raycasting*, B) Perspektivni *Raycasting*

Kod ortogonalnog *Raycasting*-a zraci polaze iz mreže koja je paralelna projekcionoj ravni, zraci su paralelni i normalni na projekcionu ravan P, a koordinate tačaka projekcije (x_p, y_p) ne zavise od udaljenosti mreže od projekcione ravni:

$$x_p = x_o, y_p = y_o \tag{9.1}$$

gde su (x_o,y_o) koordinate originalne pozicije tačke objekta.

Kod perspektivnog *Raycasting*-a svi zraci polaze iz jedne tačke posmatranja (tj. kamere) čije koordinate su (x_k, y_k, z_k). Na osnovu sličnosti trouglova, koordinate tačaka projekcije x_p i y_p u projekcionoj ravni koja je na rastojanju $z=d$ su izražene preko sledeće relacije:

$$x_p = \frac{x_o}{z_o/d}, y_p = \frac{y_o}{z_o/d} \tag{9.2}$$

gde su (x_o, y_o, z_o) koordinate originalne pozicije tačke objekta. Primenom perspektivnog *Raycasting*-a, objekti bliži tački posmatranja izgledaju veći, a dalji objekti izgledaju manji.

9.1.1. Projekcija maksimalnog intenziteta

Projekcija maksimalnog intenziteta (eng. *Maximum Intensity Projection*, *MIP*) prikazuje maksimalnu vrednost intenziteta duž imaginarnog zraka koji prolazi kroz volumen podataka. Rezultat *MIP*-a je projekcija volumena podataka pri čemu je vrednost intenziteta $\rho(x_p,y_p)$ svakog piksela projekcije:

$$\rho(x_p, y_p) = \max_z \rho(x, \mathrm{y}, z) \tag{9.3}$$

gde je $\rho(x, y, z)$ vrednost intenziteta tačke unutar volumena podataka, a maksimum se traži duž z pravca.

9.1.2. Projekcija sumiranog intenziteta

Projekcija sumiranog intenziteta (eng. *Summed Intensity Projection*, *SIP*) prikazuje sumiranu vrednost intenziteta duž imaginarnog zraka koji prolazi kroz volumen podataka. Rezultat *SIP*-a je projekcija volumena podataka pri čemu je vrednost intenziteta $\rho(x_p,y_p)$ svakog piksela projekcije:

$$\rho(x_p, y_p) = \sum_z \rho(x, \mathrm{y}, z) \tag{9.4}$$

gde je $\rho(x, y, z)$ vrednost intenziteta tačke unutar volumena podataka.

Na Pr. 9.1A je prikazan programski kôd koji ilustruje efekte *MIP* i *SIP* na ortogonalni *Raycasting* za *MRI* snimak tumora mozga „*RegLib_C01_1.nrrd*“ dostupan u okviru primera *3D slicer* alata[44] (*MRBrainTumor1*, vidi link alata u Prilogu A). Implementacija *MIP* i *SIP* je urađena primenom relacija (9.3) i (9.4) korišćenjem funkcija `np.max()` i `np.sum()` duž z pravca (`axis=2`), respektivno. Pomoću funkcije `np.flip()` je volumen flipovan u odnosu na x osu (`axis=0`). Rezultat izvršavanja kôda je prikazan na Pr. 9.1B. Prikazani su redom 81. slajs

[44] Datoteka je nakon instalacije *3D Slicer*-a i pokretanja opcije *File/Download Sample Data* dostupna u direktorijumu: *C:/Users/PC/AppData/Local/slicer.org/Slicer/cache/SlicerIO/*.

originalnog volumena i rezultati *MIP*-a i *SIP*-a. Može se primetiti da je tumor vidljiv na *SIP*-u, dok su krvni sudovi vidljiviji na *MIP*-u.

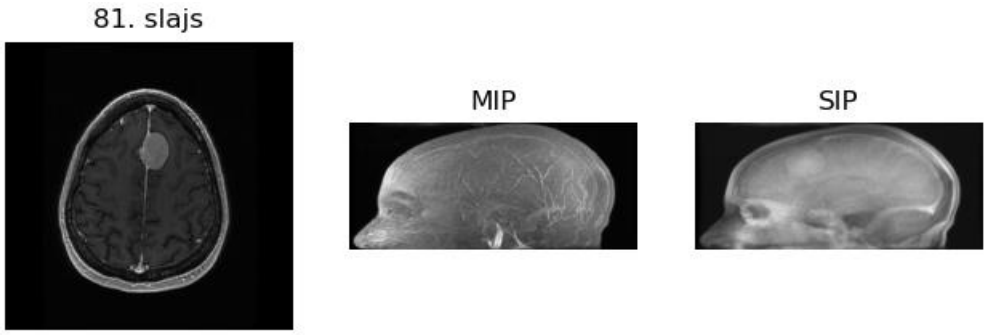
Pr. 9.1. Primer *MIP_SIP.py* – implementacija *MIP* i *SIP* ortogonalnog *Raycasting*-a

---

A. Programski kôd:

```
import SimpleITK as sitk
import matplotlib.pyplot as plt
import numpy as np

plt.close('all')
# Učitavanje volumena
volumen = sitk.ReadImage('RegLib_C01_1.nrrd')
volumen_matrica = sitk.GetArrayFromImage(volumen)

# MIP
MIP = np.max(volumen_matrica, axis=2)
MIP = np.flip(MIP, axis=0)
# SIP
SIP = np.sum(volumen_matrica, axis=2)
SIP = np.flip(SIP, axis=0)

plt.subplot(1,3,1)
plt.imshow(volumen_matrica[80,:,:], cmap='gray')
plt.title("81. slajs")
plt.axis("off")
plt.subplot(1,3,2)
plt.imshow(MIP, cmap="gray")
plt.title("MIP")
plt.axis("off")
plt.subplot(1,3,3)
plt.imshow(SIP, cmap="gray")
plt.title("SIP")
plt.axis("off")
```

B. Rezultat izvršavanja:



9.1.3. Direktno zapreminsko renderovanje

Direktno zapreminsko renderovanje (eng. *Direct Volume Rendering, DVR*) uvodi transfer funkcije boje i prozirnosti što omogućava vizuelizaciju slojeva unutar volumena. Imaginarni zrak na svom putu akumulira boju C' i prozirnost $alpha'$ na osnovu intenziteta tačaka volumena prema sledećim relacijama:

$$C' = C_a + (1 - alpha_a) \cdot C \tag{9.5}$$

$$alpha' = alpha_a + (1 - alpha_a) \cdot alpha \tag{9.6}$$

gde su C_a i $alpha_a$ akumulirana boja i prozirnost svih prethodnih uzoraka, (1-$alpha_a$) je količina svetlosti koja još nije blokirana, a C i $alpha$ su boja i prozirnost trenutnog uzorka (dodeljena na osnovu transfer funkcije tj. na osnovu mapiranja intenziteta piksela na boju i prozirnost). Proces se ponavlja za sve uzorke duž zraka sve dok se ili ne izađe iz volumena ili dok $alpha_a$ ne dostigne vrednost 1 (tada je zrak potpuno blokiran i više ne doprinosi slici).

Na Pr. 9.2.A1-A3 je prikazan programski kôd koji simulira glavu (velika sfera) sa tumorom (mala sfera) u njoj, a potom primenjuje perspektivni *Raycasting* sa *DVR* koji omogućava vizuelizaciju u interaktivnom prozoru. Na Pr. 9.2.A1 je prikazana simulacija sfera uz dodelu vrednosti intenziteta `50` za veliku sferu i vrednosti intenziteta `200` za malu sferu. Data je i konverzija trodiomenzionalnih podataka (`volumen_matrica`) u format koji *VTK* biblioteka podržava. Za konverziju se najpre formiraju `vtk_podaci` pomoću funkcije `numpy_to_vtk()` čiji argument je volumen pretvoren u niz (volumen je pretvoren u niz pomoću `ravel()` funkcije primenjene u

Fortran-style[45], `order='F'`). Funkcija `vtk.vtkImageData()` kreira *VTK* strukturu sa dimenzijama koje odgovaraju ulaznom volumenu (metod `SetDimensions()`) i koji pristupa skalarima *VTK* strukture puneći ih skalarnim vrednostima iz promenljive `vtk_podaci` (metod `GetPointData().SetScalars()`).

Pr. 9.2.A1. Primer *Raycasting_DVR.py* – primena *Raycasting* sa *DVR* – simulacija glave sa tumorom i konverzija u *VTK* format podataka

---

A. Programski kôd (simulacija i VTK format):

```
import vtk
import numpy as np
from vtkmodules.util.numpy_support import numpy_to_vtk

# Kreiranje velike sfere u čijoj unutrašnjosti je manja sfera
size = 128
x, y, z = np.mgrid[:size, :size, :size]
velika_sfera = np.sqrt((x - size // 2) ** 2 +
                       (y - size // 2) ** 2 +
                       (z - size // 2) ** 2) < 50
mala_sfera = np.sqrt((x - size // 2 + 10) ** 2 +
                     (y - size // 2) ** 2 +
                     (z - size // 2) ** 2) < 15
volumen_matrica = np.zeros((size, size, size))
volumen_matrica[velika_sfera] = 50
volumen_matrica[mala_sfera] = 200

# Konverzija volumena u VTK format
vtk_podaci = numpy_to_vtk(volumen_matrica.ravel(order='F'))
render_podaci = vtk.vtkImageData()
render_podaci.SetDimensions(volumen_matrica.shape)
render_podaci.GetPointData().SetScalars(vtk_podaci)
```

[45] *Fortran-style* za trodimenzionalne podatke formira niz tako što se najpre ređaju podaci po dubini (*z* osa), za svaki nivo dubine se elementi ređaju po *y*-osi (redovi), a unutar svakog reda se elementi ređaju po *x*-osi (kolone).

Na Pr. 9.2.A2 je prikazan deo kôda za definisanje transfer funkcije za boju i prozirnost za sva tri uzorka: pozadinu, malu sferu i veliku sferu. Najpre je kreirana odgovarajuća `boja_funkcija` kao instanca klase `vtkColorTransferFunction`, a potom su pomoću metode `AddRGBPoint()` svakom od intenziteta dodeljene RGB boje: pozadini gde su pikseli intenziteta `0` se dodeljuje (R,G,B)=`(0,0,0)`, velikoj sferi gde su pikseli intenziteta `50` se dodeljuje boja kože (R,G,B)=`(0.96,0.8,0.69)`, a maloj sferi gde su pikseli intenziteta `200` se dodeljuje crvena boja (R,G,B)=`(1,0,0)`. Slično se kreira i `prozirnost_funkcija` kao instanca klase `vtk.vtkPiecewiseFunction`, a potom su pomoću metode `AddPoint()` svakom od intenziteta dodeljeni procenti neprozirnosti: pozadini je dodeljno 0% (potpuna prozirnost), velikoj sferi koja simulira glavu je dodeljeno 50% neprozirnosti, a maloj sferi koja simulira tumor je dodeljeno 90% neprozirnosti. Konačno, definisane su osobine volumena tj. njegov izgled tokom renderovanja tako što je najpre kreirana `osobina_volumena` kao instanca klase `vtk.vtkVolumeProperty`, a potom je ova instanca povezana sa prethodno definisanim `boja_funkcija` i `prozirnost_funkcija`. Efekat osvetljenja je omogućen metodom `ShadeOn()`. Za interpolaciju je izabrana linearna interpolacija pomoću `SetInterpolationTypeToLinear()`.

Pr. 9.2.A2. Primer *Raycasting_DVR.py* – primena *Raycasting* sa *DVR* – definisanje *DVR* i osobina volumena

A. Programski kôd (DVR i osobina volumena):

```
# Definisanje transfer funkcije za boju
boja_funkcija = vtk.vtkColorTransferFunction()
boja_funkcija.AddRGBPoint(0.0, 0.0, 0.0, 0.0)
boja_funkcija.AddRGBPoint(50.0, 0.96, 0.8, 0.69)
boja_funkcija.AddRGBPoint(200.0, 1.0, 0.0, 0.0)

# Definisanje transfer funkcije za prozirnost
prozirnost_funkcija = vtk.vtkPiecewiseFunction()
prozirnost_funkcija.AddPoint(0.0, 0.0)
prozirnost_funkcija.AddPoint(50.0, 0.05)
prozirnost_funkcija.AddPoint(200.0, 0.9)
```

```
# Zadavanje osobina volumena
osobina_volumena = vtk.vtkVolumeProperty()
osobina_volumena.SetColor(boja_funkcija)
osobina_volumena.SetScalarOpacity(prozirnost_funkcija)
osobina_volumena.ShadeOn()
osobina_volumena.SetInterpolationTypeToLinear()
```

Na Pr. 9.2.A3 je prikazan deo kôda koji omogućava *Raycasting* renderovanje (po *default*-u je perspektivno renderovanje). Najpre se kreira maper kao instanca klase `vtk.vtkSmartVolumeMapper()` kojoj se za ulazne podatke daje *VTK* struktura formirana u Pr. 9.2.A1 (metod `SetInputData()`), a eksplicitno se zadaje i *Raycasting* metod renderovanja (`SetRequestedRenderModeToRayCast()`). Potom se kreira `volumen` za renderovanje (kao instanca klase `vtk.vtkVolume`) sa dodeljenim maperom (metod `SetMapper()`) i sa osobinom volumena definisanom u Pr.9.2.A2 (metod `SetProperty()`). Formira se `renderer` kao instanca klase `vtk.vtkRenderer` koja je odgovorna za proces renderovanja, a `volumen` se dodaje u `renderer` (metod `AddVolume()`). Kreira se prozor za renderovanje (`render_prozor` za vizuelizaciju volumena) kao instanca klase `vtk.vtkRenderWindow` u koji se dodaje `renderer`. Za omogućavanje interakcije korisnika za prozorom za renderovanje kreira se i `interaktor` kao instanca klase `vtk.vtkRenderWindowInteractor` kojoj se dodaje `render_prozor` (metod `SetRenderWindow()`). Konačno, sa metodom `Start()` primenjenom na `interaktor` se omogućava prikaz interaktivnog prozora za renderovanje. Rezultat kôdova Pr. 9.2A1-A3 je prikazan na Pr. 9.2.A3.B. Mogu se primetiti velika i mala sfera različitih boja i prozirnosti, a korisnik mišem može da „interaguje“ tj. da rotira prikazani volumen.

