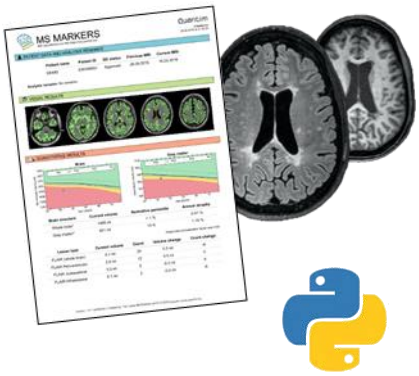
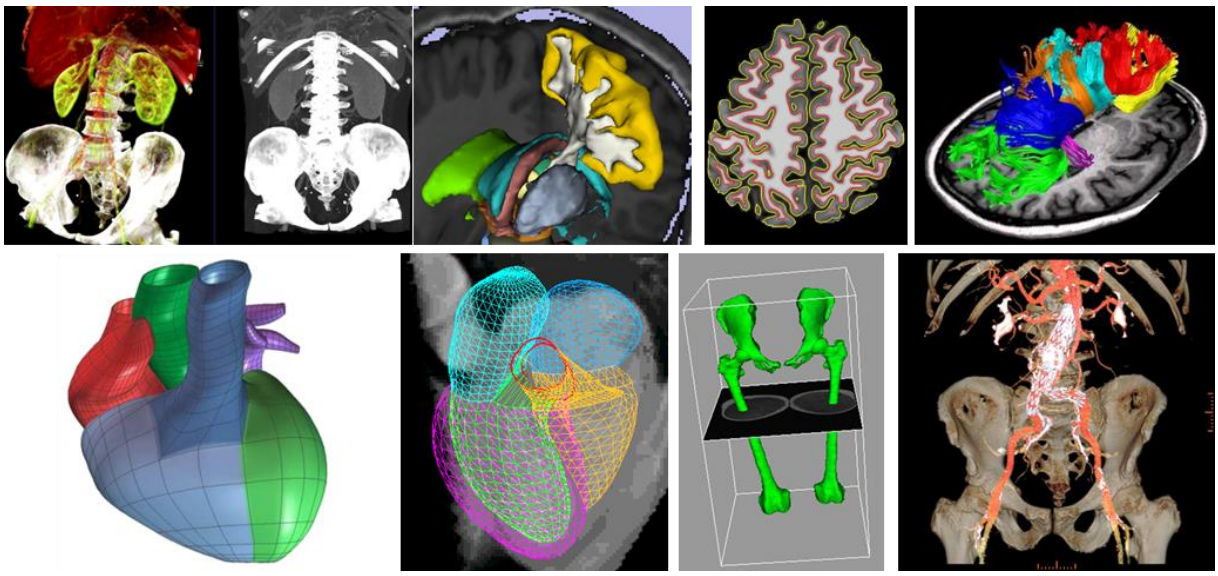
Univerza v Ljubljani
Fakulteta za elektrotehniko

Žiga Špiclin



Analiza medicinskih slik

Laboratorijske vaje v programskem jeziku Python



Univerza *v Ljubljani*
Fakulteta *za elektrotehniko*

Žiga Špiclin

Analiza medicinskih slik

Laboratorijske vaje v programskem jeziku Python

Ljubljana, 2020

Predgovor

Pričujoča zbirka nalog predstavlja dopolnilno študijsko gradivo pri predmetu Analiza medicinskih slik na Univerzitetnem študiju elektrotehnike 2. stopnje, smer Biomedicinska tehnika. Nastala je iz gradiv za laboratorijskih vaj pri tem predmetu v preteklih študijskih letih.

Namen gradiva je seznaniti študente z navodili laboratorijskih vaj in podati smernice za njihovo izvedbo. Zbirka nalog obsega 9 vaj, ki študente seznanijo z uporabo Python programskega jezika in knjižnice SimpleITK s področja analize medicinskih slik, s postopki filtriranja šuma na osnovi anizotropne difuzije, z uporabo kubičnih B-zlepkov za definicijo zvezne preslikave in njene uporabe za netogo poravnavo slik, z uporabo programa BrainSeg3D za interaktivno vizualizacijo in obdelavo ter analizo 3D medicinskih slik, s postopkom ustvarjanja projekcije 3D slike in algoritma poravnave 3D in 2D slik v kontekstu slikovnega vodenja posegov ter njegovo validacijo, razgradnjo slik z interaktivnim in avtomatskim Otsu upragovanjem, razgradnjo s poravnavo atlasov in rojenjem in s postopki vrednotenja kvantitativnih slikovnih biomarkerjev.

Avtor se zahvaljuje vsem sodelavcem Laboratorija slikovne tehnologije na Fakulteti za elektrotehniko, Univeze v Ljubljani, ki so kakorkoli pripomogli k nastanku te zbirke.

Ljubljana, Maj 2020 Žiga Špiclin

Kazalo

Univerza v Ljubljani, Fakulteta za elektrotehniko
Analiza medicinskih slik (AMS)

Uvod v Python in SimpleITK

Python in digitalne slike

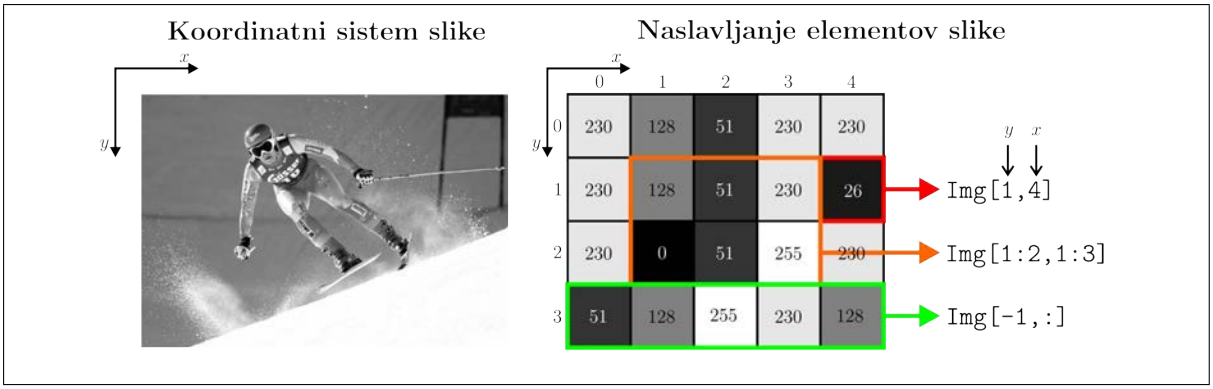
Python je odprtokodni, enostaven in zelo priljubljen interpreterski jezik. Na vajah bomo primere pripravili v programskem okolju WinPython (`http://winpython.github.io/`) z verzijo Pythona 3.x.x, ki ga je mogoče namestiti brez administratorskih pravic, lahko tudi na USB ključek. Vaja služi spoznavanju osnovnih ukazov za nalaganje, prikazovanje, shranjevanje slik in upravljanje z digitalnimi slikami v programskem jeziku Python.

V jeziku Python lahko sivinske slike predstavimo z dvorazsežnimi polji `ndarray` v knjižnici `numpy`, v katerih so slikovni elementi običajno shranjeni kot nepreznačena 8-, 16- ali 32-bitna cela števila ali v zapisu s plavajočo vejico (Tabela 1).

Tabela 1: Zapis sivinskih vrednosti s podatkovnimi tipi knjižnice numpy.

Zapis sivin	Podatkovni tip (`dtype`)	Zaloga vrednosti
binarna slika	`'bool'`	{ `False` , `True` }
8-bitni nepredznačeni	`'uint8'`	[0, 255]
16-bitni nepredznačeni	`'uint16'`	[0, 65535]
32-bitni predznačeni	`'int32'`	[-2^{31}, 2^{31}-1]
32-bitni s plavajočo vejico	`'float32'` ali `'single'`	[0.0, 1.0]
64-bitni s plavajočo vejico	`'float64'` ali `'double'`	[0.0, 1.0]

Knjižnico `numpy` naložimo z ukazom `import numpy as np`, do funkcij in spremenljivk v knjižnici pa dostopamo z `np.___`. Nekatere uporabne funkcije za inicializacijo polja `ndarray` so `zeros()`, `ones()`, `zeros_like()`, `ones_like()`, `asarray()`, za pretvorbo tipa podatka `astype()` ali `array( mArray, dtype= ... )`, za branje števila dimenzij `ndim()` in velikosti `shape()` polja in za preoblikovanje polja `reshape()` in `transpose()`. Dvodimenzionalno polje `mArray` naslavljamo z npr. `mArray[0,3]` (element v prvi vrstici, četrtem stolpcu), `mArray[:,1]` (drugi stolpec), `mArray[-1,:]` (zadnja vrstica), itd. Indekse v sliki za logične izraze nad elementi slike lahko iščemo z ukazom `where()`. Koordinatni sistem slike in primeri naslavljanja elementov v sliki so prikazani na spodnji sliki.



Za branje in pisanje slik v surovem (nezgoščenem) zapisu sta v knjižnici `numpy` uporabni funkciji `fromfile()` in `tofile()`. Za branje in pisanje slik v standardnih formatih (bmp, png, gif, eps, jpeg, itd.) pa lahko uporabimo knjižnico `PIL.Image` (če pri uvažanju knjižnice Python javi napako, potem jo najprej naložite v ukaznem oknu z ukazom `pip install pillow`). Sliko naložimo s funkcijo `open()`, ki ustvari spremenljivko tipa `Image`. Zapis slikovnih elementov v sliki preverimo z ukazom `getbands()`. S funkcijo `numpy.array` to spremenljivko pretvorimo v `numpy` podatkovno polje. Sliko v obliki `numpy` polja pretvorimo nazaj v tip `Image` s funkcijo `Image.fromarray()`, pretvorimo v poljuben format s funkcijo `Image.convert()` in shranimo z ukazom `save()`.

Za prikazovanje slik lahko uporabite knjižnico `matplotlib.pyplot`. Za izris slike je uporabna funkcija

`imshow()`, za novo prikazno okno `figure()`, za urejanje osi pa `suptitle()`, `xlabel()` in `ylabel()`, `axes()`. Prikaz osvežite s funkcijo `show()`.

Gradivo za vajo vsebuje dve 2D sliki, prva `zob-microct.png` je slika rezine zoba zajeta na mikro CT napravi, druga `misice-microscope.png` pa mikroskopska slika mišičnih vlaken. Slika 1 prikazuje dani 2D sivinski sliki. Sledi nekaj osnovnih vaj za trening sintakse in uporabe knjižnic in funkcij.

1. Uporabite knjižnico `PIL.Image` za nalaganje slike `zob-microct.png` v spremenljivo v okolju Python in nato sliko pretvorite v 2D polje tipa `numpy.array`.

2. Uporabite knjižnico `matplotlib.pyplot` za prikaz slike v obliki 2D polja v spremenljivki tipa `numpy.array`.

3. Iz originalne slike izluščite pravokotno podokno, ki ima x in y dimenzijo polovico manjše kot ustrezni x in y dimenziji originalne slike. Prikažite izluščeno pravokotno podokno.

4. Shranite izluščeno podokno slike v datoteko formata `.jpg`.

Knjižnica SimpleITK

Python knjižnica SimpleITK (www.simpleitk.org) je poenostavljen vmesnik do knjižnice ITK[1], ki je originalno napisana v jeziku C++ in ki je odprtokodna knjižnica s širokim naborom osnovnih in naprednih algoritmov ter programskih orodij za analizo (medicinskih) slik.