Pr. 9.2.A3. Primer *Raycasting_DVR.py* – primena *Raycasting* sa *DVR* – renderovanje

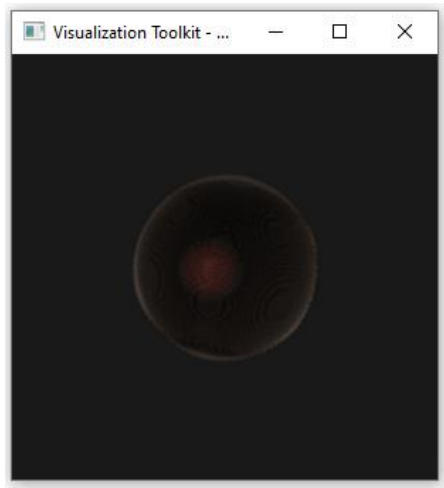
---

A. Programski kôd (renderovanje):

```
# Definisanje mapera
maper = vtk.vtkSmartVolumeMapper()
maper.SetInputData(render_podaci)
maper.SetRequestedRenderModeToRayCast()
```

```
volumen = vtk.vtkVolume()
volumen.SetMapper(maper)
volumen.SetProperty(osobina_volumena)

# Definisanje renderera, render prozora, interaktora
renderer = vtk.vtkRenderer()
renderer.AddVolume(volumen)

render_prozor = vtk.vtkRenderWindow()
render_prozor.AddRenderer(renderer)

interaktor = vtk.vtkRenderWindowInteractor()
interaktor.SetRenderWindow(render_prozor)

interaktor.Start()
```

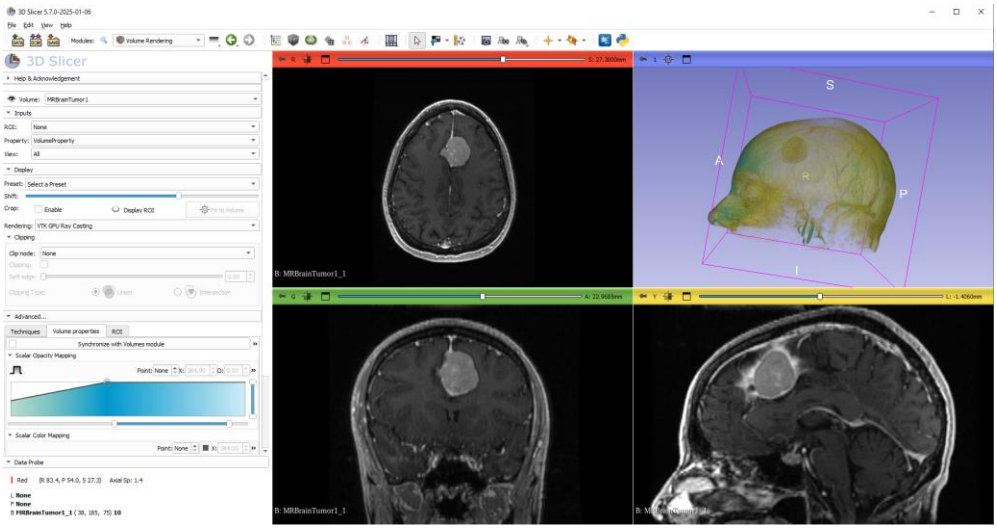
B. Rezultat izvršavanja:



Zadatak za samostalni rad 9.1. U alatu *3D Slicer* izabrati među primerima *MRBrainTumor1*, a potom u meni liniji među *Modules* uzabrati *Volume Rendering*. Pomoću opcije *Shift* menjati *Scalar Opacity Mapping* i *Scalar Color Mapping*.

Rešenje:

Na Sl. 9.2 je pikazan primer rezultata: izabrani slajsevi u aksijalnoj (gore levo), koronarnoj (dole levo) i sagitalnoj (dole desno) ravni na kojima se vidi pozicija tumora i renderovani prikaz (gore desno) na kome je takođe uočljiva pozicija tumora.



Slika 9.2. Renderovanje u *3D Slicer*-u

Zadatak za samostalni rad 9.2. Primeniti programski kôd iz Pr. 9.2.A1-A3 direktno na `volumen_matrica` iz Pr. 9.1. Diskutovati rezultat renderovanja u odnosu na prikaz sa Sl. 9.2.

9.2. Površinsko renderovanje

Površinsko renderovanje (eng. *surface rendering*) se koristi za vizuelizaciju trodimenzionalnih struktura koje su predstavljene kao površine. Za razliku od zapreminskog renderovanja gde nije potrebno uraditi segmentaciju struktura od interesa radi vizuelizacije, kod površinskog renderovanja je segmentacija neophodan korak. Metode površinskog renderovanja su računarski manje zahtevne od zapreminskog renderovanja.

Marching Cubes je jedna od često korišćenih metoda koja omogućava ekstrakciju izopovršina iz trodimenzionalnih skalarnih polja (npr. 3D volumena dobijenih biomedicinskim slikanjem) i njihovu konverziju u

poligonalne mreže koje dalje predstavljaju standardni ulaz u render. Osnovni koraci *Marching Cubes* metode su sledeći:

- svaki voksel volumena ima osam tačaka (osam vrhova kocke koja predstavlja voksel) i za vrednosti u tih osam tačaka se proverava da li su iznad (logička „1“) ili ispod izo-praga (logička „0“). Spram ovakve provere, za svaki voksel je moguće identifikovati jedan od 2^8=256 mogućih konfiguracija.
- određivanje ivica koje seku površinu primenom unapred definisane *lookup* tabele (ako je jedna tačka iznad praga, a susedna tačka ispod praga, površina mora da preseca ivicu koja ih povezuje). Za identifikovanu ivicu iz prethodnog koraka, algoritam određuje tačku preseka koristeći interpolaciju između vrednosti na krajevima ivice.
- na osnovu preseka se generiše niz trouglova koji aproksimiraju površinu unutar kocke. Formiranje trouglova je takođe definisano *lookup* tabelom.
- trouglovi svih voksela se kombinuju u poligonalnu mrežu.

Za detaljnije upoznavanje za *Marching Cubes* algoritmom preporučena literatura je [Lorenson and Cline 1987].

Na Pr. 9.3.A1-A2 je prikazan programski kôd za površinsko renderovanje koji za ekstrakciju površina koristi *Marching Cubes* algoritam. Datoteka koja sadrži *MRI* snimak tumora mozga „*RegLib_C01_1.nrrd*“ i koja je korišćena i u Pr. 9.1. je korišćena i za potrebe ovog primera. Međutim, za potrebe površinskog renderovanja najpre je urađena manuelna segmentacija tumora primenom *Segment Editor* opcije u *3D Slicer*-u i sačuvana je kao datoteka „*Segmentation_tumor.nrrd*“. Datoteka sa segmentiranim tumorom je ulazni podatak u programskom kôdu Pr. 9.3A1. Najpre je kreiran `ulaz` kao instanca klase `vtk.vtkNrrdReader` kome je postavljeno ime datoteke `ulaz.SetFileName` i iniciran proces čitanja (metoda `Update()`). Segmentirani podaci su bili sačuvani tako da su sa „1“ bili označeni vokseli tumora, a sa „0“ ostali vokseli. U nastavku kôda je kreirana instanca objekta `ThresholdBetween()` koja je nazvana `labeliranje` i povezana je sa učitanom segmentacijom (metod `SetInputConnection()`), a vrednosti formiranog *VTK* objekta su postavljene na 255 (tamo gde je ulazna segmentacija imala vrednost „1“, metod `SetInValue()`) ili na 0 (tamo gde je ulazna segmentacija imala vrednost „0“, metod `SetOutValue()`). Primena *Marching Cubes* algoritma je realizovana kreiranjem `mcubes` kao instance objekta `vtk.vtkMarchingCubes()` povezanog sa `labeliranje.` U ovom primeru se računa samo jedna izopovršina sa

nivoom intenziteta 255 (`mcubes.SetValue(0, 255)`). Za određivanje geometrijskih osobina izopovršine iskorišćena je instanca objekta `vtk.vtkMassProperties()` koja je povezana sa `mcubes` za koju metoda `GetVolume()` može da proceni volumen ako je površina zatvorena.

Pr. 9.3.A1. Primer *povrsinsko_renderovanje.py* – primena *Marching Cubes* – učitavanje segmentacije, labeliranje, ekstrakcija izopovršina i proračun volumena tumora

A. Programski kôd (ekstrakcija izopovršina, volumen tumora):

```
import vtk

ulaz = vtk.vtkNrrdReader()
ulaz.SetFileName('Segmentation_tumor.nrrd')
ulaz.Update()

# Labeliranje
labeliranje = vtk.vtkImageThreshold()
labeliranje.ThresholdBetween(1, 1)
labeliranje.SetInValue(255)
labeliranje.SetOutValue(0)
labeliranje.SetInputConnection(ulaz.GetOutputPort())

# Marching cubes
mcubes = vtk.vtkMarchingCubes()
mcubes.SetInputConnection(labeliranje.GetOutputPort())
mcubes.SetValue(0, 255)

# Proračun volumena tumora
osobina_mase = vtk.vtkMassProperties()
osobina_mase.SetInputConnection(mcubes.GetOutputPort())
volumen = osobina_mase.GetVolume()
print(f"Zapremina tumora je: {volumen} mm3")
```

Na Pr. 9.3A2 je prikazan deo kôda koji omogućava definisanje mapera poligonskih mreža i potom renderovanje. Maper je inicijalizovan kao instanca klase `vtk.vtkPolyDataMapper()` koja je povezana sa `mcubes`. Potom je kreirana površina kao instanca klase `vtk.vtkActor()` i povezana sa definisanim maperom. Za prozirnost je postavljena vrednost 50% (metod `GetProperty().SetOpacity()`). Ostatak kôda koji opisuje inicijalizaciju i startovanje procesa renderovanja i interakcije sa prozorom za renderovanje je identičan kao u Pr. 9.2A3. Rezultat izvršavanja kôda za Pr. 9.3A1-A2 je prikazan na Pr. 9.3.A2.B.

Pr. 9.3.A2. Primer *povrsinsko_renderovanje.py* – primena *Marching Cubes* – renderovanje

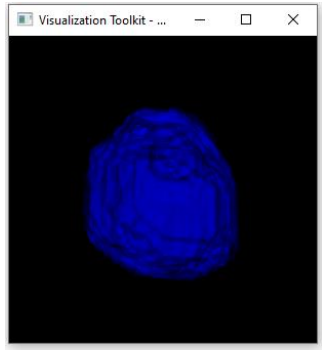
---

A. Programski kôd (renderovanje):

```
# Definisanje mapera
maper = vtk.vtkPolyDataMapper()
maper.SetInputConnection(mcubes.GetOutputPort())

povrsina = vtk.vtkActor()
povrsina.SetMapper(maper)
povrsina.GetProperty().SetOpacity(0.5)

# Definisanje renderera, render prozora, interaktora
renderer = vtk.vtkRenderer()
renderer.AddActor(povrsina)
render_prozor = vtk.vtkRenderWindow()
render_prozor.AddRenderer(renderer)
interaktor = vtk.vtkRenderWindowInteractor()
interaktor.SetRenderWindow(render_prozor)
interaktor.Start()
```

B. Rezultat izvršavanja:

```
Zapremina tumora je:
15583.88533923868 mm3
```



9.3. Rezime poglavlja

- Renderovanje je osnovna tehnologija za generisanje vizuelizacija trodimenzionalnih volumena dobijenih tehnikama biomedicinskog slikanja.
- Osnovna podela metoda renderovanja je na zapreminsko i površinsko renderovanje.
- Zapreminsko renderovanje je računarski zahtevnije, ali ne zahteva prethodnu segmentaciju trodimenzionalnih struktura koje treba vizuelizovati. Površinsko renderovanje prikazuje samo spoljašnje površine (izopovršine) objekta.
- *Raycasting* renderovanje koristi imaginarne zrake koji prolaze kroz trodimenzionalnu strukturu i računa vrednosti piksela na osnovu osobina materijala kroz koje zraci prolaze.
- Projekcija maksimalnih intenziteta i projekcija sumiranih intenziteta određuju vrednost piksela kao maksimum, tj. sumu, respektivno, intenziteta piksela duž imaginarnog zraka. Direktno zapreminsko renderovanje uvodi transfer funkcije boja i prozračnosti u procesu vizuelizacije.
- *Marching Cubes* generiše poligonsku mrežu trodimenzionalnih izopovršina tako što identifikuje preseke površina sa ivicama voksela.