Knjižnico SimpleITK lahko enostavno naložimo v katerokoli Python okolje tako, da v ukazni vrstici (program WinPython Command Prompt.exe v mapi WinPython) izvedemo naslednji ukaz:

```
pip3 install SimpleITK
```

Knjižnico lahko nato uvozite v Python okolje z ukazom:

```
import SimpleITK as sitk
```

kjer smo kot ime knjižnice uporabili kratek psevdonim `sitk`. Tabela 2 podaja opis osnovnih objektov in funkcij v knjižnici SimpleITK.

V nadaljevanju vaje bomo spoznali uporabo nekaterih funkcij in manipulacijo z objektom tipa `sitk.Image`.

5. Ustvarite objekt tipa `sitk.Image` iz slike `zob-microct.png`. Preberite lastnosti objekta, kot so velikost slike, tip podatka, korak vzorčenja, izhodišče in smeri koordinatnega sistema.

6. Ponastavite korak vzorčenja in izhodišče slike ter sliko shranite v datoteko formata `.nrrd`. Odprite datoteko kot tekstovno ASCII datoteko s poljubnim bralnikom in si oglejte zapis slike ter preverite lastnosti slike. Ponovite enako z uporabo formata `.nii.gz`.

7. Naložite datoteko v formatu `.nrrd` in dobljeni objekt `Image` pretvorite sliko v spremenljivko tipa `numpy.array`. Katere lastnosti slike se pri tem *izgubijo*?

Na spletu je na voljo veliko primerov uporabe knjižnice SimpleITK, naprimer na spletni strani http://insightsoftwareconsortium.github.io/SimpleITK-Notebooks. Za vajo osnov manipulacije z `sitk.Image` objektom priporočam ogled opisov in kode pod Image Details. Dokumentacijo vseh funkcij v knjižnici najdete na spletni strani https://itk.org/SimpleITKDoxygen/html/index.html.

Obnova medicinskih slik s SimpleITK

Obnova medicinskih slik se uporablja za povečanje zaznavnosti objektov oz. struktur na slikah za namen boljše klinične interpretacije ali pa kot postopek predobdelave slik za nadaljnjo avtomatsko analizo slik. Obnova slik obsega postopke za povečanje kontrasta na slikah, poudarjanje robov, povečevanje sivinske in prostorske ločljivosti, za zmanjšanje šuma in prostorskih sivinskih nehomogenosti.

Za potrebe avtomatske analize medicinskih slik se najbolj pogosto uporabljajo postopki za zmanjšanje šuma z nelinearnim filtriranjem, ki ohranja robove[2] in pa postopki za zmanjšanje prostorskih sivinskih nehomogenosti. Slika 1a prikazuje rezini mikro CT[3] slike zoba pred in po nelinearnem filtriranju za

Tabela 2: Opis osnovnih objektov in funkcij v knjižnici SimpleITK.

Element	*Tip elementa*	*Opis*
`sitkUInt8`, `sitkInt16`, `sitkFloat32`,...	celoštevilska konstanta (`int`)	določa podatkovni tip elementov slike
`Image`	objekt	vsebuje metapodatke o sliki in vrednosti elementov 2D ali 3D slike
`Image( size, valueEnum )`	konstruktor objekta `Image`	v `size` podamo velikost slike v spremenljivki `list` ali `tuple`, v `valueEnum` pa tip podatka, npr. `sitkUInt8`
`Image.GetPixelIDValue()`	funkcija objekta `Image`	vrne tip podatka v sliki v obliki celoštevilske konstante (npr. `sitkFloat32`)
`Image.GetPixelIDTypeAsString()`	funkcija objekta `Image`	vrne tip podatka v sliki v obliki besede
`Image.GetSize()`	funkcija objekta `Image`	vrne vektor z velikostmi slik v posamezni dimenziji
`Image.GetSpacing()` `Image.SetSpacing(spacing)`	funkcija objekta `Image`	vrne oziroma nastavi korak vzorčenja v posamezni dimenziji slike
`Image.GetOrigin()` `Image.SetOrigin(origin)`	funkcija objekta `Image`	vrne oziroma nastavi vektor s koordinatami izhodišča slike
`Image.GetDirection()` `Image.SetDirection( direction)`	funkcija objekta `Image`	vrne oziroma nastavi 2×2 ali 3×3 matriko smernih vektorjev 2D ali 3D slike, ki predstavljajo smeri osi koordinatnega sistema dejanskega, fizičnega objekta
`ReadImage( filename, outputPixelType )`	funkcija	nalaganje slike v poljubnem formatu (`.png`, `.jpg`, `.nrrd`, `.nii.gz`, `.dcm`, ...), s parametrom `outputPixelType` lahko eksplicitno vnaprej določimo tip podatka, funkcija vrne sliko kot objekt tipa `Image`
`WriteImage( image, fileName, useCompression )`	funkcija	shranjevanje slike `image` v poljubnem formatu, ki ga določimo s končnico v imenu datoteke `fileName`, s parametrom `useCompression` vklopimo/izklopimo kompresijo podatkov
`GetArrayFromImage( image )`	funkcija	pretvori slikovne podatke v objektu `image` v spremenljivko `numpy.array`
`GetImageFromArray( array )`	funkcija	pretvori `numpy.array` polje v spremenljivki `array` v objekt tipa `sitk.Image`, pri čemer nastavi velikost slike in tip podatka, ostale lastnosti imajo privzete vrednosti

zmanjšanje šuma, slika 1b pa mikroskopski sliki mišičnih vlaken pred in po zmanjšanju prostorskih sivinskih nehomogenosti.

Postopki za zmanjševanje šuma in prostorskih sivinskih nehomogenosti imajo podoben cilj, to je, čim bolj zmanjšati variabilnosti sivinskih vrednosti v znotraj iste strukture, pri čemer predpostavljamo, da imajo iste oz. sorodne strukture homogeno sivinsko vrednost po celotnem vidnem polju. Razlika med tema dvema skupinama postopkov je, da prvi variabilnost signala zmanjšujejo lokalno drugi pa globalno. Pri razvoju teh postopkov lahko predpostavimo naslednji model degradacije sivinske slike:

$$f(x,y) = g(x,y) * m(x,y) + a(x,y) + n(x,y), \tag{1}$$

pri čemer je $f(x,y)$ zajeta, degradirana slika, $g(x,y)$ pa nedegradirana slika. Polji $m(x,y)$ in $a(x,y)$ predstavljata multiplikativno in aditivno sivinsko nehomogenost, $n(x,y)$ pa additivni šum.

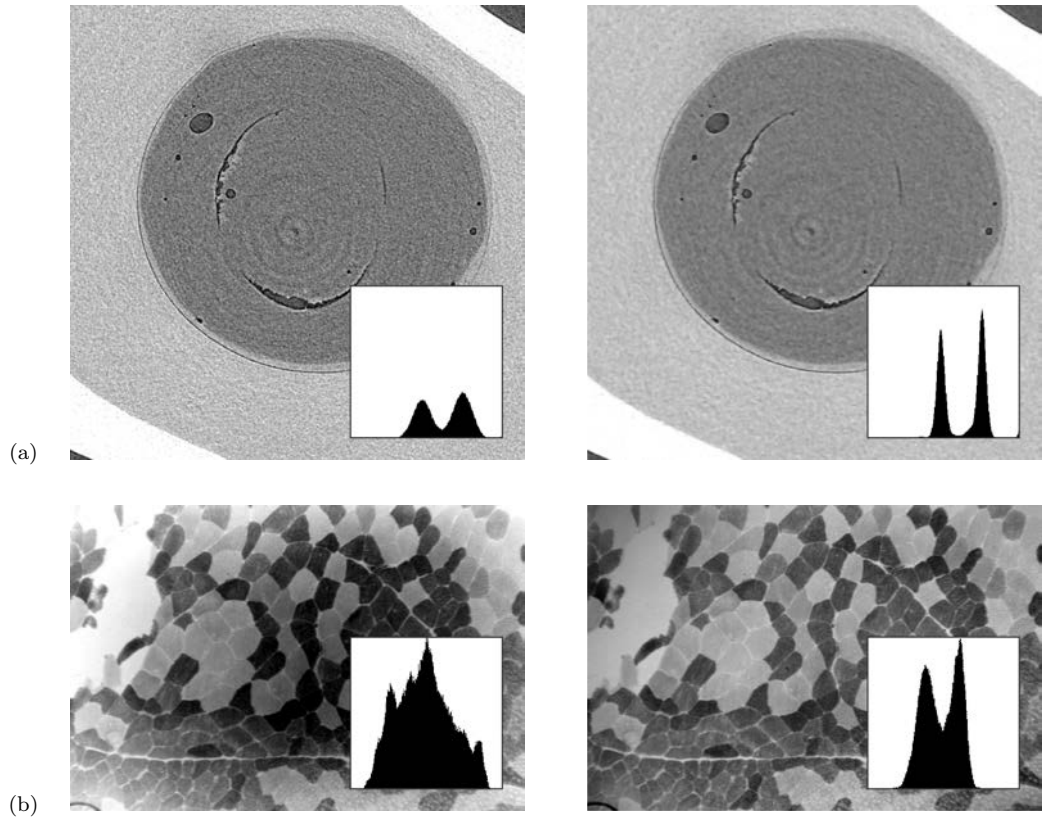
Postopki za zmanjšanje šuma običajno predpostavljajo, da je pričakovana vrednost $E(\cdot)$ za komponento additivnega šuma enaka nič ($E(n(x,y)) = 0$), zato člen za šum v enačbi (1) odpade:

$$E(f(x,y)) = E(g(x,y) * m(x,y)) + E(a(x,y)). \tag{2}$$

Osnovni princip zmanjševanja šuma je torej lokalno povprečenje oz. filtriranje slike, ki jo najlažje izvedemo z lokalnim povprečenjem, še bolj robustna izvedba pa je z uporabo *medianinega* filtra.

8. Uporabite funkciji za glajenje slike s povprečenjem in z mediano v knjižnici SimpleITK:

[1] ITK: *Insight Segmentation and Registration Toolkit*, www.itk.org
[2] ang. *edge-preserving smoothing*
[3] CT: ang. *Computed Tomography*



Slika 1: (a) Rezini mikro CT slike zoba pred (*levo*) in po (*desno*) zmanjšanju šuma z nelinearnim filtriranjem. (b) Mikroskopski sliki mišičnih vlaken pred (*levo*) in po (*desno*) zmanjšanju prostorskih sivinskih nehomogenosti. Iz pripadajočih sivinskih histogramov je razvidno povečanje kontrasta med strukturami različnih sivin.

```
# glajenje s povprečenjem
Mean(img, radius)
# glajenje z mediano
Median(img, radius)
```

kjer je `img` vhodna slika tipa `itk.Image`, parameter `radius` pa predstavlja radij filtra K, zato bo dejanska velikost filtra $2K + 1$. Obe funkciji vrneta sliko v obliki spremenljivke tipa `itk.Image`.

Preizkusite delovanje funkcij na 2D mikro CT sliki `zob-microct.png` in prikažite ter kritično ovrednotite ustreznost obnove slike.

Glavna slabost teh filtrov je, da poleg variabilnosti signala zaradi šuma zmanjšujejo tudi variabilnost koristnega signala pri prehodu med dvema strukturama in posledično zabrišejo robove ali celo izničijo signal pomembne, drobne strukture. Zato se v praksi uporabljajo postopki z nelinearnim filtriranjem, ki ohranjajo robove, naprimer anizotropna difuzija [4].

9. Uporabite funkcijo za gradientno anizotropno difuzijo v knjižnici SimpleITK:

```
# anizotropna difuzija na podlagi operatorja odvajanja
GradientAnisotropicDiffusion(img, timeStep, conductanceParameter,
  conductanceScalingUpdateInterval, numberOfIterations)
```

Z uporabo dokumentacije knjižnice SimpleITK raziščite pomen parametrov in ustrezno nastavite vrednosti parametrov. Funkcijo z vašimi nastavitvami parametrov preizkusite na 2D mikro CT sliki `zob-microct.png` in prikažite ter kritično ovrednotite ustreznost obnove slike.

[4] ang. *anisotropic diffusion*

Dodatne naloge

Dodatne naloge naj služijo za poglobitev spretnosti programiranja, boljšemu razumevanju snovi in vsebine vaje in spoznavanju dodatnih načinov za obdelavo in analizo medicinskih slik. Opravljanje dodatnih nalog je neobvezno, vendar pa priporočljivo, saj je na nek način to priprava na zagovor laboratorijskih vaj.

1. Knjižnica SimpleITK vključuje implementacije različnih nelinearnih filtrov, ki ohranjajo robove v slikah in ki temeljijo na različnih principih, denimo anizotropna difuzija, bilateralni filter, nelokalno povprečenje. Preizkusite delovanje naslednjih treh funkcij:

   ```
   # anizotropna difuzija na podlagi operatorja ukrivljenosti
   CurvatureAnisotropicDiffusion(img, timeStep, conductanceParameter,
     conductanceScalingUpdateInterval, numberOfIterations)

   # bilateralni filter
   Bilateral(img, domainSigma, rangeSigma, numberOfRangeGaussianSamples)

   # nelokalno povprečenje
   PatchBasedDenoising(image1, noiseModel, kernelBandwidthSigma, patchRadius,
     numberOfIterations, numberOfSamplePatches, sampleVariance, noiseSigma,
     noiseModelFidelityWeight)
   ```

   - Z uporabo dokumentacije knjižnice SimpleITK raziščite pomen parametrov in ustrezno nastavite vrednosti parametrov za vsako od posameznih funkcij.
   - Funkcije z vašimi nastavitvami parametrov preizkusite na 2D mikro CT sliki `zob-microct.png` in prikažite ter kritično ovrednotite ustreznost obnove slike. Pri tem si lahko pomagate z izrisom histograma intenzitet slike pred in po obnovi.
   - Kakovost obnovljenih 2D mikro CT slik lahko objektivno ovrednotite z oceno stopnje šuma na področjih slike s približno homogeno intenziteto. Stopnjo šuma naprimer ocenite z izračunom standardne deviacije intenzitet v izbranem področju s homogeno intenziteto. Primerjajte dobljeno vrednost pred in po obnovi. Na podlagi vrednosti te cenilke določite po vašem najboljši postopek/parametre za zmanjševanje šuma v slikah.

2. Knjižnica SimpleITK vključuje implementacijo popularnega postopka N4 za zmanjšanje prostorskih sivinskih nehomogenosti. Primer uporabe:

   ```
   # ustvari objekt
   corrector = N4BiasFieldCorrectionImageFilter()
   # nastavi število iteracij po nivojih
   corrector.SetMaximumNumberOfIterations([iMaxIter] * iNumLevels)
   # zaženi postopek, ki vrne obnovljeno sliko
   oImage = corrector.Execute(iImage, iMask)
   # izračunaj multiplikativni popravek
   oBiasField = itk.Divide(iImage, oImage)
   ```

   kjer so vhodne spremeljivke `iImage`, `iMask`, `iMaxIter` in `iNumLevels`, ki prestavljajo vhodno sliko dimenzij $X \times Y$, pripadajočo masko dimenzij $X \times Y$, maskimalno število iteracij postopka v vsakem nivoju in število nivojev. Izhodni spremenljivki `oImage` in `oBiasField` prestavljata obnovljeno sliko dimenzij $X \times Y$ in pripadajoče polje multiplikativnega popravka vhodne slike.

   - Preizkusite delovanje funkcije na mikroskopski sliki `misice-microscope.png`, pri čemer naj bo maska `iMask` enaka 1 na celotni vhodni sliki, parametra `iMaxIter` in `iNumLevels` pa nastavite sami.
   - Kvalitativno preverite uspešnost odprave sivinskih nehomogenosti z izrisom 1D profila intenzitet po diagonali slike (npr. od levega gornjega do desnega spodnjega kota). Primerjajte profila intenzitet pred in po obnovi.
   - Mikroskopska slika prikazuje dva dominantna tipa mišičnih vlaken. Preverite ali se to odraža, in na kakšen način, na obliki katerega od histogramov intenzitet slike pred oziroma po obnovi.

Univerza v Ljubljani, Fakulteta za elektrotehniko
Analiza medicinskih slik (AMS)

Filtriranje z anizotropno difuzijo

Model degradacije slike

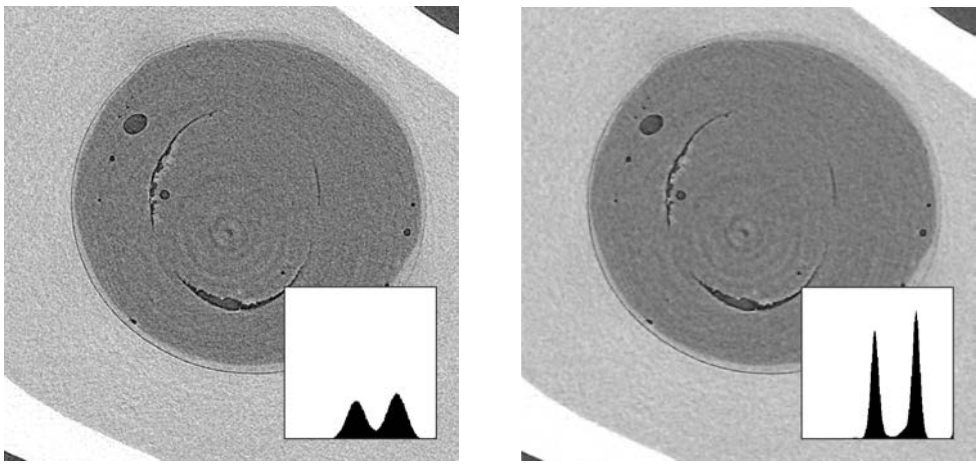
Postopki za odstranjevanje šuma variabilnost signala zmanjšujejo lokalno, pri čemer predpostavljajo naslednji model degradacije sivinske slike:

$$f(x,y) = g(x,y) + n(x,y), \tag{1}$$

pri čemer je $f(x,y)$ zajeta, degradirana slika, $g(x,y)$ pa nedegradirana slika. Polje $n(x,y)$ predstavlja additivni šum. Pri razvoju postopkov odstranjevanja šuma je običajna predpostavka, da je pričakovana vrednost $E(\cdot)$ za komponento additivnega šuma enaka nič ($E(n(x,y)) = 0$), zato člen za šum v enačbi (1) odpade:

$$E(f(x,y)) = E(g(x,y)). \tag{2}$$

Osnovni princip zmanjševanja šuma je torej lokalno povprečenje oz. filtriranje slike. V praksi uporabljajo postopki z nelinearnim filtriranjem, ki ohranjajo robove, naprimer anizotropna difuzija (ang. *anisotropic diffusion*), bilateralni filter, nelokalno povprečenje (ang. *non-local means*), itd. Slika 1 prikazuje primer filtriranja z anizotropno difuzijo.



Slika 1: Rezini mikro CT slike zoba pred (*levo*) in po (*desno*) zmanjšanju šuma z nelinearnim filtriranjem z anizotropno difuzijo.

Filtriranje slike in difuzija

Filtriranje po principu difuzije temelji na rešitvi linearne enačbe transporta toplote oz. Laplaceovi enačbi:

$$\frac{\partial f(x,y,t)}{\partial t} = \Delta f(x,y,t) = \frac{\partial^2 f(x,y,t)}{\partial x^2} + \frac{\partial^2 f(x,y,t)}{\partial y^2}\,, \tag{3}$$

ki predstavlja izotropno difuzijo in katere rešitev je linearno filtriranje z Gaussovim jedrom $\mathcal{N}(x,y|\mu = 0, \sigma^2 = t)$ z varianco t:

$$f(x,y,t) = f(x,y,0) \otimes \mathcal{N}(x,y|\mu = 0, \sigma^2 = t)\,, \tag{4}$$

kjer je $\otimes$ operator konvolucije. V tem primeru s povečevanjem časa t povečujemo stopnjo difuzije oz. stopnjo glajenja.

Filtriranje z anizotropno difuzijo

Pri anizotropni difuziji želimo gladiti le na homogenih področjih in prečno na smer roba. Če enačbo (3) zapišemo kot $\partial f(x,y,t)/\partial t = \nabla \cdot [\nabla f(x,y)]$, kjer $\nabla\cdot$ predstavlja operator za divergenco, potem lahko opazimo, da v tej enačbi nastopa odvod ∇f, ki kodira jakost in smer roba. To informacijo lahko uporabimo za ohranjanje robov v sliki na način, da enačbo prilagodimo:

$$\frac{\partial f(x,y,t)}{\partial t} = \nabla \cdot [w(\|\nabla f(x,y)\|)\nabla f(x,y)] \;, \tag{5}$$

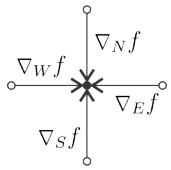
kjer je $w(\cdot)$ utežna funkcija. Naj bo $f_x = \partial f(x,y)/\partial x$ (in podobno f_y), potem velja $\psi(f_x) = w(|f_x|)\cdot f_x$, zato dobimo:

$$\frac{\partial f(x,y,t)}{\partial t} = \frac{\partial \psi(f_x)}{\partial x} + \frac{\partial \psi(f_y)}{\partial y} = \psi'(f_x)\cdot f_{xx} + \psi'(f_y)\cdot f_{yy}\,. \tag{6}$$

Difuzija naj bo večja, ko bo $\psi'(f_x) > 0$ oz. $\psi'(f_y) > 0$ in jakost roba upada in obratno, $\psi'(f_x) < 0$ oz. $\psi'(f_y) < 0$ in jakost roba narašča. Primerni utežni funkciji, ki se pogosto uporabljata sta:

$$w(\|\nabla f\|) = \frac{\kappa^2}{\kappa^2 + \|\nabla f\|^2} \quad \text{ali} \quad w(\|\nabla f\|) = \exp(-\|\nabla f\|^2/\kappa^2)\,. \tag{7}$$

Slika 2 podaja diskretno obliko enačb za implementacijo nelinearnega filtra z anizotropno difuzijo. Osnovni parametri filtra so κ, časovni korak dt in število korakov oz. iteracij t_{max}. Večje vrednosti κ zmanjšujejo anizotropičnost filtra (za $\kappa = \infty$ dobimo izotropen filter), časovni korak pa mora biti majhen, da velja diskretna aproksimacija. Majhen korak zahteva več iteracij za konvergenco, zato moramo velikost koraka in število iteracij ustrezno nastaviti.



$$\nabla_N f = f(x,y+1) - f(x,y)$$
$$\nabla_S f = f(x,y-1) - f(x,y)$$
$$\nabla_E f = f(x+1,y) - f(x,y)$$
$$\nabla_W f = f(x-1,y) - f(x,y)$$
$$w_d = w(\|\nabla_d f\|),\; d = N,S,E,W$$

$$f(x,y,t+dt) = f(x,y,t) + dt\cdot(w_N\nabla_N f + w_S\nabla_S f + w_E\nabla_E f + w_W\nabla_W f)$$

Slika 2: Diskretna implementacija nelinearnega filtra z anizotropno difuzijo.