10. RADIOMIKA

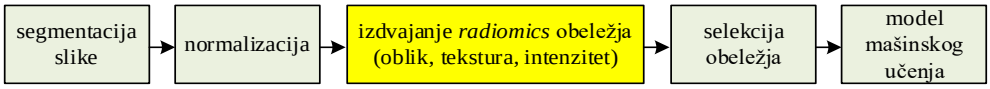
Radiomika (eng. *Radiomics*) je napredna oblast biomedicinskog slikanja koja se bavi ekstrakcijom i analizom kvantitativnih karakteristika iz biomedicinskih slika, a sa krajnjim ciljem da se poboljša dijagnostika bolesti, predikcija toka bolesti ili da se personalizuje terapija. Prema vrsti metoda obrade slike i modela mašinskog učenja koji se koristi, radiomika se deli na tradicionalnu (klasičnu) i duboku radiomiku. Moguće je koristiti i hibridne pristupe koji kombinuju tradicionalnu i duboku radiomiku.

10.1. Tradicionalna radiomika

Tradicionalna radiomika se bavi ekstrakcijom velikog broja kvantitativnih obeležja (za oblik, teksturu, intenzitet) iz biomedicinskih slika i primenom statističke analize i tradicionalnih modela mašinskog učenja na izdvojenim obeležjima.

Na Sl. 10.1 je prikazan celokupan tok postupka koji se primenjuje prilikom korišćenja tradicionalne radiomike. Pre izdvajanja obeležja potrebno je najpre uraditi:

- segmentaciju slika tj. izdvajanje regiona ili zapremine od interesa (koja je detaljno opisana u poglavlju 6 *Segmentacija*)
- „homogenizaciju“ slika (tj. normalizaciju slika spram različitih dimenzija piksela, intenziteta i slično kako bi kasnije izdvajana obeležja bila robusna i reproducibilna).



Slika 10.1. Tok primene tradicionalne radiomike

U poglavlju 5 *Filtriranje* su dati primeri obeležja koja je moguće izdvojiti iz biomedicinskih slika. S obzirom na to da postoji velika raznovrsnost u načinima proračuna obeležja, za izdvajanje tzv. *radiomics* obeležja se preporučuje poštovanje vodiča koji je kreiran od strane *Image Biomarker Standardization Initiative* (*IBSI*)[46]. Postoje različiti besplatno dostupni alati za izdvajanje *radiomics* obeležja prema IBSI vodiču: biblioteka

[46] *IBSI website*: https://ibsi.readthedocs.io/en/latest/index.html (poslednji pregled Decembar 2024)

Python-a *pyRadiomics*[47] (postoji i kao *plugin* za *3D Slicer* softer), *LifEx*[48] softver, *MaZda*[49] softver i dr.

Nakon izdvajanja obeležja se razmatra selekcija obeležja koja se može temeljiti: na statističkoj analizi (npr. izbor najboljih pojedinačnih obeležja, eng. *univariate feature selection*), analizi značajnosti obeležja (eng. *feature importance*, npr. primenom algoritama slučajnih šuma, eng. *Random Forest*[50]), metodama redukcije broja dimenzija (npr. metodom rekurzivne eliminacije obeležja, eng. *Recursive Feature Elimination*) i analizi kolinearnosti i korelacije obeležja (analiza koja rezultuje klasterima visoko korelisanih obeležja iz kojih se za dalju analizu može uzeti po jedan predstavnik iz svakog klastera). Konačno, redukovani skup obeležja (dobijen na skupu za obučavanje) se koristi za dalju primenu u modelima tradicionalnog mašinskog učenja.

Dva osnovna tipa mašinskog učenja koja se koriste u oblasti radiomike su nagledano (eng. *supervised*) i nenadgledano (eng. *unsupervised*) mašinsko učenje. Modeli nenadgledanog mašinskog učenja (*K-means* klasterovanje, hijerarhijsko klasterovanje i sl.) se koriste kada nema označenih tj. labeliranih podataka. Na primer, nenadgledano učenje se može iskoristiti za grupisanje pacijenata u podgrupe sa različitim prognostičkim ishodima, a na osnovu sličnosti u obeležjima biomedicinskih slika. Nenadgledano učenje nema konkretne oznake (labele) na osnovu kojih vrši proces učenja, već u samim biomedicinskim karakteristikama pokušava da pronađe obrasce. Sa druge strane, modeli nadgledanog mašinskog učenja (logistička regresija, k najbližih suseda, model slučajnih šuma, model potpornih vektora, potpuno povezane neuralne mreže itd.) se koriste kada postoji poznata labela (npr. poznata je dijagnoza ili ishod terapije) i model se trenira na označenim tj. labeliranim podacima. Na primer, pri dijagnostici tumora, nadgledani algoritmi mašinskog učenja mogu se iskoristiti za klasifikaciju biomedicinskih slika („slika sa tumorom“ vs „slika bez tumora“) na osnovu prethodno labeliranih slika. Za potrebe tradicionalne radiomike, modeli nadgledanog mašinskog učenja se mnogo češće koriste nego modeli nenadgledanog mašinskog učenja. Za detaljnije informisanje o načinu rada pomenutih modela nadgledanog i nenadgledanog mašinskog učenja preporučena literatura je [Bishop 2006], [Russel et al. 2010] i [Hastie et al. 2008].

[47] *Radiomics website*: https://www.radiomics.io/index.html (poslednji pregled Decembar 2024)
[48] *LifEx website*: https://www.lifexsoft.org/ (poslednji pregled Decembar 2024)
[49] *MaZda website*: https://qmazda.p.lodz.pl/ (poslednji pregled Decembar 2024)
[50] Primeri implementacije proračuna značajnosti obeležja primenom *Random Forest* algoritma su dostupni na linku: https://www.geeksforgeeks.org/feature-importance-with-random-forests/ (poslednji pregled Decembar 2024)

10.1.1. Tradicionalni pristupi nadgledanog mašinskog učenja

U ovom poglavlju će biti dat kratak pregled principa najčešće korišćenih pristupa u nadgledanom mašinskom učenju: logistička regresija, k najbližih suseda, model slučajnih šuma, model potpornih vektora i potpuno povezane neuralne mreže. Svi navedeni algoritmi mašinskog učenja se zasnivaju na istom principu, a to je da ne postoje eksplicitno definisana pravila na osnovu kojih se obrađuju ulazni podaci, već se koriste podaci skupa za obučavanje kako bi se vršilo mapiranje ulaznih parametara na izlazne (npr. mapiranje karakteristika biomedicinskih slika na klase „zdrav"/„bolestan").

Logistička regresija (eng. *Logistic Regression, LR*) klasifikuje na osnovu poređenja verovatnoće da podatak *y* pripada određenoj klasi (npr. klasi "1") sa pragom, pri čemu se verovatnoća dobija primenom logističke (sigmoidne) funkcije na linearnu kombinaciju ulaznih podataka *x*. Formula za verovatnoću kod binarne logističke regresije je:

$$P(y = 1 \mid x) = \frac{1}{1+e^{-(w^T x+b)}} \tag{10.1}$$

gde je *x* vektor ulaznih podataka, a *w* i *b* parametri. Navedena formula se može uopštiti za slučaj više klasne klasifikacije.

Model k najbližih suseda (eng. *k-Nearest Neighbors, kNN*) klasifikuje podatak na osnovu „glasanja" *k* najbližih suseda pri čemu se za procenu udaljenosti suseda obično koristi Euklidska udaljenost. Mogu se koristiti i druge metrike udaljenosti, kao i metrike gde različiti ulazni parametri imaju različite težine za svoju udaljenost.

Model slučajnih šuma (eng. *Random Forest, RF*) za klasifikaciju koristi više stabala odlučivanja za donošenje odluka (svako stablo se trenira na nasumičnom podskupu podataka), pri čemu se konačna odluka o pripadnosti klasi donosi na osnovu većinskog glasanja stabala.

Model potpornih vektora (eng. *Support Vector Machine, SVM*) razdvaja klase podataka tako što nalazi tzv. hiper-ravni koje razdvajaju podatke pri čemu je kriterijum za postavljanje hiper-ravni taj da se one nalaze što dalje od najbližih tačaka svake od klasa tj. takozvanih „potpornih vektora".

Arhitektura neuralnih mreža širenja unapred (eng. *Fully Connected Neural Networks, FCNN*) se sastoji iz veštačkih neurona organizovanih u sledeće slojeve: ulazni sloj (ulazni podaci), jedan ili više skrivenih slojeva za obradu podataka i izlazni sloj (daje rezultat klasifikacije). Svaki neuron u sloju je povezan sa neuronima u sledećem sloju. Za svaki neuron se izračunava linearna kombinacija sa pomerajem (eng. *bias*) njegovih ulaznih podataka *x* ($y=w^Tx+b$, gde je *w* vektor težina veza između neurona, a *b* vektor pomeraja tj. *bias*-a). Potom se primenjuje aktivaciona funkcija (npr. sigmoidna – vidi

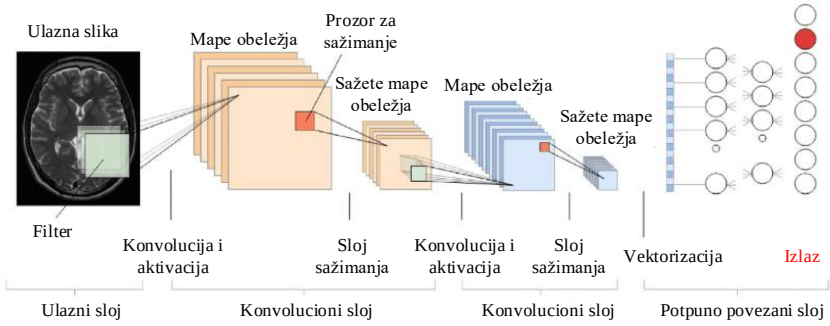
(10.1), *ReLU* [51] i sl.) i prosleđuje se rezultat dalje sledećem sloju. Na svom izlazu, *FCNN* mreža daje predikciju klase koja se poredi sa labeliranim vrednostima i proračunava se greška koja se zatim koristi za prilagođavanje parametara modela (težine veza i pomeraja) kako bi se greška u narednoj iteraciji smanjila.

10.2. Duboka radiomika

Duboka radiomika koristi duboke neuralne mreže, pre svega konvolucione neuralne mreže (eng. *Convolutional Neural Networks*, *CNN*) za ekstrakciju i analizu kvantitativnih karakteristika iz biomedicinskih slika. Za duboku radiomiku nije potrebno ručno izdvajanje obeležja kao kod tradicionalne radiomike, već duboka neuralna mreža automatski uči i izdvaja karakteristike iz slika tokom procesa obuke. Duboka radiomika omogućava da se uzmu o obzir složeniji obrasci u slikama zahvaljujući sposobnosti dubokih neuralnih mreža da uče složene reprezentacije podataka. U narednom poglavlju će biti objašnjene osnove rada *CNN* mreža kao najzastupljenijih u analizi medicinske slike, a za detaljnije informisanje o modelima dubokih neuralnih mreža preporučena literatura je [Goodfellow et al. 2016] i [Zhang et al. 2023].

10.2.1. Model konvolucionih neuralnih mreža

Standardni elementi *CNN* mreže su: ulazni sloj tj. slika (eng. *Input layer*), konvolucioni sloj (eng. *Convolutional layer*), sloj sažimanja (eng. *Pooling layer*), potpuno povezani sloj (eng. *Fully connected layer*) i izlazni sloj (eng. *Output layer*). Na Sl. 10.2 je kao primer ilustrovana arhitektura jedne *CNN* mreže.



Slika 10.2. Ilustracija primera arhitekture *CNN* mreže. Slika je preuzeta iz [Zhang et al. 2023] uz prevod natpisa. Slika je podeljena pod CC BY 4.0 DEED licencom (https://creativecommons.org/licenses/by/4.0/), poslednji pregled Decembar 2024

[51] *ReLU* (eng. *Rectified Linear Unit*) aktivaciona funkcija se definiše na sledeći način: $f(x)=\max(0,x)$

Konvolucioni slojevi izdvajaju lokalna obeležja iz slika (npr. ivice, uglove, teksturu) primenom prostornih filtera tj. kernela (opisanih u poglavlju 5 *Filtriranje*) i generišu mape obeležja. Treća dimenzija mape obeležja je jednaka broju primenjenih filtera. Konvolucioni sloj integriše i primenu neke od aktivacionih funkcija (npr. *ReLU*) koje omogućavaju mreži da nauči složene nelinearne relacije među podacima.