Programerski izziv

Diskretna oblika enačbe v sliki 2 je primerna za direktno implementacijo filtra na osnovi gradientne anizotropne difuzije. Napišite funkcijo za zmanjševanje šuma, ki temelji na postopku gradientne anizotropne difuzije:

```
def denoiseGradientAnisotropicDiffusion(iImage, iStep, iMaxIter, iKappa):
        return oImage
```

kjer je `iImage` vhodna slika dimenzij $X \times Y$, `iStep` časovni korak postopka (parameter dt, slika 2), `iMaxIter` maksimalno število iteracij postopka, `iKappa` pa parameter $\kappa = [0,\infty]$ utežne funkcije $w(\cdot)$ v enačbi:

$$w(\|\nabla f\|) = \exp(-\|\nabla f\|^2/\kappa^2)\,. \tag{8}$$

Večje vrednosti κ zmanjšujejo anizotropičnost filtra (za $\kappa = \infty$ dobimo izotropen filter), časovni korak pa mora biti majhen, da velja diskretna aproksimacija. Majhen korak zahteva več iteracij za konvergenco, zato moramo velikost koraka in število iteracij ustrezno nastaviti. Funkcija vrne obdelano sliko `oImage`, ki ima dimenzije enake kot vhodna slika.

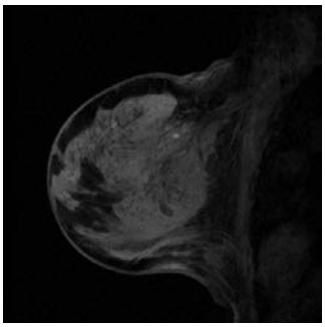
Preizkusite delovanje funkcije na 2D mikro CT sliki `zob-microct.png` s parametri `iStep=1/16.0`, `iMaxIter=30` in `iKappa=16.0` in obnovljeno sliko primerjajte z originalno, degradirano sliko. Primerjajte rezultate s tistimi, ki jih dobite z uporabo funkcije `GradientAnisotropicDiffusion()` v knjižnici SimpleITK.

Univerza v Ljubljani, Fakulteta za elektrotehniko
Analiza medicinskih slik (AMS)

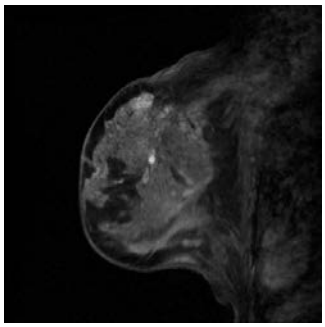
Netoga poravnava slik

Navodila

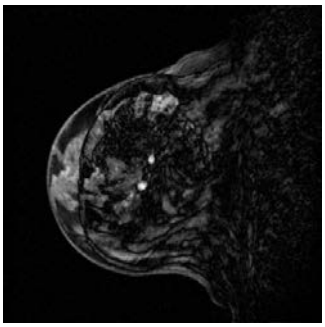
Poravnava medicinskih slik se uporablja v številnih kliničnih aplikacijah in poskusih, lahko pa tudi kot orodje za razgradnjo slik in modeliranje oblike. Naprimer, poravnava slik različnih oseb oz. bolnikov omogoča karakterizacijo razlik v anatomski in funkcionalni informaciji in vrednotenje razlik glede na referenčno informacijo v obliki atlasa. Poravnava slik istega bolnika se uporablja za zlivanje informacije med komplementarnimi slikovnimi tehnikami z namenom izboljšane diagnostike in spremljanja ali načrtovanja zdravljenja. Primer zelo razširjenega longitudinalnega spremljanja za potencialna rakava obolenja je slikanje dojke z magnetno resonanco (MR[1]), ki se pri bolnicah s povečanim tveganjem za nastanek in agresivnejši razvoj te bolezni opravlja enkrat letno. Z uporabo intravenozno vbrizganega kontrastnega sredstva se poudari žilje in patološke strukture vpete na žilje, kot je tumor. Ojačene strukture lahko z uporabo poravnave še poudarimo z odštevanjem slike brez in s kontrastom (Slika 1).



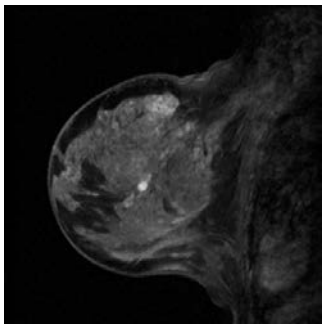
MR dojke brez kontrasta



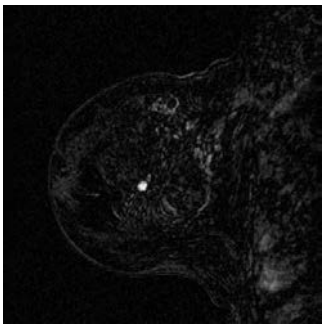
MR dojke po kontrastu



Razlika slik brez in s kontrastom



MR dojke po kontrastu
z netogo poravnavo



Razlika slik brez in s kontrastom

Slika 1: Diagnostika in spremljanje raka dojke z MR brez in s kontrastom (*zgoraj*). Z netogo poravnavo lahko prostorsko normaliziramo strukture za odštevno slikanje, ki poveča občutljivost tehnike na patološke spremembe v dojki (*spodaj*).

Prostorska poravnava anatomskih struktur, ki prestavljajo mehka tkiva, je običajno bolj zahtevna od poravnave togih oz. nedeformabilnih struktur kot so kosti. Zahtevnost zavisi predvsem od števila prostostnih stopenj modela preslikave, pri čemer za poravnavo slik kostnih struktur in možganov običajno zadostujeta toga, podobnostna ali afina preslikava (6, 7, 12 parametrov v 3D), za poravnavo slik jeter ali dojk pa je potrebno uporabiti kompleksnejše (nelinearne) modele preslikave, ki imajo lahko tudi več 100 parametrov. Takim postopkom zato pravimo **netoga poravnava** ali **linearna poravnava** slik.

Netoga poravnava je proces iskanja optimalne geometrijske preslikave $\mathcal{T} : \mathbb{R}^d \rightarrow \mathbb{R}^d; d \in \{2, 3\}$, ki anatomske strukture v modelu oz. sliki I_m nekega objekta preslika v tako lego, ki je *skladna* z lego anatomskih struktur v modelu oz. sliki I_f. Glavni izziv pri netogi poravnavi je izbira ustreznega modela preslikave $\mathcal{T}$, ki verno opisuje dejanske fizikalne deformacije opazovanih anatomskih struktur in ima čim

[1] MR – ang. Magnetic Resonance

manj prostih parametrov. Večje število prostih parametrov sicer omogoča opis bolj poljubne preslikave, vendar je pri tem lahko kritična slabost numerična nestabilnost postopka poravnave, z več parametri pa hitro narašča tudi računska zahtevnost. Pri vaji boste načrtali postopek netoge poravnave s splošnim in zelo popularnim modelom preslikave z B-zlepki in optimizacijo mere podobnosti MP med premično sliko I_m in referenčno sliko I_f.

B-zlepki

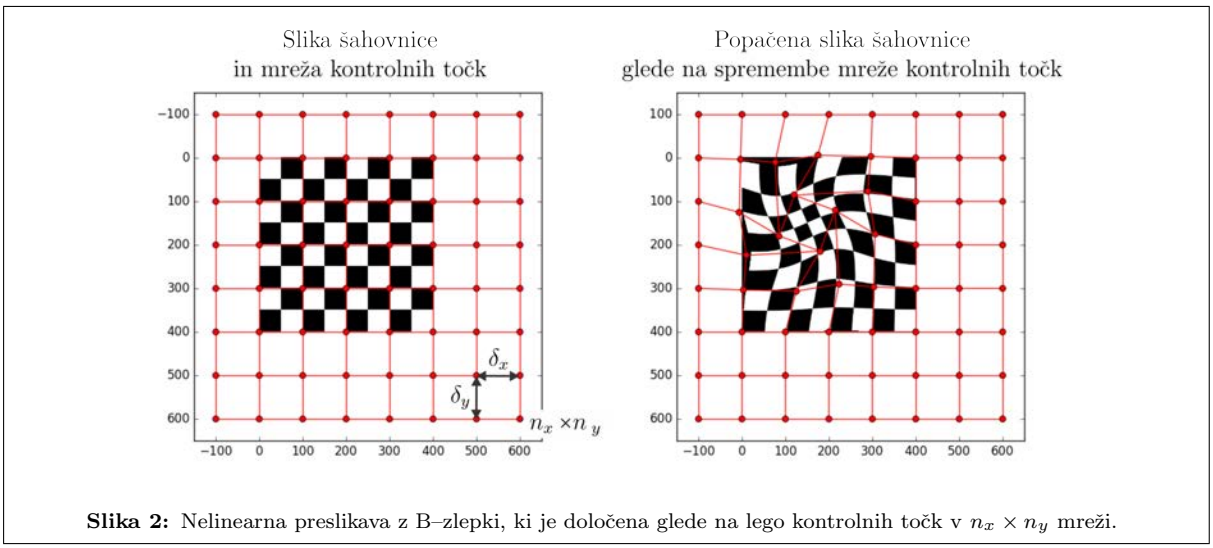
B–zlepki omogočajo modeliranje poljubne deformacije[2] v poljubno dimenzionalnih prostorih, pri čemer je deformacija določena na podlagi položaja mreže kontrolnih točk. S spreminjanjem položaja kontrolnih točk vplivamo na deformacijo, tako dobljena preslikava pa bo vedno gladka in zvezna. Deformacijo 2D slike z B–zlepki definiramo na področju slike $\Omega = \{\mathbf{p} = (x, y) \,|\, 0 \leq x \leq X, 0 \leq y \leq Y\}$. Naj Ψ označuje $n_x \times n_y$ mrežo kontrolnih točk $\psi_{i,j}$, ki ima pripadajoči razmak δ_x in δ_y v x in y osi slike (Slika 2). Uporabili bomo uni-variatne B–zlepke, ki jih s tenzorskim produktom lahko hitro razširimo na poljubno število dimenzij. Za 2D slike preslikavo s kubičnimi B–zlepki zapišemo kot:

$$\mathcal{T}_{2D}(\mathbf{p}) = \sum_{l=0}^{3} \sum_{m=0}^{3} B_l(u)\, B_m(v)\, \psi_{i+l,j+m} \tag{1}$$

pri čemer so $i = \lfloor \frac{x}{\delta_x} \rfloor$, $j = \lfloor \frac{y}{\delta_y} \rfloor$, $u = \frac{x}{\delta_x} - i$, $v = \frac{y}{\delta_y} - j$ in kjer B_l predstavlja l-to bazno funkcijo B–zlepka:

$$B_0 = (1-u)^3/6 \tag{2}$$
$$B_1 = (3u^3 - 6u^2 + 4)/6 \tag{3}$$
$$B_2 = (-3u^3 + 3u^2 + 3u + 1)/6 \tag{4}$$
$$B_3 = u^3/6. \tag{5}$$

Z B–zlepki lahko načrtamo zelo učinkovito poravnavo slik tudi z velikim številom kontrolnih točk, kajti bazne funkcije vplivajo na preslikavo le v lokalni okolici kontrolne točke (področje 4×4 kontrolnih točk za kubične B–zlepke) in zato spreminjanje lege točke $\psi_{i,j}$ vpliva le na preslikavo v ustrezni okolice te točke (Slika 2). Da bo polje deformacij znotraj celotne slike mora, razen v primeru B–zlepkov ničtega reda, mreža kontrolnih točk segati izven slike. Za kubične B–zlepke in sliko velikosti $X \times Y$ mrežo določimo kot je prikazano na Sliki 2, pri čemer je $n_x = \lceil \frac{X}{\delta_x} \rceil + 3$ in $n_y = \lceil \frac{Y}{\delta_y} \rceil + 3$. V prikazanem primeru imamo 64 kontrolnih točk $\psi_{i,j}$ v 2D, kar pomeni 128 prostih parametrov preslikave.



Slika 2: Nelinearna preslikava z B–zlepki, ki je določena glede na lego kontrolnih točk v $n_x \times n_y$ mreži.

Netoga poravnava z B-zlepki

Prostorsko ujemanje med slikama lahko ovrednotimo z mero podobnosti MP; to je skalarna funkcija, ki zavzame optimum, ko se položaji korespondenčnih struktur v slikah I_m in I_f medsebojno prekrivajo.

[2] FFD – ang. Free-Form Deformation

Mero podobnosti je potrebno smiselno izbrati tako, da je čim manj občutljiva na motilna slikovna neskladja in čim bolj občutljiva na dejanska geometrijska neskladja med slikama. Za poravnavo medicinskih slik se pogosto uporablja **medsebojna informacija** MI[3].

Tekom poravnave nam izbrana optimizacijska metoda iterativno spreminja parametre $\psi_{i,j}$ geometrijske preslikave $\mathcal{T}(\mathbf{p}) = \mathcal{T}(\mathbf{p}|\psi_{i,j})$ tako, da naprimer maksimizira mero podobnosti:

$$\psi^*_{i,j} = \text{argmax}_{\psi_{i,j}} MP\Big(I_m\big(\mathcal{T}(\mathbf{p}|\psi_{i,j})\big), I_f(\mathbf{p})\Big)\,,$$

kjer so $\psi^*_{i,j}$ optimalni parametri preslikave $\mathcal{T}(\mathbf{p}|\psi_{i,j})$.

Učinkovite implementacije netoge poravnave z B–zlepki temeljijo na uporabi gradientnih optimizacijskih postopkov, saj je model preslikave z B–zlepki enostavno odvedljiv in ima lokalni vpliv na deformacije v sliki, to pa poenostavi izračun odvoda mere podobnosti. Za prikaz uporabe netoge poravnave z B–zlepki bomo uporabili implementacijo, ki je na voljo v Python knjižnici SimpleITK.

Analiza polja deformacij

Netoga poravnava slik se precej uporablja za zaznavanje sprememb in gibanja med zaporednimi slikovnimi preiskavami. Z netogo poravnavo je možno tudi natančno kvantificirati oz. izmeriti te spremembe, naprimer tako, da izmerimo spremembo prostornine na področju zanimanja Ω_O točno določenega organa O. Lokalna sprememba prostornine v infinitezimalno majhni okolici slikovnega elementa je določena z determinanto lokalno ocenjene Jacobijeve matrike:

$$J = \begin{bmatrix} \partial T_x/\partial x & \partial T_x/\partial y \\ \partial T_y/\partial x & \partial T_y/\partial y \end{bmatrix}. \tag{6}$$

Če za preslikavo uporabimo B–zlepke je izračun matrike J še posebej enostaven. Prvi stolpec matrike J lahko tako dobimo z odvajanjem preslikave:

$$\frac{\partial}{\partial x}\mathcal{T}_{2D}(\mathbf{p}) = \sum_{l=0}^{3}\sum_{m=0}^{3}\left(\frac{d}{du}B_l(u)\right) B_m(v)\,\psi_{i+l,j+m}\,, \tag{7}$$

podobno izračunamo $\frac{\partial}{\partial y}\mathcal{T}_{2D}(\mathbf{p})$ za drugi stolpec. Ker so bazne funkcije B_l zvezne in zvezno odvedljive jih lahko z analitičnim odvajanjem ($\frac{d}{du}B_l(u)$) prilagodimo za izračun elementov matrike J. Na ta način v vsaki točki dobimo oceno matrike J, determinanta te matrike ($\det J$) pa kaže relative spremembe prostornine posameznega slikovnega elementa kot posledica poravnave slik (> 1 raztezanje, < 1 krčenje, 1 nespremenjeno). Ker je z uporabo B–zlepkov polje deformacij in njegov odvod določen za vsak slikovni element lahko ustvarimo kar sliko $\det J$, za področje zanimanja $\Omega_O \subset \Omega$ pa nato z integriranjem določimo relativno spremembo prostornine $\nabla V_O = 100\% \cdot (\frac{1}{|\Omega_O|}\sum_{\Omega_O}\det J - 1)$. Slika 3 prikazuje polje deformacij in pripadajoče polje $\det J(\mathrm{p})$ za netogo poravnavo MR slik dojke iz slike 1.

Naloge

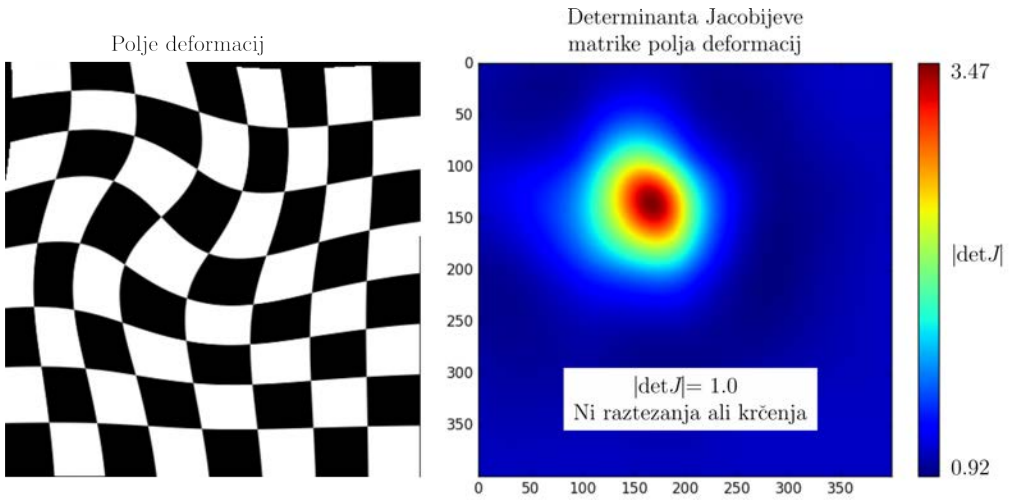
Gradivo za vajo vsebuje dve 2D MR sliki dojke, `mr-nonenhanced.png` in `mr-enhanced.png`, prvo brez in drugo s kontrastnim sredstvom. Slike naložite z ukazom `Image.open()`, knjižnice `PIL.Image`, in pretvorite v `numpy.array` polje. Sliki sta sivinski v nepredznačenem 8–bitnem zapisu, korak vzorčenja pa je 1 milimeter. Slika 1 prikazuje dani MR sliki. Dana je tudi Python skripta `AMS2_NetogaPoravnava.py`, ki jo uporabite za izvedbo vaje.

1. Napišite funkcije za izračun kubičnih B-zlepkov B_0, B_1, B_2 in B_3. Za nadaljnjo rabo funkcije organizirajte v obliki seznama tipa `tuple`.

2. Napišite funkcijo, ki za dane dimenzije slike in korak vzorčenja ustvari 2D mrežo kontrolnih točk za kubične B–zlepke:

```
def getCubicBSpline2DGrid(iImageSize, iStep):
        return oCPx, oCPy
```

[3] MI – ang. Mutual Information



Slika 3: Primer polja deformacij za poravnavo para danih MR slik (*levo*) in pripadajoča mapa determinante Jacobijeve matrike (*desno*), ki kaže relative spremembe prostornine posameznega slikovnega elementa kot posledica poravnave slik (> 1 raztezanje, < 1 krčenje).

kjer je `iImageSize` vektor 1×2 in podaja dimenzije slike $X \times Y$, parameter `iStep` pa korak vzočenja (δ_x, δ_y) mreže kontrolnih točk vzdolž x in y osi slike. Funkcija naj vrne dve polji kontrolnih točk za kubične B–zlepke `oCPx` in `oCPy` z dimenzijami $n_x \times n_y$, pri čemer sta $n_x = \lceil \frac{X}{\delta_x} \rceil + 3$ in $n_y = \lceil \frac{Y}{\delta_y} \rceil + 3$. Uporabite lahko ukaz `numpy.meshgrid()`, vendar poskrbite da bodo kubični B–zlepki določeni med vsako kontrolno točko v sliki, kar pomeni da morajo obstajati vzdolž vsake osi slike 4 kontrolne točke, od tega ena levo/zgoraj in dve desno/spodaj od najbližje kontrolne točke v sliki.

Ustvarite sliko šahovnice z dimenzijami $(X \times Y) = (400 \times 400)$ in velikostjo polja v šahovnici 50×50 tako, da uporabite funkcijo `getCubicBSpline2DGrid(iImageSize, iStep)`, kjer je `iImageSize` vektor 1×2 in podaja dimenzije izhodne slike šahovnice, `iArraySize` pa spremenljivka, ki podaja dimezije posameznega polja v šahovnici. Nato ustvarite mrežo kontrolnih točk za sliko šahovnice in prikažite mrežo kontrolnih točk kot je prikazano na Sliki 2.

3. Napišite funkcijo, ki za dano 2D mrežo kontrolnih točk kubičnih B–zlepkov izračuna polje deformacij:

```
def getCubicBSpline2DDeformation(iImageSize, iCPx, iCPy, iStep):
        return oGx, oGy
```

kjer je `iImageSize` vektor 1×2 in podaja dimenzije slike $X \times Y$, `iCPx` in `iCPy` polji kontrolnih točk za kubične B–zlepke z enakimi dimenzijami $n_x \times n_y$, parameter `iStep` pa korak vzočenja (δ_x, δ_y) originalne mreže kontrolnih točk vzdolž x in y osi slike. Funkcija naj vrne dve dvorazsežni polji `oGx` in `oGy`, ki imata dimenzije kot jih podaja vhodni parameter `iImageSize`.

Funkcijo najprej preizkusite z originalno mrežo kontrolnih točk, pri čemer morate v izhodnih spremenljivkah `oGx` in `oGy` dobiti koordinate vzorčnih točk slike, kot jih vrne ukaz `numpy.meshgrid()`.

4. Napišite funkcijo, ki za dano 2D mrežo kontrolnih točk kubičnih B–zlepkov izračuna polje deformacij in prevzorči sliko:

```
def deformImageBSpline2D(iImage, iCPx, iCPy, iStep):
        return oImage
```

kjer je `iImage` sivinska 2D slika dimenzij $X \times Y$, `iCPx` in `iCPy` pripadajoči polji kontrolnih točk $\psi_{i,j}$ za kubične B–zlepke z dimenzijami $n_x \times n_y$, parameter `iStep` pa korak vzočenja (δ_x, δ_y) originalne mreže kontrolnih točk vzdolž x in y osi slike. Funkcija naj izračuna polje deformacij $d(\mathbf{p}) = \mathcal{T}(\mathbf{p}|\psi_{i,j}) - \mathbf{p}$ s funkcijo `getCubicBSpline2DDeformation()` in določi nove vzorčne koordinate kot $\mathbf{p}' = \mathbf{p} - d(\mathbf{p})$. Z uporabo (bi)linearne interpolacije določite sivinske vrednosti v koordinatah $\mathbf{p}'$ na vhodni sliki `iImage`. Izhodna slika `oImage` ima enake dimenzije kot vhodna.