Slojevi sažimanja redukuju prostorne dimenzije mapa obeležja radi računarske efikasnosti i izbegavanja preobučavanja (eng. *overfitting*) tj. radi povećanja otpornosti modela na male varijacije u ulaznim podacima. Postoje dva glavna načina sažimanja:

- uzimanjem najveće vrednosti intenziteta iz malog regiona (eng. *max pooling*)
- uzimanjem srednje vrednosti intenziteta iz malog regiona (eng. *average pooling*).

Na kraju, sažete mape obeležja se transformišu u vektorski oblik, tj. u niz vrednosti). Na ovako dobijeni vektor, u potpuno povezanom sloju se primenjuje neuralna mreže (*FCNN*, rekurentna mreža i sl.) kako bi se identifikovao željeni izlaz (rezultat detekcije, segmentacije, klasifikacije ili registracije). Najčešće korišćena aktivaciona funkcija u potpuno povezanom izlaznom sloju je *softmax*[52] funkcija i njen izlaz daje verovatnoću svake klase.

10.3. Primena nadgledanog mašinskog učenja u radiomici

Labeliranje. Nadgledano mašinsko učenje podrazumeva da su dostupne labele klasa tj. prave vrednosti (eng. *ground truth*) pripadnosti klasama. Labele se mogu dobiti:

- glasanjem u kome učestvuje više observera (eksperata) pri čemu se za vrednost labele usvaja vrednost dobijena većinskom odlukom (eng. *Consensus voting*)
- korišćenjem dodatne medicinske tehnike (npr. biopsije ili nekog drugog modaliteta slikanja).

Obučavanje. Razvoj modela nadgledanog mašinskog učenja podrazumeva da se na skupu za obučavanje podešavaju parametri modela kako bi performanse modela bile što bolje, tj. funkcija gubitaka (eng. *Loss function*), koja odražava stepen neslaganja predikcije modela i labele što manja. Za funkciju gubitaka se često u modelima klasifikacije koristi krosentropijska funkcija gubitaka L_{CE} (eng. *Cross-Entropy Loss*) koja se definiše na sledeći način:

[52] *Softmax* funkcija se definiše na sledeći način: $f(x)_i = \frac{e^{x_i}}{\sum_{j=1}^{n} e^{x_j}}$, $i \in [1,n]$, n - broj klasa.

$$L_{CE} = -\sum_{i=1}^{k} y_i \log(\hat{y}_i) \quad (10.2)$$

gde je y_i labela, $\hat{y}_i$ verovatnoća predikcije te klase (izlaz *softmax* funkcije), a k broj klasa.

U slučaju nebalansiranih skupova podataka (kada nisu sve klase jednako zastupljene), neophodno je uzeti u obzir težinske faktore svake klase, tj. u slučaju nebalansiranih klasa funkcija gubitaka se modifikuje u težinsku funkciju gubitaka. Na primeru krosentropijske funkcije gubitaka, bi težinska krosentropijska funkcija gubitaka L_{WCE} (eng. *Weighted Cross-Entropy Loss*) bila definisana kao:

$$L_{WCE} = -\sum_{i=1}^{k} w_i y_i \log(\hat{y}_i) \quad (10.3)$$

gde je y_i labela, $\hat{y}_i$ verovatnoća predikcije te klase (izlaz *softmax* funkcije), k broj klasa, a w_i su težinski faktori koji se računaju kao odnos broja pripadnika svim drugim klasama osim klasi i i ukupnog broja uzoraka.

U slučaju primene modela nadgledanog mašinskog učenja za proces segmentacije slika, često se koristi funkcija gubitaka bazirana na *Dice* metrici, $L_{Soft\ Dice}$ (eng. *Soft Dice Loss function*) koja je ranije pomenuta u poglavlju 6 *Segmentacija*:

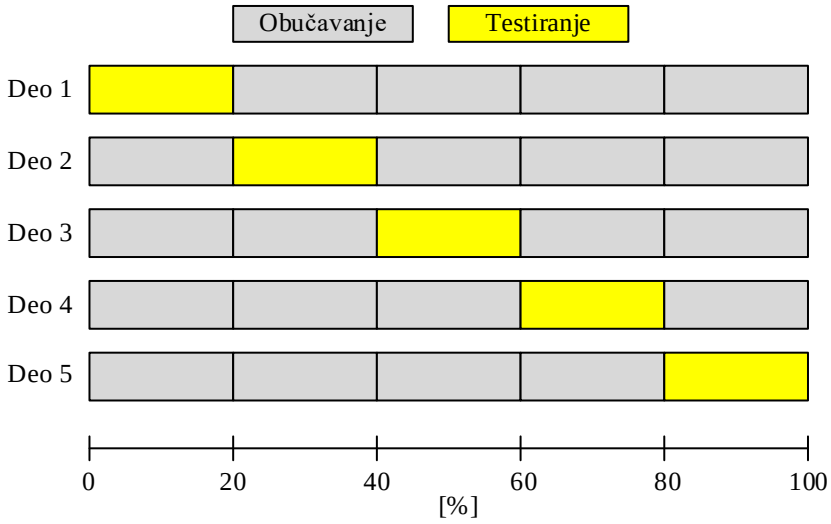
$$L_{Soft\ Dice} = 1 - \frac{2\sum_{i=1}^{n} y_i \hat{y}_i}{\sum_{i=1}^{n} y_i^2 + \sum_{i=1}^{n} \hat{y}_i^2} \quad (10.4)$$

gde je y_i labela, $\hat{y}_i$ verovatnoća predikcije te klase, a n broj piksela u slici ili segmentu. Napomena: za potrebe implementacije se u brojiocu i imeniocu razlomka dodaje mali broj ε kako bi se izbeglo deljenje s nulom.

Proces obučavanja modela se u slučaju primene modela klasičnih neuralnih mreža u tradicionoj radiomici i dubokih neuralnih mreža u dubokoj radiomici uobičajeno sprovodi kroz iteracije tj. epohe. Epoha je jedan kompletan prolaz kroz ceo obučavajući skup podataka i na kraju svake epohe se dobijaju parametri mreže ažurirani tako da funkcija gubitaka bude što manja. Kod dubokih neuralnih mreža, radi efikasnosti i stabilizacije obučavanja, obično se ulazni podaci grupišu u manje podskupove tzv. *batch*-eve pa se ažuriranje parametara mreže radi na celom *batch*-u.

Testiranje. U procesu validacije i testiranja se obučeni model nadgledanog mašinskog učenja primenjuje na validacioni i test skup respektivno. Validacija je opciona i ona služi za procenu generalizacije modela, optimizovanje parametara modela i zaštitu od preobučavanja. Test skup treba da sadrži podatke koje prethodno obučeni model nije ranije „video“ kako bi evaluacija modela bila adekvatna. Ukoliko je skup podataka relativno mali, za procenu performansi modela mašinskog učenja se može koristiti tehnika krosvalidacije (eng. *cross-validation*). Krosvalidacija podrazumeva da se ceo skup podataka podeli na k jednakih delova (eng. *k-fold*). Potom se primeni k iteracija pri čemu

se u svakoj iteraciji jedan od delova skupa koristi kao test skup, a *k-1* delova se koristi kao skup za obučavanje, Sl. 10.3. Metrike za evaluaciju (vidi poglavlje 10.3.1) se saopštavaju kao prosečne vrednosti svih *k* iteracija. Pri podeli na delove, može se voditi računa o tome da svaki deo ima približno isti odnos pripadnika klasa, i ovakav oblik krosvalidacije se naziva stratifikovana (eng. *Stratified*) krosvalidacija. Poseban slučaj krosvalidacije je kada je broj *k* jednak ukupnom broju uzoraka, i ovakva tehnika se naziva krosvalidacija sa izostavljanjem jednog uzorka (eng. *Leave-One-Out*). U biomedicinskim primenama, česta je i upotreba tzv. eksterne validacije, tj. testiranje modela mašinskog učenja na potpuno novom skupu podataka koji nije bio dostupan tokom treniranja ili krosvalidacije kako bi se proverilo koliko dobro model generalizuje podatke.



Slika 10.3. Ilustracija krosvalidacije za *k*=5 (eng. *5-fold cross-validation*)

10.3.1.Metrike za evaluaciju modela

Evaluacija modela nadgledanog mašinskog učenja je važna faza u razvoju modela jer omogućava da se proceni njegova tačnost, pouzdanost i sposobnost da generalizuje na nove podatke koje model prethodno nije „video“. U ovom poglavlju sve metrike će biti definisane za klasifikaciju na dve labelirane klase, pri čemu će sa „+“ biti označena tzv. pozitivna klasa (y=1, ciljna klasa koja treba da se identifikuje ili predvidi), a sa „-“ tzv. negativna klasa (y=0, klasa suprotne kategorije od pozitivne klase).

Tačnost modela (eng. *accuracy*, *Acc*) koji klasifikuje na „+“ i „-“ klasu se definiše kao verovatnoća da klasa bude ispravno klasifikovana (uključujući i pozitivne i negativne slučajeve):

$$Acc = P(+ \mid \text{stvarno } +)P(\text{stvarno } +) + P(- \mid \text{stvarno } -)P(\text{stvarno } -) \qquad (10.5)$$

gde je:

- $P(+|\text{stvarno }+)$ senzitivnost (eng. *sensitivity* ili *recall* ili *true positive rate*, *Sens*), tj. uslovna verovatnoća da će model identifikovati slučaj kao pozitivan pod uslovom da stvarno pripada pozitivnoj klasi
- $P(\text{stvarno }+)$ je verovatnoća da je slučaj stvarno pozitivan što zapravo predstavlja zastupljenost (prevalencu) pozitivne klase u posmatranom uzorku i može se označiti sa zastupljenost_+
- $P(-|\text{stvarno }-)$ specifičnost (eng. *specificity*, *Spec*) tj. uslovna verovatnoća da će model identifikovati slučaj kao negativan pod uslovom da stvarno pripada negativnoj klasi
- $P(\text{stvarno }-)$ je verovatnoća da je slučaj stvarno negativan što zapravo predstavlja zastupljenost negativne klase u posmatranom uzorku tj. $(1-\text{zastupljenost}_+)$.

Relacija (10.5) se može napisati i u sledećem obliku, u skladu sa prethodno uvedenim oznakama za senzitivnost, specifičnost i zastupljenost pozitivne klase:

$$Acc = Sens \cdot zastupljenost_+ + Spec \cdot (1 - zastupljenost_+) \qquad (10.6)$$

Matrica konfuzije koja pokazuje performanse modela mašinskog učenja za dve klase (pozitivnu i negativnu) tj. broj tačnih i pogrešnih predikcija u zavisnosti od stvarne klase je prikazan na Sl. 10.4.

Stvarna pripadnost klasi	Izlaz modela	
	Predikcija: pozitivno	Predikcija: negativno
Stvarna klasa: pozitivno	Tačno pozitivni (*TP*)	Lažno negativni (*FN*)
Stvarna klasa: negativno	Lažno pozitivni (*FP*)	Tačno negativni (*TN*)

Slika 10.4. Matrica konfuzije za model mašinskog učenja sa dve klase

U matrici konfuzije figurišu sledeći brojevi:

- broj tačno pozitivnih slučajeva (eng. *true positive*, *TP* – broj slučajeva koje je model klasifikovao kao pozitivne, a koji su stvarno pozitivni)
- broj lažno pozitivnih slučajeva (eng. *false positive*, *FP* – broj slučajeva koje je model klasifikovao kao pozitivne, a koji su stvarno negativni)
- broj lažno negativnih slučajeva (eng. *false negative*, *FN* – broj slučajeva koje je model klasifikovao kao negativne, a koji su stvarno pozitivni)
- broj tačno negativnih slučajeva (*eng. true negative*, *TN* – broj slučajeva koje je model klasifikovao kao negativne, a koji su stvarno negativni).

Na osnovu uvedenih brojeva *TP*, *FP*, *FN* i *TN* se definišu sledeće metrike:

- Tačnost (*Acc*):

$$A_{cc} = \frac{TP+TN}{TP+TN+FP+FN} \tag{10.7}$$

- Senzitivnost (*Sens*):

$$Sens = \frac{TP}{TP+FN} \tag{10.8}$$

- Specifičnost (*Spec*):

$$Spec = \frac{TN}{TN+FP} \tag{10.9}$$

- Preciznost (eng. *precision, Prec*), tj. tačnost pozitivnih predikacija modela (eng. *Positive Predictive Value, PPV*):

$$Prec = \frac{TP}{TP+FP} \tag{10.10}$$

  Preciznost odgovara uslovnoj verovatnoći da slučaj stvarno pripada pozitivnoj klasi pod uslovom da je model identifikovao slučaj kao pozitivan, *P*(stvarno +|+). Ako nema lažno pozitivnih predikcija tj. *FP*=0, preciznost je jednaka 1 tj. 100%.

- Tačnost negativnih predikacija (eng. *Negative Predictive Value*):

$$NPV = \frac{TN}{TN+FN} \tag{10.11}$$

  NPV odgovara uslovnoj verovatnoći da slučaj stvarno pripada negativnoj klasi pod uslovom da je model identifikovao slučaj kao negativan, *P*(stvarno -|-). Ako nema lažno negativnih predikcija tj. *FN*=0, *NPV* je jednak 1 tj. 100%.

- *F*1 skor (eng. *F1 score*) je metrika koja balansira senzitivnost i preciznost i definiše se kao harmonijska sredina preciznosti i senzitivnosti:

$$F1\ skor = 2 \cdot \frac{Prec \cdot Sens}{Prec + Sens}. \tag{10.12}$$

  Posmatranje ove metrike je od posebnog značaja u slučaju nebalansiranih klasa (npr. kada ima više pozitivnih nego negativnih slučajeva ili obrnuto).

Performanse modela za različite pragove klasifikacije *t* se mogu grafički prikazati pomoću *Receiver Operating Characteristic* (*ROC*) krive. Prag *t* je vrednost s kojom se poredi predikcija verovatnoće slučaja: ako je predikcija verovatnoće da je slučaj pozitivan veća od praga *t*, tj. (*P*(+)>*t*), onda će slučaj biti svrstan u pozitivnu klasu, u suprotnom će biti svrstan u negativnu klasu

(uobičajeni izbor za vrednost praga t u modelima je 0.5). *ROC* kriva predstavlja zavisnost senzitivnosti (stope tačnih pozitivnih predikcija, eng. *true positive rate, TPR*) od stope lažnih pozitivnih predikacija (eng. *false positive rate, FPR*) pri različitim pragovima t. Stopa lažnih pozitivnih predikacija se definiše kao:

$$FPR = \frac{FP}{FP + TN} = 1 - Spec. \tag{10.13}$$

Na osnovu *ROC* krive se može odrediti još jedna performansa modela kao površina ispod *ROC* krive (eng. *Area Under Curve, AUC*) koja pokazuje koliko efikasno model razlikuje klase bez obzira na konkretan prag za binarnu klasifikaciju.

10.3.2. Praktične preporuke

Raspoređivanje uzorka. Prilikom primena metoda mašinskog učenja na podacima pacijenata, za ispravnu evaluaciju performansi modela je izuzetno bitno da se podaci istog pacijenta ne nađu i u skupu za obučavanje i u skupu za validaciju ili testiranje.

Mala baza podataka. Jedno od rešenja u slučaju nedovoljne količine podataka za duboku radiomiku je primena tzv. transfera učenja (eng. *Transfer learning*) tj. korišćenja postojećih arhitektura dubokih mreža za pretreniranje, nakon čega se na konkretnoj bazi podataka vrši fino podešavanje parametara (eng. *fine-tuning*). U medicinskom slikanju se za transfer učenja često koriste sledeće arhitekture *CNN* mreža koje su pretrenirane na velikim skupovima prirodnih slika (npr. *ImageNet*-u [Deng et al. 2009]): *LeNet* [LeCun et al. 1998], *AlexNet* [Krizhevsky et al. 2017], *VGGNet* [Simonyan et al. 2014], *ResNet* [He et al. 2016], *DenseNet* [Huang et al. 2017], *U-Net* [Ronneberger et al. 2015].

Drugo rešenje za nedovoljno velike baze podataka je tzv. augmentacija podataka (eng. *data augmentation*) u kojoj se vrše različite vrste transformacija (rotiranje, transliranje, promena kontrasta i sl. ili kombinacija transformacija) kako bi se baza podataka veštački uvećala.

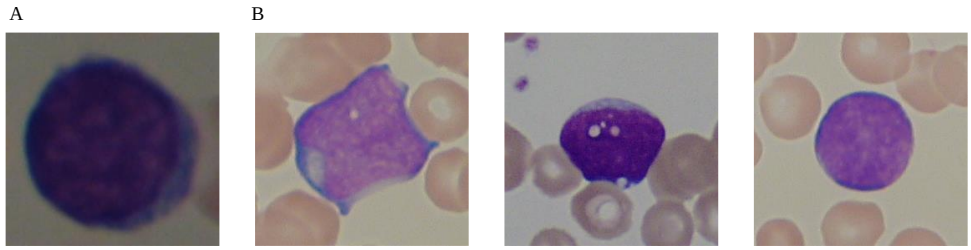
Nebalansirana baza podataka. U slučaju nebalansiranog skupa podataka, preporuka je da se najpre odvoji test skup tako da bude balansiran, potom se odvoji skup za validaciju (ako se koristi), a podaci koji preostanu se koriste za obučavanje.

Zaštita od preobučavanja. Za duboku radiomiku, preporučuje se primena tehnike ranog zaustavljanja (eng. *Early stopping*) radi sprečavanja preobučavanja. Ova tehnika prati ponašanje funkcije gubitaka ili tačnosti na skupu za validaciju i ukoliko se performanse modela ne poboljšavaju kroz određeni broj epoha, treniranje se zaustavlja. Takođe, za duboku radiomiku se

radi zaštite od preobučavanja obično preporučuju tzv. *Dropout* (u potpuno povezanom sloju) ili *Batch* normalizacija (pre aktivacionih funkcija u konvolucionim slojevima ili u potpuno povezanom sloju). *Dropout* tehnika u svakoj iteraciji obučavanja nasumično isključuje (tj. postavlja na nulu) određeni procenat neurona u sloju (npr. za *dropout* parametar 0.3, 30% neurona se isključuje iz svake iteracije nasumično). *Batch* normalizacija na nivou podskupa ulaznih podataka (tj. na nivou *batch*-a) vrši normalizaciju tako što oduzima srednju vrednost unutar *batch*-a i razliku deli sa standardnom devijacijom unutar *batch*-a.

10.3.3. Primer primene tradicionalne radiomike

Na osnovu javno dostupne baze slika ALL-IDB2 koja sadrži slike limfoblasta i zdravih leukocita (korišćene već u Poglavlju 5), Sl. 10.5, izvršena je segmentacija ovih ćelija i za svaku ćeliju je ekstrahovano šest teksturalnih *GLCM* obeležja (kontrast, nesličnost, homogenost, ASM, energija i korelacija) za četiri ugla (0°, 45°, 90° i 135°). Dakle, za svaku sliku segmentirane ćelije su izdvojena ukupno 24 obeležja koja su smeštena u datoteku *GLCM_obelezja.csv*. U prvoj koloni datoteke su labele („1“ za limfoblast, „0“ za zdrav leukocit), u svakoj od narednih kolona je po jedno od *GLCM* obeležja, pri čemu prvi red prikazuje nazive obeležja, a svaki naredni red odgovara po jednoj slici ćelije.



Slika 10.5. Primeri slika iz ALL-IDB2 baze slika: A) zdrav leukocit, B) primeri različitih tipova limfoblasta

Na Pr. 10.1 su prikazani: A) programski kôd *tradicionalna_radiomika.py* koji prepoznaje akutnu limfoblastnu leukemiju na osnovu *GLCM* obeležja i *kNN* klasifikatora i B) rezultat njegovog izvršavanja. U ovom primeru se najpre učitavaju *GLCM* obeležja iz datoteke *GLCM_obelezja.csv* u promenljivu `podaci` pomoću funkcije *Pandas* biblioteke `pd.read_csv` (prvi red datoteke je označen kao argument funkcije *header=0*). Potom se prvih pet redova datoteke prikazuje pomoću instrukcije `podaci.head()`. Potom se iz učitane matrice podataka izdvaja vektor `labela` i matrica `obeležja`.

Pomoću funkcije `train_test_split()` se skup podataka deli na trening skup i test skup. `X_train` je podskup koji čini 80% obeležja, `y_train` je podskup odgovarajućih labela za `X_train`, a `X_test` je podskup koji čini 20% obeležja, `y_test` je podskup odgovarajućih labela za `X_test`. Veličina test skupa je zadata argumentom `test_size=0.2`, pri čemu se vodi računa o jednakoj zastupljenosti klasa u trening i test skupu postavljanjem argumenta `stratify=labela`. Reprodukcija istih rezultata pri svakom pokretanju programa se postiže izborom vrednosti argumenta `random_state` (u ovom primeru je izabrano `random_state=0`).

U narednom koraku je definisan *kNN* klasifikator za pet suseda (`n_neighbors=5`) pomoću funkcije `KNeighborsClassifier()` i treniran na trening skupu pomoću instrukcije `knn_klasif.fit()`. Na `X_test` skupu su određene predikcije `y_pred_test` pomoću `knn_klasif.predict()`. Matrica konfuzije na test skupu i izveštaj performansi klasifikacije su određeni redom pomoću funkcija `confusion_matrix()` i `classification_report()`. Izveštaj klasifikacije obuhvata: 1) preciznost (*precision*), senzitivnost (*recall*) i F1 skor (*f1-score*) za svaku klasu, usrednjeno za sve klase ne uzimajući u obzir veličine klasa (*macro avg*), usrednjeno za sve klase uzimajući u obzir veličinu svake klase (*weighted avg*), 2) tačnost (*accuracy*) i 3) broj stvarnih slučajeva u svakoj klasi (*Support*).

Na Pr. 10.2 su prikazani: A) deo programskog kôda *tradicionalna_radiomika_ROC_stratified.py* koji predstavlja dopunu na prethodni primer Pr. 10.1 i B) rezultat izvršavanja dopunjenog dela kôda. Primer Pr. 10.2 omogućava najpre iscrtavanje ROC krive tj. zavisnosti senzitivnosti (*TPR*) od *FPR* na test skupu pomoću instrukcije `metrics.roc_curve()`. Potom se pomoću funkcje `StratifiedKFold()` primenjuje stratifikovana krosvalidacija za 5 foldova (argument `n_splits=5`). Obezbeđena je nasumična raspodela podataka po foldovima postavljanjem argumenta `shuffle=True`. Reprodukcija istih rezultata pri svakom pokretanju programa se postiže izborom vrednosti argumenta `random_state` (u ovom primeru je izabrano `random_state=42`). Instrukcija `cross_val_score()` je iskorišćena za proračun tačnosti (argument `scoring='accuracy'`) i površine ispod *ROC* krive, *AUC* (argument `scoring='roc_auc'`). Rezultati stratifikovane krosvalidacije su prikazani i kao usrednjene vrednosti za sve foldove za tačnost i za AUC.

Pr. 10.1. Primer *tradicionalna_radiomika.py* – prepoznavanje akutne limfoblastne leukemije na osnovu *GLCM* obeležja i *kNN* klasifikatora

---

A. Programski kôd:

```
import matplotlib.pyplot as plt
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import classification_report
from sklearn.metrics import confusion_matrix

plt.close('all')

# Učitavanje GLCM obeležja
podaci=pd.read_csv ('GLCM_obelezja.csv', header=0)
print(podaci.head())

podaci_matrica = podaci.to_numpy()
labela = podaci_matrica[:, 0]
obelezja = podaci_matrica[:, 1:]

# Podeliti skup na 80% za treniranje i 20% za testiranje
X_train, X_test, y_train, y_test = train_test_split(
    obelezja, labela, test_size=0.2,
    stratify=labela, random_state=0)

# Klasifikacija pomocu knn klasifikatora za 5 suseda
knn_klasif = KNeighborsClassifier(n_neighbors=5)

#Trenirati klasifikator
knn_klasif.fit(X_train, y_train)

# Odrediti predikciju za test skup
y_pred_test = knn_klasif.predict(X_test)
```

```
# Prikazati matricu konfuzije
print ("Matrica konfuzije:")
print (confusion_matrix(y_test, y_pred_test))

# Prikazati izvestaj klasifikacije za test skup
print (classification_report(y_test, y_pred_test))
```

B. Rezultat izvršavanja:

```
   Labela  Ugao 0 Kontrast  ...  Ugao 135 Energija  Ugao 135 Korelacija
0       1         0.660584  ...          -0.368730             0.113021
1       1        -0.328962  ...          -0.246000             0.998400
2       1         0.698269  ...           0.473048             0.520512
3       1         0.675973  ...           1.641500             0.197133
4       1         1.325520  ...           0.677723            -1.178890

[5 rows x 25 columns]
Matrica konfuzije:
[[12  3]
 [ 3 23]]
              precision    recall  f1-score   support

         0.0       0.80      0.80      0.80        15
         1.0       0.88      0.88      0.88        26

    accuracy                           0.85        41
   macro avg       0.84      0.84      0.84        41
weighted avg       0.85      0.85      0.85        41
```

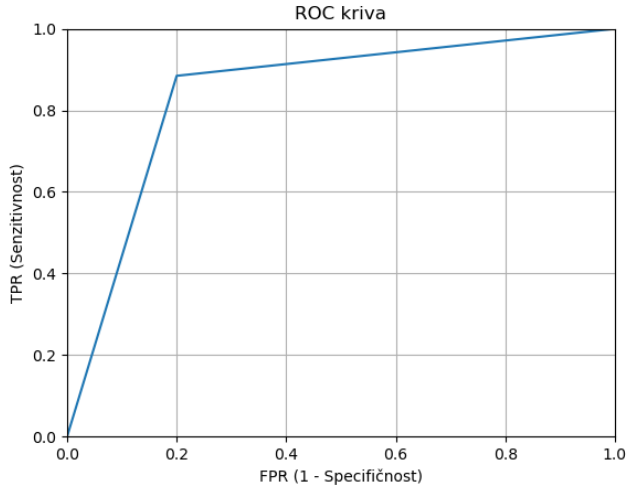
Pr. 10.2. Primer *tradicionalna_radiomika_ROC_stratified.py* – dopuna na primer dat u *tradicionalna_radiomika.py* koja sadrži crtanje ROC krive i procenu tačnosti stratifikovane krosvalidacije.