Preizkusite funkcijo tako, da translirate ($\pm 20mm$) posamezne x in y koordinate kontrolnih točk $\psi_{i,j}$ in deformirate sliko šahovnice iz druge naloge.

5. Z uporabo knjižnice SimpleITK preizkusite postopek poravnave z B–zlepki na danem paru MR slik:

```
import SimpleITK as itk
# naloži slike
fixed = itk.ReadImage('mr-nonenhanced.png', itk.sitkFloat32)
moving = itk.ReadImage('mr-enhanced.png', itk.sitkFloat32)

# inicializacija postopka
R = itk.ImageRegistrationMethod()

# inicializacija preslikave z B-zlepki
bsplineGrid = 8
bTr = itk.BSplineTransformInitializer(fixed, [bsplineGrid] * 2)
R.SetInitialTransform(bTr, inPlace=True)

# inicializacija mere podobnosti
R.SetMetricAsMattesMutualInformation(50)
R.SetMetricSamplingPercentage(0.10)
R.SetMetricSamplingStrategy(R.RANDOM)

# inicializacija optimizacije
R.SetOptimizerAsGradientDescentLineSearch(learningRate=5.0,
                                          numberOfIterations=100,
                                          convergenceMinimumValue=1e-5,
                                          convergenceWindowSize=5)
R.SetOptimizerScalesFromPhysicalShift()

# zagon poravnave
outTx = R.Execute(fixed, moving)

# ustvarjanje izhodne slike
S = itk.ResampleImageFilter()
S.SetReferenceImage(fixed)
S.SetInterpolator(itk.sitkLinear)
S.SetDefaultPixelValue(0)
S.SetTransform(outTx)
outImage = S.Execute(moving)

# shranjevanje izhodne slike
itk.WriteImage(itk.Cast(outImage, itk.sitkUInt8),
                    'mr-enhanced-registered.png', True)
```

Prikažite poravnano sliko in preverite ali je bila poravnava uspešna. Prikažite še odšteti sliki pred in po poravnavi tako, kot je prikazano na sliki 1. Pred odštevanjem slik ustrezno medsebojno prilagodite sivinske vrednosti slik.

Dodatne naloge

Dodatne naloge naj služijo za poglobitev spretnosti programiranja, boljšemu razumevanju snovi in vsebine vaje in spoznavanju dodatnih načinov za obdelavo in analizo medicinskih slik. Opravljanje dodatnih nalog je neobvezno, vendar pa priporočljivo, saj je na nek način to priprava na zagovor laboratorijskih vaj.

1. Z 2D togo preslikavo $(t_x, t_y, \alpha) = (40mm, -20mm, 30°)$ preslikajte 2D mrežo kontrolnih točk, ki ste jih ustvarili pri Nalogi 2 (`createCubicBSpline2DGrid()`).

- Izrišite polje deformacij v obliki vektorskega polja z ukazom `quiver` v knjižici `matplotlib.pyplot`.
- Sliko šahovnice iz Naloge 2 preslikajte z dobljenim poljem deformacij in jo prikažite.

2. Analitično izračunajte odvode d/du baznih funkcij kubičnih B–zlepkov $B_l(u), l = 0, 1, 2, 3$.

3. Napišite funkcijo, ki za dano 2D mrežo kontrolnih točk kubičnih B–zlepkov izračuna polje Jacobijeve determinante:

```
def getJacobianDet(iImageSize, iCPx, iCPy, iStep):
        return oJacDet
```

kjer je `iImageSize` vektor 1×2 in podaja dimenzije slike $X \times Y$, `iCPx` in `iCPy` polji kontrolnih točk za kubične B–zlepke z enakimi dimenzijami $n_x \times n_y$, parameter `iStep` pa korak vzočenja (δ_x, δ_y) originalne mreže kontrolnih točk vzdolž x in y osi slike. Funkcija naj vrne dvorazsežno polje `oJacDet`, ki ima dimenzije kot jih podaja vhodni parameter `iImageSize`.

- Za kontrolne točke iz Dodatne naloge 1 prikažite polje Jacobijeve determinante kot barvno sliko z lestvico `jet` (primer na Sliki 3).
- Za kontrolne točke, ki ste jih ročno premaknili pri Nalogi 4 prikažite polje Jacobijeve determinante kot barvno sliko z lestvico `jet`.
- Izračunajte spremembo prostornine v % preko celotne slike za prejšnji dve točki.

4. Z netogo poravnavo iz Naloge 5, pri čemer so kontrolne točke B–zlepkov dane v spremenljivki `outTx`, lahko z naslednjimi ukazi pridobite polje deformacij:

```
# ustvari polje deformacij na osnovi kontrolnih točk
displacement = itk.TransformToDisplacementField(
    outTx,
    size=fixed.GetSize(),
    outputOrigin=fixed.GetOrigin(),
    outputSpacing=fixed.GetSpacing(),
    outputDirection=fixed.GetDirection()
)

# ustvari numpy.array polje
arr = itk.GetArrayFromImage()
```

- Podobno kot pri Dodatni nalogi 1 izrišite polje deformacij v obliki vektorskega polja z ukazom `quiver` v knjižici `matplotlib.pyplot`.
- Z ukazom `itk.DisplacementFieldJacobianDeterminant(displacement, True)` izračunajte polje Jacobijeve determinante in ga prikažite kot barvno sliko z lestvico `jet`.
- Izračunajte spremembo prostornine v % preko celotne slike za prejšnjo točko.

Univerza v Ljubljani, Fakulteta za elektrotehniko
Analiza medicinskih slik (AMS)

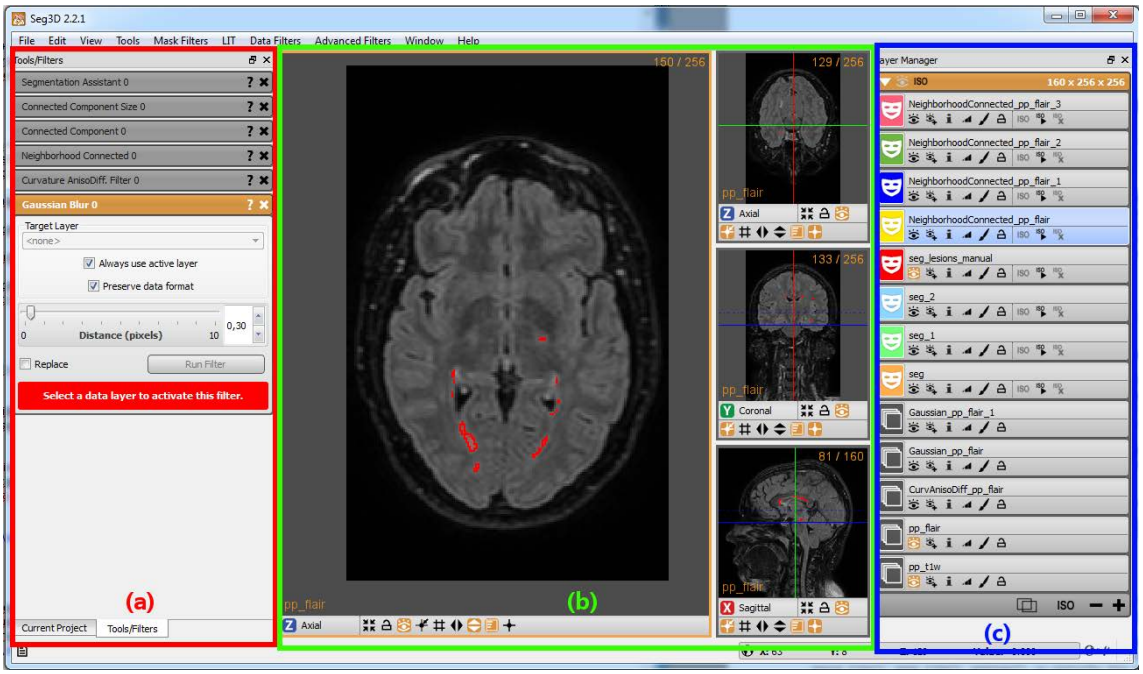
Interaktivno prikazovanje 3D slik

BrainSeg3D

Seg3D (http://www.sci.utah.edu/cibc-software/seg3d.html) je odprtokodni program za interaktivno prikazovanje, obdelavo in analizo 3D slik. Med drugim omogoča ročno, interatktivno (polavtomatsko) in avtomatsko obrisovanje struktur in uporabo drugih postopkov za obdelavo slik iz Insight Toolkit (ITK) knjižnice. Uporabnik lahko pregleduje 3D slike po posameznih rezinah vzdolž osi slik ali s pomočjo 3D vizualizacije. Na vajah bomo uporabljali BrainSeg3D, razširjeno različico programa Seg3D, ki vsebuje še dodatna orodja za interaktivno razgradnjo slik.

Okno programa BrainSeg3D (Slika 1) tvorijo tri glavna področja:

- **Orodja:** izbira želenega orodja in nastavitev njegovih parametrov.
- **Prikazovalnik:** prikaz izbrane slike po rezinah ali s 3D prikazom volumna.
- **Urejevalnik:** dodajanje/brisanje sklada slik ali mask ter določanje svetlosti in kontrasta slik, pojavnosti mask (npr. barve) in vklop/izklop 2D/3D prikaza.



Slika 1: Okno programa BrainSeg3D delimo na tri glavna področja: (a) orodja, (b) prikazovalnik in (c) urejevalnik.

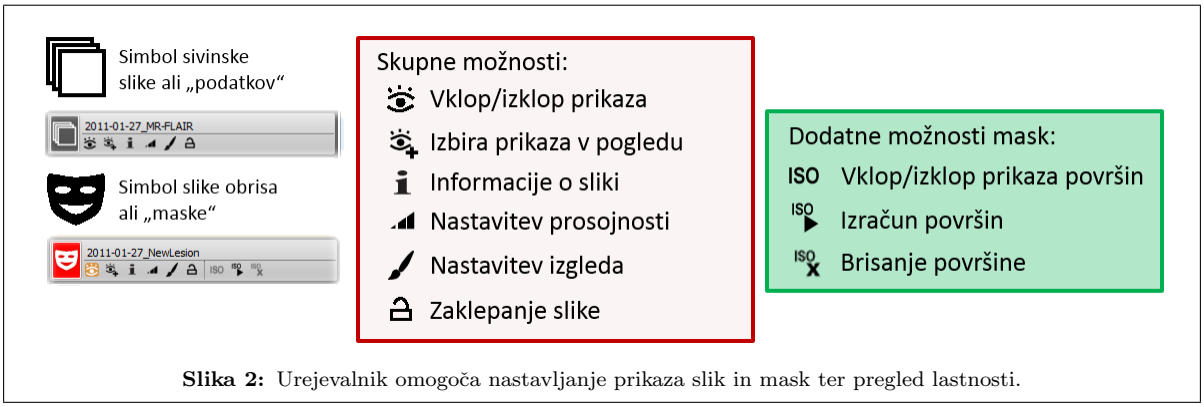
Program BrainSeg3D se uporablja za interaktivni vizualizacijo, obdelavo in analizo 3D medicinskih slik. Predvsem ga uporabljamo za obrisovanje struktur v medicinskih slikah. Naprimer, obrise lahko najprej pridobimo avtomatsko, z razgradnjo slike, in jih nato po potrebi ročno popravimo z interaktivnimi orodji. V tej vaji si bomo pogledali obrisovanje različnih normalnih in patoloških struktur v medicinskih slikah, kot so žile, jetra, anevrizme in lezije bele možganovine.