A. Programski kôd (dopuna na Pr. 10.1.):

```
# Prikazati ROC krivu za test skup
from sklearn import metrics
fpr, tpr, thresholds = metrics.roc_curve(y_test, y_pred_test)
plt.plot(fpr, tpr)
plt.xlim([0.0, 1.0])
plt.ylim([0.0, 1.0])
plt.title('ROC kriva')
plt.xlabel('FPR (1 - Specifičnost)')
plt.ylabel('TPR (Senzitivnost)')
```

```
plt.grid(True)

# Tačnost za stratifikovanu krosvalidaciju sa 5 foldova
from sklearn.model_selection import StratifiedKFold
from sklearn.model_selection import cross_val_score
cv = StratifiedKFold(
    n_splits=5, shuffle=True, random_state=42)
scores_ACC = cross_val_score(
    knn_klasif, obelezja, labela, cv=cv, scoring='accuracy')
print("Tačnosti po foldovima: ", scores_ACC)
print("Srednja tacnost: ", scores_ACC.mean())
scores_AUC = cross_val_score(
    knn_klasif, obelezja, labela, cv=cv, scoring='roc_auc')
print("AUC po foldovima: ", scores_AUC)
print("Srednja AUC: ", scores_AUC.mean())
```

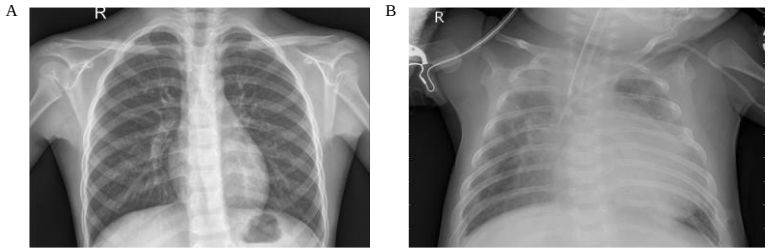
B. Rezultat izvršavanja:




```
Tačnosti po foldovima:  [0.75609756 0.87804878 0.75609756 0.85365854 0.87804878]
Srednja tacnost:  0.8243902439024391
AUC po foldovima:  [0.78333333 0.91794872 0.81025641 0.94487179 0.95125   ]
Srednja AUC:  0.8815320512820513
```

10.3.4. Primer primene duboke radiomike

Na raspolaganju je javno dostupna baza podataka *Pneumonia_Dataset* koja sadrži 11236 rendgenskih slika pluća pacijenata sa pneumonijom i zdravih ispitanika[53]. Obe klase slika su u bazi podataka jednako zastupljene (svaka klasa sadrži po 5618 slika), i dostupne su u .jpeg formatu, Sl. 10.6. Svaka od klasa (*NORMAL* i *PNEUMONIA*) se nalazi u poddirektorijumu čiji naziv odgovara imenu klase, u okviru direktorijuma *Pneumonia_Dataset*.



Slika 10.6. Primeri *Pneumonia_Dataset* rendgenskih slika pluća za: A) zdravog ispitanika, B) pacijenta sa pneumonijom. Slike su podeljene na linku https://www.kaggle.com/datasets/sangeethakallat/chest-radiograph-images-pneumonia-and-normal/data, poslednji pregled Decembar 2024, pod CC BY-NC-SA 4.0 licencom (https://creativecommons.org/licenses/by-nc-sa/4.0/)

Na Pr. 10.3.A1-A5 su prikazani delovi programskog kôda ***duboka_radiomika.py*** koji prepoznaje pneumoniju na bazi *Pneumonia_Dataset* primenom *CNN*, kao i rezultati njihovog izvršavanja koji uključuju prikaz strukture *CNN* modela po slojevima, funkciju gubitaka na trening i validacionom skupu, tačnost na validacionom skupu i matricu konfuzije na validacionom skupu. Na Pr. 10.3.A1, nakon podešavanja putanje za bazu podataka *Pneumonia_Dataset*, vrši se podela slika na trening skup `Xtrain` i validacioni skup `Xval` pomoću instrukcije iz *keras* biblioteke `image_dataset_from_directory()` čiji argumenti su: 1) putanja do baze podataka (`putanja_pneumonia`), 2) tip skupa (argument `subset` je '*training*' ili '*validation*' u zavisnosti od toga da li se radi o skupu za treniranje ili validaciju, respektivno), 3) udeo podataka u validacionom skupu (argument `validation_split` podešen na 20%), dimenzije na koje se svaka slika skalira (argument `image_size` podešen na dimenziju 128x128), broj slika po *batch*-u (argument `batch_size` podešen na 32 slike) i argument `seed` koji omogućava reprodukciju podele na trening i validacioni skup pri svakom pokretanju programa (`seed`=123).

[53] Baza podataka je dostupna na linku: https://www.kaggle.com/datasets/sangeethakallat/chest-radiograph-images-pneumonia-and-normal/data, poslednji pregled Decembar 2024, i podeljena je pod CC BY-NC-SA 4.0 licencom (https://creativecommons.org/licenses/by-nc-sa/4.0/)

Pr. 10.3.A1. Primer *duboka_radiomika.py* – primena *CNN* na detekciju pneumonije pluća – učitavanje baze podataka, podela na trening i validacioni skup.

A. Programski kôd (učitavanje i podela podataka):

```
import numpy as np
import matplotlib.pyplot as plt
import os
from keras.utils import image_dataset_from_directory

#Podešavanje putanje do baze podataka
trenutni_dir= os.getcwd()
print(trenutni_dir)
putanja_pneumonia = os.path.join(
    trenutni_dir, 'Pneumonia_Dataset')

#Podela slika na trening i validacioni skup
velicina_slike = (128,128)
Xtrain = image_dataset_from_directory(
    putanja_pneumonia,
    subset='training',
    validation_split=0.2,
    image_size=velicina_slike,
    batch_size=32,
    seed=123)
Xval = image_dataset_from_directory(
    putanja_pneumonia,
    subset='validation',
    validation_split=0.2,
    image_size=velicina_slike,
    batch_size=32,
    seed=123)

klase = Xtrain.class_names
broj_klasa = len(klase)
```

Na Pr. 10.3.A2 prikazan je deo kôda za definisanje sekvencijalnog modela koji se sastoji iz slojeva za augmentaciju slika. Primenjene su tri tehnike augmentacije: definisan je sloj za nasumično horizontalno flipovanje slike pomoću `layers.RandomFlip` (argument `input_shape` definiše dimenzije ulaznih podataka: 128x128x3, gde je 128x128 veličina na koju su slike skalirane u prethodnom koraku u Pr. 10.3.A1, a 3 odgovara broju od 3 kanala *RGB* slike), potom sloj za nasumičnu rotaciju pomoću `layers.RandomRotation` (zadato je 25% u odnosu na punu rotaciju) i sloj za nasumično zumiranje pomoću `layers.RandomZoom` (primenjeno je uvećavanje do 10%).

Pr. 10.3.A2. Primer *duboka_radiomika.py* – primena *CNN* na detekciju pneumonije pluća – augmentacija slika.

A. Programski kôd (augmentacija slika):

```
#Augmentacija slika
from keras import layers
from keras import Sequential

data_augmentation = Sequential(
  [
    layers.RandomFlip('horizontal',
                      input_shape=(
                          velicina_slike[0],
                          velicina_slike[1], 3)),
    layers.RandomRotation(0.25),
    layers.RandomZoom(0.1),
  ]
)
```

Na Pr. 10.3.A3 prikazan je deo kôda za definisanje *CNN* klasifikacije slika 128x128 sa 3 *RGB* kanala. Dimenzije ulaza 128x128x3 su definisane pomoću `layers.Input()`. Najpre je pozvana sekvenca za augmentaciju slika koja je definisana u Pr. 10.3.A2. Potom se pomoću `layers.Rescaling()` normalizuju pikseli slika (inicijalno dati u opsegu [0-255]) na opseg [0-1]. Nakon augmentacije i normalizacije slika se primenjuju redom sledeći slojevi:

- `layers.Conv2D()` - konvolucioni sloj sa 8 kernela dimenzija 3x3 i *ReLU* aktivacionom funkcijom pri čemu je obezbeđeno da dimenzije izlaza budu iste kao dimenzije ulaza (argument `padding='same'`)
- `layers.MaxPooling2D()` - sloj *max pooling* slažimanja, 2x2
- `layers.Conv2D()` - konvolucioni sloj sa 16 kernela dimenzije 3x3 i *ReLU* aktivacionom funkcijom pri čemu je obezbeđeno da dimenzije izlaza budu iste kao dimenzije ulaza (argument `padding='same'`)
- `layers.MaxPooling2D()` - sloj *max pooling* slažimanja, 2x2
- `layers.Conv2D()` - konvolucioni sloj sa 32 kernela dimenzije 3x3 i *ReLU* aktivacionom funkcijom pri čemu je obezbeđeno da dimenzije izlaza budu iste kao dimenzije ulaza (argument `padding='same'`)
- `layers.MaxPooling2D()` - sloj *max pooling* slažimanja, 2x2
- `layers.Conv2D()` - konvolucioni sloj sa 64 kernela dimenzije 3x3 i *ReLU* aktivacionom funkcijom pri čemu je obezbeđeno da dimenzije izlaza budu iste kao dimenzije ulaza (argument `padding='same'`)
- `layers.Dropout()` - *Dropout* sloj sa parametrom 0.5, 50% neurona se isključuje iz svake iteracije nasumično
- `layers.Flatten()` - sloj za vektorizaciju koji transformiše višedimenzioni izlaz u jednodimenzioni vektor
- `layers.Dense()` - potpuno povezani sloj sa 120 neurona i *ReLU* aktivacionom funkcijom
- `layers.Dense()` - potpuno povezani sloj sa brojem neurona jednakim broju klasa (2 neurona s obzirom na to da je `broj_klasa=2`) i *softmax* aktivacionom funkcijom čiji izlaz daje verovatnoću svake klase.

Pr. 10.3.A3. Primer *duboka_radiomika.py* – primena *CNN* na detekciju pneumonije pluća – definisanje CNN arhitekture.

A. Programski kôd (definisanje CNN arhitekture):

```
#Definisanje CNN arhitekture
model = Sequential([
```

```
    layers.Input(shape=(128,128, 3)),
    data_augmentation,
    layers.Rescaling(1./255),
    layers.Conv2D(8, 3, padding='same', activation='relu'),
    layers.MaxPooling2D(),
    layers.Conv2D(16, 3, padding='same', activation='relu'),
    layers.MaxPooling2D(),
    layers.Conv2D(32, 3, padding='same', activation='relu'),
    layers.MaxPooling2D(),
    layers.Conv2D(64, 3, padding='same', activation='relu'),
    layers.Dropout(0.5),
    layers.Flatten(),
    layers.Dense(120, activation='relu'),
    layers.Dense(broj_klasa, activation='softmax')
])

model.summary()
```

B. Rezultat izvršavanja:

Layer (type)	Output Shape	Param #
sequential_31 (Sequential)	(None, 128, 128, 3)	0
rescaling_17 (Rescaling)	(None, 128, 128, 3)	0
conv2d_68 (Conv2D)	(None, 128, 128, 8)	224
max_pooling2d_51 (MaxPooling2D)	(None, 64, 64, 8)	0
conv2d_69 (Conv2D)	(None, 64, 64, 16)	1,168
max_pooling2d_52 (MaxPooling2D)	(None, 32, 32, 16)	0
conv2d_70 (Conv2D)	(None, 32, 32, 32)	4,640
max_pooling2d_53 (MaxPooling2D)	(None, 16, 16, 32)	0
conv2d_71 (Conv2D)	(None, 16, 16, 64)	18,496
dropout_17 (Dropout)	(None, 16, 16, 64)	0
flatten_17 (Flatten)	(None, 16384)	0
dense_32 (Dense)	(None, 120)	1,966,200
dense_33 (Dense)	(None, 2)	242

```
Total params: 1,990,970 (7.59 MB)
Trainable params: 1,990,970 (7.59 MB)
Non-trainable params: 0 (0.00 B)
```

Na Pr. 10.3.A4 prikazan je deo kôda koji optimizuje i trenira *CNN* model pomoću `model.compile()` i `model.fit()` instrukcija respektivno. Za optimizaciju parametara modela se koristi najčešće korišćeni *Adaptive Moment Estimation* (*Adam*) optimizator sa podešenom brzinom učenja (veličinom koraka za jedan prolaz kroz model) na 0.001 (argument `learning_rate`). Za funkciju gubitaka je iskorišćena `sparse_categorical_crossentropy`[54] koja se koristi kada su oznake klasa celobrojne vrednosti. Tokom procesa treniranja je izabrano da se prati i tačnost modela (`metrics=['accuracy']`). Za zaštitu od preobučavanja je izabrano rano zaustavljanje pomoću `EarlyStopping()` instrukcije koja prati vrednost gubitaka (argument `monitor='val_loss'`) i očekuje da se `val_loss` minimizira (argument `mode='min'`). Treniranje se zaustavlja ako nakon pet epoha nema poboljšanja (`patience=5`) i tada se težine modela vraćaju na one koje su bile sa minimalnom vrednosti `val_loss` (argument `restore_best_weights=True`). Sam proces treniranja na `Xtrain` skupu predviđa maksimalno 50 epoha (argument `epochs=50`) u *batch*-evima od po 10 slika (`batch_size=10`), s tim što se proces usled primene tehnike ranog zaustavljanja (`callbacks=[es]`) može prekinuti i ranije. Argument `verbose` je postavljen na 1 kako bi se tokom procesa treniranja dobijale povratne informacije o gubitku i tačnosti na kraju svake epohe uz prikaz progres bara (za vrednost 0 nema ispisa povratnih informacija, a za vrednost 2 se povratne informacije ispisuju, ali bez progres bara. Funkcija gubitaka se posmatra i na trening `Xtrain` i na validacionom `Xval` skupu i prikazuje se grafički po epohama.

Pr. 10.3.A4. Primer *duboka_radiomika.py* – primena *CNN* na detekciju pneumonije pluća – optimizacija, rano zaustavljanje, treniranje modela, validacija.

A. Programski kôd (optimizacija, rano zaustavljanje, treniranje modela, validacija):

```
from keras.optimizers import Adam
model.compile(optimizer=Adam(learning_rate=0.001),
              loss='sparse_categorical_crossentropy',
              metrics=['accuracy'])
```

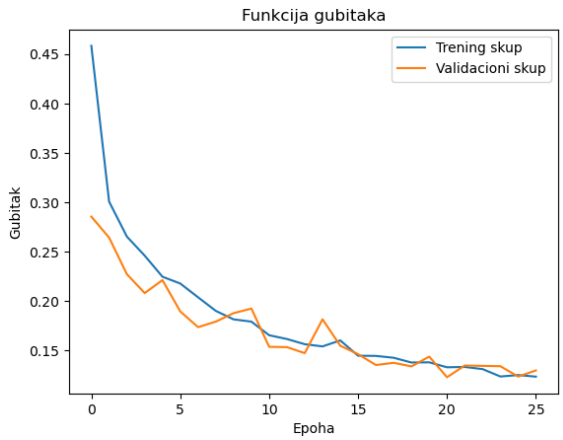
[54] *Sparse categorical* krosentropijska funkcija gubitaka se računa kao:

$$L_{SCCE} = -\log\left(\hat{y}_{stvarna\_labela}\right)$$

gde je $\hat{y}_{stvarna\_labela}$ verovatnoća koju model pridružuje stvarnoj klasi.

```
from keras.callbacks import EarlyStopping
es = EarlyStopping(monitor='val_loss', mode='min',
                   patience=5, restore_best_weights=True)

history = model.fit(Xtrain,
                    epochs=50,
                    batch_size=10,
                    validation_data=Xval,
                    callbacks=[es],
                    verbose=1)

trening_loss = history.history['loss']
validacija_loss = history.history['val_loss']

plt.figure()
plt.xlabel('Epoha')
plt.ylabel('Gubitak')
plt.plot(trening_loss, label='Trening skup')
plt.plot(validacija_loss, label='Validacioni skup')
plt.title('Funkcija gubitaka')
plt.legend()
plt.show()
```

B. Rezultat izvršavanja:



Na Pr. 10.3.A5 prikazan je ostatak kôda koji određuje tačnost CNN modela na validacionom skupu i matricu konfuzije na validacionom skupu na osnovu labela i predikcija. Za svaki `img` ulaz iz `Xval` skupa, instrukcija `model.predict(img, verbose=0)` pravi predikciju rezultujući nizom verovatnoća za svaku klasu (verovatnoće da ulaz pripada određenoj klasi). Potom `np.argmax()` po redovima (argument `axis=1`) traži indeks klase sa najvećom verovatnoćom za svaki ulaz `img` i pronađeni indeks klase se smešta u vektor `predikcije` pomoću `np.append()`. Na osnovu nizova `labele` i `predikcije` se primenom `accuracy_score()` određuje tačnost na validacionom skupu, a potom se izračunavaju normalizovane (argument `normalize=True`) vrednosti matrice konfuzije `mk` pomoću `confusion_matrix()` i grafički se prikazuju pomoću `ConfusionMatrixDisplay()`.

Pr. 10.3.A5. Primer *duboka_radiomika.py* – primena *CNN* na detekciju pneumonije pluća – rezultati na validacionom skupu.

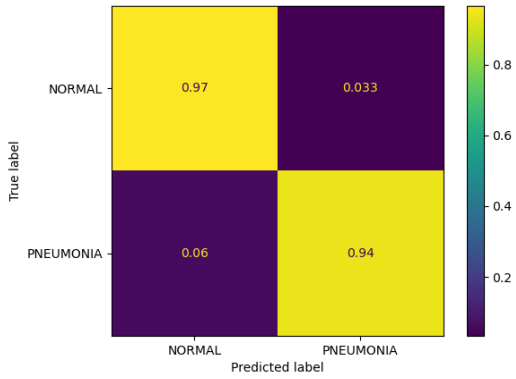
A. Programski kôd (rezultati na validacionom skupu):

```
# Prikaz rezultata na validacionom skupu
labele = np.array([])
predikcije = np.array([])

for img, lab in Xval:
    labele = np.append(labele, lab)
    pred=np.argmax(model.predict(img, verbose=0), axis=1)
    predikcije = np.append(predikcije, pred)

from sklearn.metrics import accuracy_score
tacnost=100*accuracy_score(labele, predikcije)
print('Tačnost modela je: ' + str(tacnost) + '%')

from sklearn.metrics import confusion_matrix
from sklearn.metrics import ConfusionMatrixDisplay
mk = confusion_matrix(labele, predikcije, normalize='true')
mk_prikaz = ConfusionMatrixDisplay(
    confusion_matrix=mk, display_labels=klase)
mk_prikaz.plot()
```

```
plt.title('Matrica konfuzije')
plt.show()
```

B. Rezultat izvršavanja:

```
Tačnost modela je: 95.37160658655985%
```



10.4. Rezime poglavlja

- Tradicionalna radiomika ekstrahuje kvantitativna obeležja (oblik, tekstura, intenzitet) iz biomedicinskih slika ili regiona/zapremina od interesa i potom koristi metode statističke analize i klasične modele mašinskog učenja (logistička regresija, k najbližih suseda, model slučajnih šuma, model potpornih vektora, neuralne mreže i sl.) za klasifikaciju izdvojenih obeležja.
- Duboka radiomika koristi duboke neuralne mreže, pre svega konvolucione neuralne mreže, za automatsku ekstrakciju i analizu karakteristika biomedicinskih slika, bez manuelne selekcije.
- Standardni elementi konvulucionih neuralnih mreža su: ulazni sloj tj. slika, konvolucioni sloj, sloj sažimanja, potpuno povezani sloj i izlazni sloj.
- Razvoj modela nadgledanog mašinskog učenja podrazumeva da se na skupu za obučavanje podešavaju parametri modela kako bi performanse modela bile što bolje, tj. funkcija gubitaka koja odražava stepen neslaganja predikcije modela i labele što manja.
- U slučaju nebalansiranih skupova podataka (kada nisu sve klase jednako zastupljene), neophodno je uzeti u obzir težinske faktore svake klase. U slučaju primene modela nadgledanog mašinskog učenja za proces

segmentacije slika, često se koristi funkcija gubitaka bazirana na *Dice* metrici.

- Ukoliko je skup podataka relativno mali, za procenu performansi modela mašinskog učenja se može koristiti tehnika krosvalidacije.
- Standardne metrike za evaluaciju klasifikacionih modela nadgledanog mašinskog učenja su: tačnost, senzitivnost, specifičnost, preciznost, tačnost negativnih predikacija, F1 skor.
- Performanse modela za različite pragove klasifikacije se mogu grafički prikazati pomoću *Receiver Operating Characteristic* krive. Kao standarna metrika za procenu performansi modela koristi se površina ispod *ROC* krive.
- Praktične preporuke za primenu nadgledanog mašinskog učenja: 1) podaci istog pacijenta ne treba da se nađu i u skupu za obučavanje i u skupu za validaciju ili testiranje, 2) u slučaju malih baza se primenjuje augmentacija podataka i/ili transfer učenje (korišćenje postojećih arhitektura dubokih mreža za pretreniranje, a potom se vrši fino podešavanje parametara na konkretnoj bazi podataka), 3) rano zaustavljanje (*Early Stopping*), *Dropout* i *Batch* normalizacija su tehnike koje štite model od preobučavanja.

LITERATURA

[1] Badža Atanasijević M., Radović T., Janković M. M., Barjaktarović M., Open-Source Application for Mri and Ct Registration Using Homography Transformation, Proceedings of the 9th International Conference on Bioinformatics Research and Applications (ICBRA), 18-20 September, Berlin, Germany, pp. 111 - 115, 2022.

[2] Bankman I.N., Handbook of Medical Image Processing and Analysis, second edition, Elsevier Inc., Amsterdam, Netherlands, 2009.

[3] Besl, P.J. and McKay, N.D., Method for registration of 3-D shapes, SPIE, Sensor fusion IV: control paradigms and data structures, vol. 1611, pp. 586-606, 1992.

[4] Birkfellner W., Applied Medical Image Processing, CRC Press, Taylor and Francis Group, Boca Raton, USA, 2011.

[5] Bishop C., Pattern Recognition and Machine Learning, Springer, New York, 2007.

[6] Buades, A., Coll, B., Morel, J.M., A non-local algorithm for image denoising, Proc. of the IEEE computer society conference on computer vision and pattern recognition (CVPR'05), vol. 2, pp. 60-65, 2005.

[7] Bushberg J.T., Seibert J.A., Leidholdt Jr E.M., Boone J.M., The Essential Physics of Medical Imaging, 4th Edition, Wolters Kluwer, Amsterdam, Netherlands, 2020.

[8] Ceder N., The Quick Python Book, Third Edition, Manning, New York, USA, 2018.

[9] Chityala R., Pudipeddi S., Image Processing and Acuisition using Python, Second Edition, CRC Press, Boca Raton, USA, 2021.

[10] Coelho L. P., Shariff A., and Murphy R. F., Nuclei Segmentation In Microscope Cell Images: A Hand-Segmented Dataset And Comparison Of Algorithms, June 28-July 1, Boston, Massachusetts, USA, pp. 518-521, 2009.

[11] Proceedings of the 2009 IEEE International Symposium on Biomedical Imaging (ISBI 2009), pp. 518-521.

[12] Cooley J.W., Tukey J.W., An Algorithm for the machine calculation of complex Fourier series, Mathematics of Computation, vol. 19, pp. 297-301, 1965.

[13] Crum, W.R., Hartkens, T. and Hill, D.L.G., Non-rigid image registration: theory and practice, The British journal of radiology, vol. 77(suppl_2), pp.S140-S153, 2004.

[14] Čutura D., Đerić N., Analiza radiografije grudnog koša – prepoznavanje upale pluća, seminarski rad, Univerzitet u Beogradu – Elektrotehnički fakultet, 2024.

[15] Deng J., Dong W., Socher R., Li L.J., Li K. and Fei-Fei L., Imagenet: A large-scale hierarchical image database, IEEE conference on computer vision and pattern recognition, 20-25 June, Miami, Florida, pp. 248-255, 2009.

[16] Đulinac A., Stamatović L., Ilić N., „Segmentacija aorte na CT snimcima abdomena primenom Hafove transformacije“, seminarski rad, Univerzitet u Beogradu – Elektrotehnički fakultet, 2024.

[17] Dhawan A, Medical Image Analysis, second edition, A John Wiley & Sons, INC, New York, USA, 2011.

[18] Downey A.B., Think Python: How to Think Like a Computer Scientist, 3rd Edition, O'Reilly Media, Sebastopol, USA, 2024.

[19] Gonzalez R.C., Woods, R.E., Digital Image Processing, 3rd Edition, Pearson Education, New Jersey, USA, 2008.

[20] Goodfellow I., Bengio Y., Courville A., Deep learning, MIT press, Cambridge, 2016.

[21] Hastie T., Tibshirani R., Friedman J., The Elements of Statistical Learning: Data Mining, Inference, and Prediction, Springer, New York, 2008.

[22] He K., Zhang X., Ren S., Sun J., Deep residual learning for image recognition, Proc. of the IEEE Conference on Computer Vision and Pattern Recognition, Las Vegas, USA, 27–30 June, pp. 770–778, 2016.

[23] Huang G., Liu Z., Van Der Maaten L., Weinberger K.Q.. Densely connected convolutional networks. In Proc. of the IEEE Conference on Computer Vision and Pattern Recognition, Honolulu, USA, 21–26 July, pp. 4700–4708, 2017.

[24] Hunter, J. D., Matplotlib: A 2D Graphics Environment, Computing in Science & Engineering 9, no. 3, pp. 90–95, 2007.

[25] Kass M., Witkin A., and Terzopoulos D.. "Snakes: Active contour models." International journal of computer vision 1, no. 4, pp. 321-331, 1988.

[26] Keys R.G., Cubic Convolution Interpolation for Digital Image Processing," IEEE Transactions on Acoustics, Speech, and Signal Processing, vol. 29, No. 6, 1981.

[27] Krizhevsky, A., Sutskever, I., Hinton, G.E., ImageNet classification with deep convolutional neural networks, Communications of the ACM, 60(6), pp.84-90, 2017.