Urejevalnik

Ločimo dva tipa slik: sivinske slike (podatki – ang. *data*) in slike obrisa (maske – ang. *masks*). V urejevalniku so navedene vse slike in maske, ki so trenutno odprte v programu BrainSeg3D. Urejevalnik

deluje kot *sklad*: maske više v skladu so prikazane nad tistimi niže v skladu, podobno je prikazana le slika najviše v skladu. Z vklopom/izklopom prikaza lahko izberemo prikaz slike.

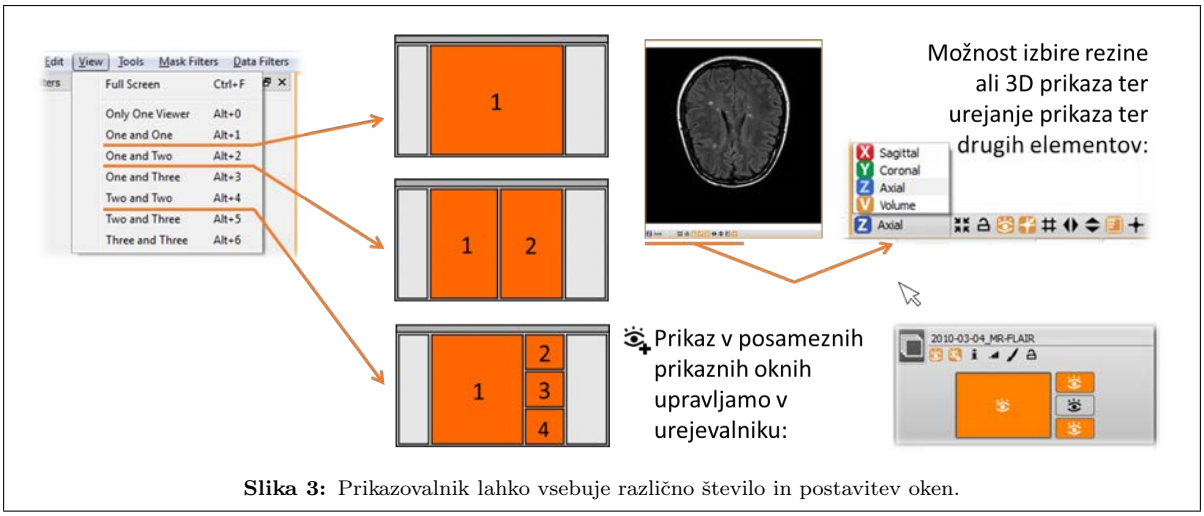
Tip slike določa možnosti obdelave in izbiro orodij, ki jih lahko uporabimo. Običajno prikazujemo le eno sivinsko sliko, medtem, ko je število prikazanih mask lahko poljubno z barvnim kodiranjem. Maska je binarna slika, ki jo večinoma uporabljamo za obrisovanje želenih struktur v sliki. Take obrise lahko tvorimo avtomatsko z uporabo računalniških orodij ali pa z ročnim obrisovanjem. Sivinske slike običajno pregledujemo po rezinah ali pa s 3D upodabljanjem, maske pa v 2D kot področja ali robove ter v 3D kot površine. Podroben opis možnosti v urejevalniku prikazuje Slika 2.



Slika 2: Urejevalnik omogoča nastavljanje prikaza slik in mask ter pregled lastnosti.

Prikazovalnik

Prikazovalnik lahko vsebuje različno število in postavitev oken (Slika 3). Željeno nastavitev izberemo pod **View** → ⋯ . Vsako prikazno okno ima v spodnjem levem kotu možnosti nastavitev, ki vključujejo izbiro prikaza ravnine (*prečno, stransko, čelno*) ali 3D upodabljanje, skaliranje prikaza, zaklepanje rezine (za sinhrono prikazovanje z drugim prikaznim oknom), vklop/izklop prikaza rezine, sinhrono fokusiranje prikaza, prikaz diskretne mreže, zrcaljenje v horizontalni ali vertikalni osi, vklop/izklop informacij o prikazu, itd.



Slika 3: Prikazovalnik lahko vsebuje različno število in postavitev oken.

Orodja

Spoznali se bomo z naslednjimi orodji v BrainSeg3D:

- **Upragovanje (Tools → Threshold)**: Razgradnja slike v binarno masko na podlagi pražne sivinske vrednosti.
- **Čopič (Tools → Paint Brush)**: Obdelava binarne maske z ročnim obrisovanjem.

- **Povezane komponente (Mask Filters → Connected components)**: Izločanje željenih struktur iz binarne slike.
- **Glajenje slike (Data Filters → Gaussian blur)**: Odstranjevanje šuma z glajenjem slike.
- **Logične operacije (Mask Filters → Boolean)**: Logične operacije med dvema binarnima slikama: konjunkcija, disjunkcija in negacija.

Naloge

Pregledovanje 3D medicinskih slik je ključno za analizo zdravih in odkrivanje patoloških struktur v človeškem telesu ter kasnejše diagnosticiranje. Pri tem so nam v veliko pomoč prikazovalniki 3D slik in različna avtomatska in polavtomatska orodja.

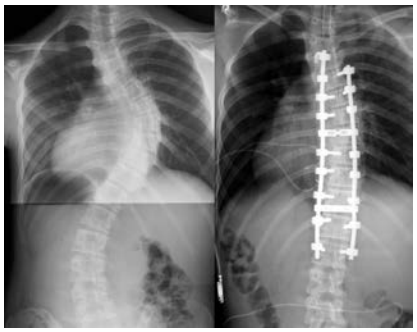
1. V BrainSeg3D odprite CT sliko abdomna `ct-abdomen-liver.nrrd` in s pomočjo predstavljenih orodij izluščite maske žilja, jeter in tretje poljubne strukture. Prikažite si jih v 3D prostoru.
2. V mapi `aneurysm_detection` se nahaja pet 3D-DSA slik možganskega žilja `3d-dsa-brain-aneurysm#.nrrd`. Na vsaki sliki razgradite žilje, si ga prikažite v 3D prostoru in določite število anevrizem na sliki. Slike lahko vsebujejo med 0 in 4 anevrizme.
3. V mapi `lesion_detection` odprite MR sliko možganov `mr-flair-head-ms.nrrd` pacienta z multiplo sklerozo. V 64. rezini obrišite lezije bele možganovine.
4. Povečanje/zmanjšanje lezij bele možganovine spremljamo skozi čas in tako določamo napredek bolezni. Odprite sliki `mr-flair1-brain-ms-longit1.nrrd` in `mr-flair2-brain-ms-longit1.nrrd` in obrišite vse spremembe lezij, ki jih lahko najdete.

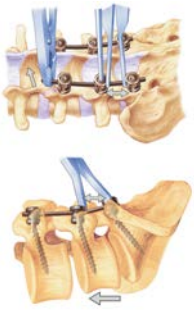
Univerza v Ljubljani, Fakulteta za elektrotehniko
Analiza medicinskih slik (AMS)

Poravnava 3D in 2D slik

Navodila

Poravnava preoperativno zajetih 3D medicinskih slik na medoperativno zajete 2D slike omogoča medoperativno določanje položaja anatomije v 3D prostoru pacienta in prenos ter fuzijo operativnih načrtov. Ta tehnologija je zelo uporabna pri posegu za stabilizacijo hrbtenice, pri kateri kirurg v posamezno vretence vstavi pedikularni vijak in nato vijake poveže s kovinsko palico tako da izravna oz. stabilizira hrbtenico (Slika 1). Pred posegom kirurg na osnovi 3D slike računalniške tomografije (CT[1]) preuči stanje hrbtenice in določi trajektorijo vstavljanja pedikularnih vijakov (Slika 2), s pomočjo medoperativne žive rentgenske slike pa ta načrt prenese v prostor bolnika. Pri vaji boste načrtali in ovrednotili postopek za poravnavo 3D CT in 2D rentgenskih slik vretenc in preko poravnave slik prenesli predoperativni načrt vstavljanja pedikularnega vijaka iz 3D CT slike na 2D rentgensko sliko.





Slika 1: Pacient z degenerativno skoliozo *levo* in stabilizacija hrbtenice s pedikularnimi vijaki *desno*.

Slika 2: 3D model vretenca izluščen iz predoperativne CT slike in načrtovana trajektorija vstavljanja pedikularnega vijaka.

Postopek prileganja 3D slike na 2D sliko imenujemo **3D-2D poravnava**. 3D-2D poravnava je proces iskanja optimalne geometrijske preslikave $\mathcal{T} : \mathbb{R}^3 \rightarrow \mathbb{R}^3$, ki 3D model oz. sliko V nekega objekta preslika v tako lego, ki je *skladna* s projekcijo istega objekta na zajeti 2D sliki P. Glavni izziv pri poravnavi 3D in 2D slik je prostorska neskladnost slikovne informacije (3D vs. 2D), ki jo lahko odpravimo na dva načina: 1) **s projekcijo 3D informacije v 2D slikovni prostor** ali 2) z rekonstrukcijo 3D slike iz različnih 2D projekcij. Pri vaji boste načrtali postopek poravnave s projekcijo 3D slike v 2D ($\mathcal{P} : \mathbb{R}^3 \rightarrow \mathbb{R}^2$) in maksimizacijo mere podobnosti MP med projekcijo $\mathcal{P}(V)$ in 2D sliko P.

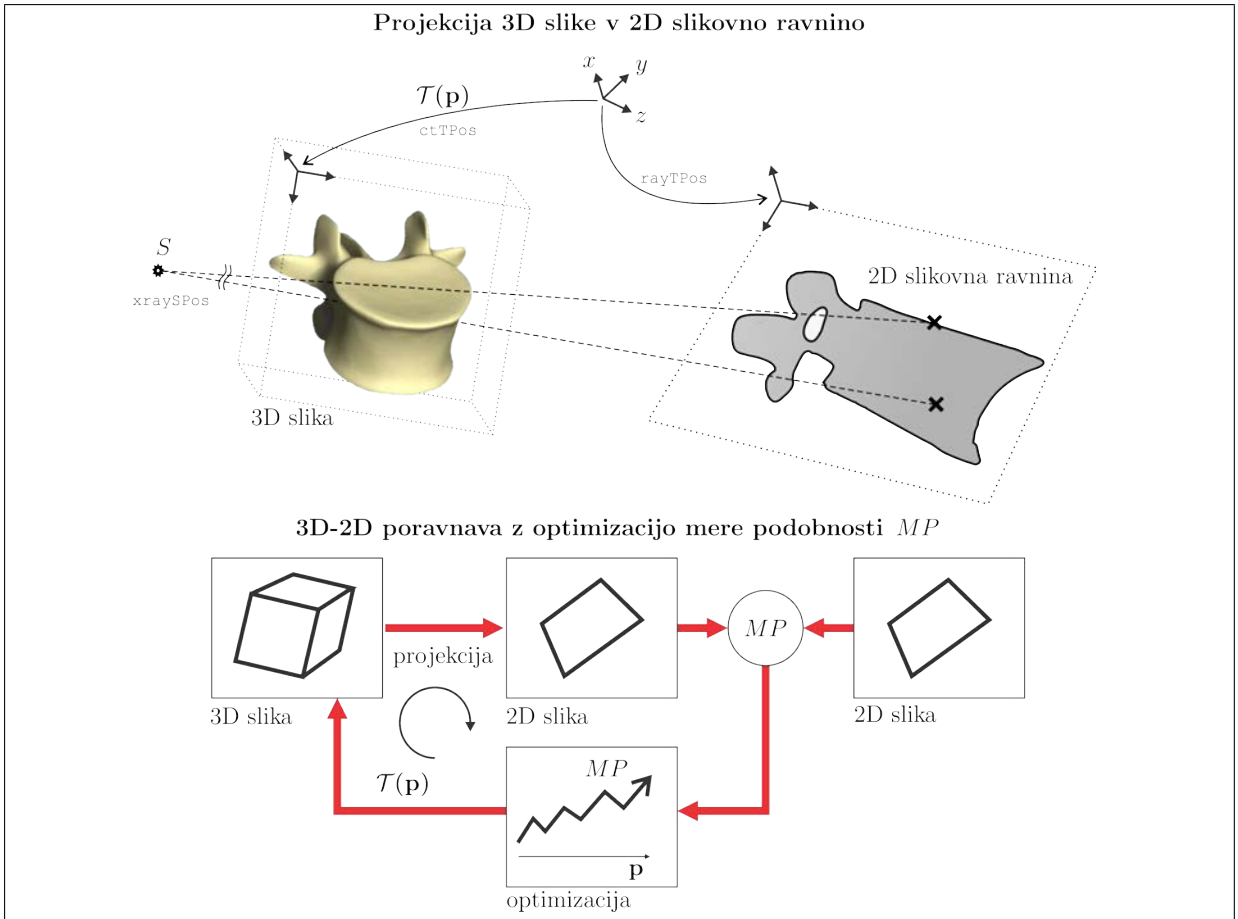
Način projekcije zavisi od oblike 3D informacije. Pri vaji bo dana 3D CT slika, v kateri vsak slikovni element predstavlja atenuacijski koeficient $\mu(\mathbf{x})$ zajete 3D scene. Za potrebe poravnave 3D in 2D slik boste s pomočjo 3D CT slike simulirali 2D rentgenske slike, pri katerih se projekcija iz 3D prostora v 2D sliko izraža kot integral atenuacijskega koeficienta vzdolž premice od izvora rentgenskih žarkov S do 2D slikovne ravnine $P = \int_l \mu(l)dl$. Tovrstno projekcijo 3D CT slike imenujemo digitalno rekonstruirani rentgenski posnetek (DRR[2]). Projekcija in pripadajoče količine so prikazane na sliki 3.