[28] Kucharsky H.R., Alter R.A., Sojoudi S., Ardekani, B., Kuzniecky R., Pardoe H.R., Corpus callosum area and brain volume in autism spectrum disorder: quantitative analysis of structural MRI from the ABIDE database, Journal of Autism and Developmental Disorders, vol. 45, no. 10, pp. 3107-3114, 2015.

[29] Labati, R.D., Piuri, V., Scotti, F., ALL-IDB: the acute lymphoblastic leukemia image database for image processing, Proc. of the 2011 IEEE Int. Conf. on Image Processing (ICIP 2011), Brussels, Belgium, September 11-14, pp. 2045-2048, 2011.

[30] LeCun, Y., Bottou L., Bengio Y., Haffner P., Gradient-based learning applied to document recognition, Proceedings of the IEEE 86(11), pp. 2278-2324, 1998.

[31] Lorensen, W.E. and Cline, H.E., Marching cubes: A high resolution 3D surface construction algorithm, ACM SIGGRAPH Computer Graphics, vol. 21, no. 4, pp. 163–169, 1987.

[32] Milić Lj., Dobrosavljević Z., Ćertić J., Uvod u digitalnu obradu signala, Akademska misao, Beograd, 2015.

[33] Ming, Y., Wu, N., Qian, T., Li, X., Wan, D.Q., Li, C., Li, Y., Wu, Z., Wang, X., Liu, J. and Wu, N., Progress and future trends in PET/CT and PET/MRI molecular imaging approaches for breast cancer, Frontiers in oncology, 10, p.1301, 2020.

[34] Mladenović K., „Korekcija pomeraja na renogramskim snimcima i kvantitativna analiza korigovanih renograma“, Univerzitet u Beogradu – Elektrotehnički fakultet, 2024.

[35] Paris, S., Kornprobst, P., Tumblin, J., Durand, F., Bilateral filtering: Theory and applications, Foundations and Trends, Now Publishers, Hanover, Massachusetts, USA, 2009.

[36] Pešić I., Segmentacija i klasifikacija slika leukocita za detekciju akutne limfoblastne leukemije, diplomski rad, Univerzitet u Beogradu – Elektrotehnički fakultet, 2019.

[37] Piuri, V., Scotti, F., Morphological classification of blood leucocytes by microscope images, Proc. of the 2004 IEEE Int. Conf. on

Computational Intelligence for Measurement Systems and Applications (CIMSA 2004), Boston, MA, USA, July 12-14, pp. 103-108, 2004.

[38] Popović D., Medicinska instrumentacija i merenja, Akademska misao, Beograd, Srbija, 2014.

[39] Proakis J., Manolakis D., Digital Signal Processing: Principles, Algorithms, and Applications, Prentice Hall, 2006.

[40] Rangaraj R., Biomedical Image Analysis, CRC Press LLC, Boca Raton, USA, 2005.

[41] Ronneberger O., Fischer P., Brox T., 2015. U-net: Convolutional networks for biomedical image segmentation. Proc. of Medical image computing and computer-assisted intervention–MICCAI 2015: 18th international conference, Munich, Germany, October 5-9, part III 18, pp. 234-241, 2015.

[42] Russel S., Norvig P., Artificial Intelligence: A Modern Approach, Pearson, London, 2010.

[43] Scotti, F., Automatic morphological analysis for acute leukemia identification in peripheral blood microscope images, Proc. of the 2005 IEEE Int. Conf. on Computational Intelligence for Measurement Systems and Applications (CIMSA 2005), Giardini Naxos - Taormina, Italy, July 20-22, pp. 96-101, 2005.

[44] Scotti, F., Robust Segmentation and Measurements Techniques of White Cells in Blood Microscope Images, Proc. of the 2006 IEEE Instrumentation and Measurement Technology Conf. (IMTC 2006), Sorrento, Italy, April 24-27, pp. 43-48, 2006.

[45] Semmlow J.L., Griffel B., Biosignal and Medical Image Processing, CRC Press, Taylor and Francis Group, Boca Raton, USA, third edition, 2014.

[46] Serway R.A., Jewett J.W., Physics for Scientists and Engineers, 6th Edition, Thomson Brooks/Cole, Belmont, USA, 2004.

[47] Simonyan, K., Zisserman, A., Very deep convolutional networks for large-scale image recognition. arXiv preprint arXiv:1409.1556, 2014.

[48] Spanhol F.A., Oliveira L.S., Petitjean C., Heutte L., A Dataset for Breast Cancer Histopathological Image Classification, IEEE Transactions on Biomedical Engineering, vol. 63, no. 7, pp. 1455-1462, July 2016.

[49] Tondeur M, Nogarede C, Donoso G, Piepsz A. Inter- and intra-observer reproducibility of quantitative renographic parameters of differential function and renal drainage in children. Scandinavian Journal of Clinical and Laboratory Investigation, 73(5), pp. 414-421, 2013.

[50] Van Timmeren, J.E., Cester, D., Tanadini-Lang, S., Alkadhi, H. and Baessler, B., Radiomics in medical imaging—"how-to" guide and critical reflection. Insights into imaging, 11(1), p.91, 2020.

[51] Vujisić M., Primena zračenja u medicini: medicinsko slikanje, radioterapija i radiohirurgija, autorizovane beleške sa predavanja, Univerzitet u Beogradu – Elektrotehnički fakultet, 2019.

[52] Yaniv Z., Lowekamp B. C., Johnson H. J., Beare R., SimpleITK Image-Analysis Notebooks: a Collaborative Environment for Education and Reproducible Research, Journal of Digital Imaging, vol. 31, no. 3, pp. 290-303, 2018.

[53] Zhang A., Lipton Z.C., Li M., Smola A.J., Dive into Deep Learning, Cambridge University Press, Cambridge, 2023.

[54] Zhang, H. and Qie, Y., Applying deep learning to medical imaging: a review. Applied Sciences, 13(18), p.10521, 2023.

[55] Webb A., Introduction to Biomedical Imaging, 2nd Edition, Wiley-IEEE Press, Hoboken, New Jersey, USA, 2022.

SPISAK SKRAĆENICA

ACR	eng. *American College of Radiology*
ASM	Ugaoni moment drugog reda (eng. *Angular Second Moment*)
BMP	eng. *BitMaP*
CAD	Kompjuterski podržana dijagnostika (eng. *Computer-Aided Diagnosis*)
CLAHE	Adaptivna ekvalizacija histograma (eng. *Contrast Limited Adaptive Histogram Equalization*)
CNN	Konvolucione neuralne mreže (eng. *Convolutional Neural Networks*)
CT	Kompjuterizovana tomografija (eng. *Computerized Tomography*)
DCT	eng. *Discrete Cosine Transform*
DICOM	eng. *Digital Imaging and Communication in Medicine*
DFT	Diskretna Furijeova transformacija (eng. *Discrete Fourier Transform*)
DVR	Direktno zapreminsko renderovanje (eng. *Direct Volume Rendering*)
DWT	eng. *Discrete Wavelet Transform*
FCNN	Potpuno povezana neuralna mreža (eng. *Fully Connected Neural Neutwork*)
FFT	Brza Furijeova transformacija (eng. *Fast Fourier Transform*)
FN	Lažno negativni (eng. *false negative*)
FP	Lažno pozitivni (eng. *false positive*)
FPR	Stopa lažnih pozitivnih predikcija (eng. *false positive rate*)
GLCM	Matrica ko-pojavljivanja nivoa sivog (eng. *Gray Level Co-occurrence Matrix*)
HU	Haunsfildova jedinica (eng. *Hounsfield Unit*)
IBSI	Inicijativa za standardizaciju biomarkera (eng. *Image Biomarker Standardization Initiative*)
IDFT	Inverzna diskretna Furijeova transformacija (eng. *Inverse Discrete Fourier Transform*)
ITF	Funkcija transformacije intenziteta (eng. *Intensity Transform Function, ITF*)
JPEG	eng. *Joint Photographic Experts Group*
kNN	k najbližih suseda (eng. *k-Nearest Neighbors*)

LBP	Lokalni binarni obrazac (eng. *Local Binary Pattern*)
LR	Logistička regresija (eng. *Logistic Regression*)
MHA/MHD	eng. *MetaImage Header/MetaImage Data*
MI	Međusobna informacija (eng. *Mutual Information*)
MIP	Projekcija maksimalnih intenziteta (eng. *Maximum Intensity Projection*)
MRI	Magnetna rezonanca (eng. *Magnetic Resonance Imaging*
NEMA	eng. *National Electrical Manufacturers Association*
NRRD	eng. *Nearly Raw Raster Data*
NIfTI	eng. *Neuroimaging Informatics Technology Initiative*
NIRS	Spektroskopija u bliskom infracrvenom delu spektra (eng. *Near-Infrared Spectroscopy*)
NPV	Tačnost negativnih predikacija (eng. *Negative Predictive Value*)
NTSC	*National Television System Committee*
OCT	Optička koherentna tomografija (eng. *Optical Coherence Tomography*)
PET	Pozitronska emisiona tomografija (eng. *Positron Emission Tomography*)
PGM	eng. *Portable Gray Map*
PNG	eng. *Portable Network Graphics*
PPV	Tačnost pozitivnih predikacija (eng. *Positive Predictive Value*)
RF	Model slučajnih šuma (eng. *Random Forest*)
RLE	eng. *Run Length Encoding*
ROC	eng. *Receiver Operating Characteristic*
ROI	Region od interesa (eng. *region of interest*)
SIP	Projekcija sumiranih intenziteta (eng. *Summed Intensity Projection*)
SNR	Odnos signal/šum (eng. *Signal-to-Noise Ratio*)
SVM	Model potpornih vektora (eng. *Support Vector Machine*)
SPECT	Jednofotonska emisiona kompjuterska tomografija (eng. *Single-photon emission computed tomography*)
TIFF	eng. *Tagged Image File Format*
TN	Tačno negativni (eng. *true negative*)
TP	Tačno pozitivni (eng. *true positive*)
TPR	Stopa tačnih pozitivnih predikacija (eng. *true positive rate*)

SPISAK SLIKA

SPISAK TABELA

SPISAK PROGRAMSKIH KÔDOVA

A. PREGLED ALATA

Tabela A.1 daje pregled često korišćenih alata koji su dostupni ili potpuno besplatno ili na probni period (eng. *trial version*) ili imaju besplatno dostupnu verziju sa ograničenim mogućnostima (eng. *lite*).

Tabela A.1. Pregled često korišćenih besplatnih alata: B – besplatno dostupno, P – probno dostupno, O – besplatno dostupno s ograničenim mogućnostima. Poslednji pregled linkova Decembar 2024

NAZIV	NAMENA I LINK	B/P/O
ezDICOM	Čitač medicinskih slika https://ezdicom.sourceforge.net/	B
RadiAnt	Čitač medicinskih slika, osnovna analiza, 3D prikaz https://www.radiantviewer.com/	P
Weasis	Čitač medicinskih slika, osnovna analiza, 3D prikaz, PACS integracija https://weasis.org/en/index.html	B
ClearCanvas	Čitač medicinskih slika, PACS integracija https://clearcanvas.github.io	B
Osirix	Čitač medicinskih slika, osnovna analiza, 3D prikaz, PACS integracija (samo za macOS) https://www.osirix-viewer.com	P/O
ImageJ	Čitač biomedicinskih slika, analiza, 3D prikaz, mogućnosti za dodatne funkcionalnosti kroz *Plugin*-ove https://imagej.net	B
3D Slicer	Čitač medicinskih slika, analiza, 3D prikaz, segmentacija, kvantifikacija https://www.slicer.org	B
ITK-SNAP	Manuelna segmentacija 3D slika, analiza volumena, poluautomatizovana segmentacija http://www.itksnap.org/pmwiki/pmwiki.php	B
Mazda	Analiza teksture biomedicinskih slika https://qmazda.p.lodz.pl	B
LifeX	Izdvajanje *radiomics* obeležja https://www.lifexsoft.org	B
FreeSurfer	Analiza MRI snimaka mozga,segmentacija korteksa https://surfer.nmr.mgh.harvard.edu	B
SPM	Statistička parametarska analiza snimaka mozga https://www.fil.ion.ucl.ac.uk/spm	B
FSL	Analiza snimaka mozga https://fsl.fmrib.ox.ac.uk/fsl/docs/#	B
AFNI	Analiza snimaka mozga https://afni.nimh.nih.gov	B
DSI Studio	Difuziona MRI analiza, vizuelizacija traktografije https://dsi-studio.labsolver.org	B

CIP - Каталогизација у публикацији Народна библиотека Србије, Београд

57+61]:004.932(078.5)(0.034.2)

ЈАНКОВИЋ, Милица, 1979-

Analiza biomedicinske slike [Elektronski izvor] : [elektronski udžbenik] / Milica M. Janković. - Beograd : Univerzitet, Elektrotehnički fakultet, 2025 (Beograd : M. Janković). – 1 elektronski optički disk (CD-ROM) ; 12 cm

Sistemski zahtevi: Nisu navedeni. - Nasl. sa naslovne strane dokumenta. - Tiraž 50. - Sadrži bibliografiju.

ISBN 978-86-7225-099-2

а) Биомедицина -- Информациона технологија -- Анализа слике

COBISS.SR-ID 163875337