Prostorsko ujemanje med projekcijo $\mathcal{P}(V)$ in 2D sliko P ovrednotimo z mero podobnosti MP; to je skalarna funkcija, ki zavzame optimum, ko se položaji korespondenčnih struktur v $\mathcal{P}(V)$ in P medsebojno prekrivajo. Za poravnavo slik se pogosto uporablja **medsebojna informacija** MI[3]:

$$MI(I, J) = H(I) + H(J) - H(I, J)\,,$$

kjer sta $H(I)$ in $H(J)$ entropija referenčne $I(x, y)$ in lebdeče $J(x, y)$ slike, $H(I, J)$ pa njuna skupna

[1] CT – ang. Computed Tomography
[2] DRR – ang. Digitally Reconstructed Radiograph
[3] MI – ang. Mutual Information



Slika 3: Projekcija predoperativne CT slike na ravnino 2D detektorja (*zgoraj*) in postopek 3D-2D poravnave z optimizacijo mere podobnosti (*spodaj*).

entropija:

$$H(I) = -\sum_{s_I=0}^{L-1} p_I(s_I)\log p_I(s_I) \quad \text{in} \quad H(J) = -\sum_{s_J=0}^{L-1} p_J(s_J)\log p_J(s_J)\,,$$
$$H(I,J) = -\sum_{s_I=0}^{L-1}\sum_{s_J=0}^{L-1} p_{IJ}(s_I,s_J)\log p_{IJ}(s_I,s_J)\,,$$

kjer spremenljivki s_I in s_J označujeta soležne, diskretne sivinske vrednosti referenčne slike $I(x,y)$ in lebdeče slike $J(x,y)$, L pa število diskretnih vrednosti. Verjetnostni porazdelitvi $p_I(s_I)$ in $p_J(s_J)$ ter skupno porazdelitev $p_{IJ}(s_I,s_J)$ dobimo iz pripadajočih normaliziranih histogramov $h_I(s_I)$, $h_J(s_J)$ ter $h_{IJ}(s_I,s_J)$:

$$p_I(s_I) = \frac{h_I(s_I)}{N\cdot M}\,, \quad p_J(s_J) = \frac{h_J(s_J)}{N\cdot M}\,, \quad p_{IJ}(s_I,s_J) = \frac{h_{IJ}(s_I,s_J)}{N\cdot M}\,,$$

kjer N in M predstavljata število elementov vzdolž x in y osi slike.

Tekom poravnave 3D in 2D slike nam izbrana optimizacijska metoda iterativno spreminja parametre $\mathbf{p}$ geometrijske preslikave $\mathcal{T} = \mathcal{T}(\mathbf{p})$tako, da maksimizira mero podobnosti:

$$\mathbf{p}^* = \text{argmax}_{\mathbf{p}}\, MP\Big(\mathcal{P}\big(V[\mathcal{T}(\mathbf{p})]\big), P\Big)\,,$$

kjer so $\mathbf{p}^*$ optimalni parametri preslikave $\mathcal{T}(\mathbf{p})$. Za poravnavo 3D in 2D slik vretenc boste uporabili togo preslikavo, ki je v 3D definirana s šestimi parametri $\mathbf{p} = [t_x, t_y, t_z, \alpha, \beta, \gamma]^{\mathrm{T}}$. Postopek poravnave 3D in 2D slik je prikazan na sliki 3.

Naloge

Gradivo za vajo vsebuje datoteki `ct.nrrd` in `xray.nrrd`, ki podajata 3D CT sliko ledvenega vretenca L3 in 2D rentgensko sliko ledvenih vretenc L1–L5. 3D CT slika ima dimenzije $97 \times 95 \times 39$ in je zapisana z nepredznačenimi 8-bitnimi celimi števili, 2D rentgenska slika pa ima dimenzije 446×446 in je zapisana z nepredznačenimi 16-bitnimi celimi števili. Geometrijski preslikavi CT in rentgenske slike iz prvega slikovnega elementa z indeksom (0, 0) v referenčni koordinatni sistem sta dani s homogenima matrikama:

$$T_{ct} = \begin{bmatrix} 1 & 0 & 0 & 19.9610 \\ 0 & 1 & 0 & 23.7891 \\ 0 & 0 & 1 & 164.0000 \\ 0 & 0 & 0 & 1.0000 \end{bmatrix},$$

$$T_{Xray} = \begin{bmatrix} -0.2925 & -0.0510 & -0.9549 & 397.8680 \\ -0.9542 & 0.0809 & 0.2879 & 192.7720 \\ 0.0625 & 0.9954 & -0.0723 & -107.8180 \\ 0 & 0 & 0 & 1.0000 \end{bmatrix},$$

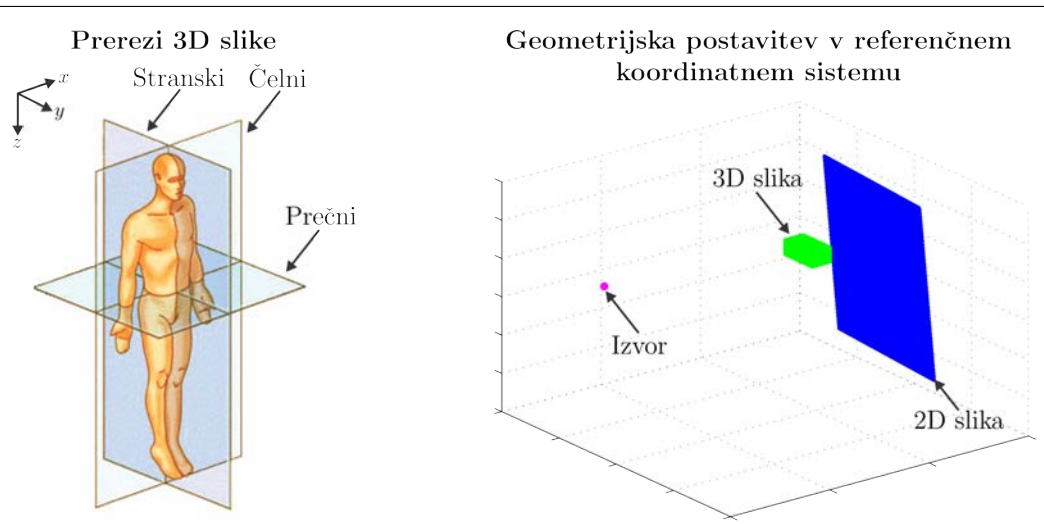
Položaj izvora rentgenskih žarkov v referenčnem koordinatnem sistemu je v točki:

$$\mathbf{s} = \begin{bmatrix} -648.4710 \\ 285.4830 \\ 117.6120 \end{bmatrix}$$

Korak vzorčenja 3D in 2D slik je 1 milimeter. Geometrijske razmere in količine so ponazorjene v sliki 3, *zgoraj*. Dana je tudi skripta `amslib.py`, ki vsebuje funkcije, ki jih boste potrebovali za izvedbo vaje.

1. Za vsako od danih slik ustvarite spremenljivko tipa `dict`, npr. `ct` in `Xray`, in pod ključem `'img'` vnesite sivinske vrednosti slike in pod ključem `'TPos'` pripadajočo preslikavo v referenčnem koordinatnem sistemu (`ctTPos` in `xrayTPos`), za 2D rentgensko sliko vnesite pod ključem `'SPos'` še položaj izvora rentgenskih žarkov (`xraySPos`).

2. Ustvarite vzorčni mreži 2D rentgenske in 3D CT slike s pomočjo ukaza `numpy.meshgrid()`. Točke na vzorčnih mrežah preslikajte v referenčni koordinatni sistem s pripadajočimi 3D togimi preslikavami `Xray['TPos']` in `ct['TPos']`. Preslikane točke 3D in 2D vzorčnih mrež in položaj izvora rentgenskih žarkov `Xray['SPos']` prikažite s pomočjo ukaza `Axes3D.scatter()`. Preverite pravilnost dobljene geometrijske postavitve izvora `s`, 3D in 2D slike v referenčnem koordinatnem sistemu s pomočjo slike 4.



Slika 4: Geometrijska postavitev izvora in vzorčnih točk 3D in 2D slike v referenčnem koordinatnem sistemu.

3. Napišite funkcijo, ki preslika poljubno točko `iPos` v 3D prostoru na 2D slikovno ravnino glede na izvor rentgenskih žarkov `s` (`xraySPos`):

```
def mapPointToPlane( iPos, Xray ):
        return oPos
```

   kjer je `iPos` matrika $N \times 3$, `Xray` pa spremenljivka s podatki o 2D rentgenski sliki. Funkcija vrne točke v matriki `oPos` v obliki matrike $N \times 3$. V ravnino preslikane točke izračunate kot presečišča 2D slikovne ravnine s premicami, ki izvirajo v izvoru rentgenskih žarkov in gredo skozi točke `iPos`. Preverite delovanje funkcije tako, da preslikate koordinate oglišč 3D slike na 2D slikovno ravnino in točke vrišete v geometrijsko postavitev iz prejšnje naloge.

4. V Python skripti `amslib.py` imate dano funkcijo za stožčasto projekcijo 3D CT slike v 2D slikovno ravnino, ki simulira delovanje rentgena (DRR):

```
def project3DTo2D( ct, Xray, iStep ):
        return oDRR, oMask
```

   kjer sta `ct` in `Xray` spremenljivki s podatki o 3D CT in 2D rentgenski sliki, `iStep` pa je korak vzorčenja v *milimetrih* vzdolž vsake premice od izvora rentgenskih žarkov do presečišča z 2D slikovno ravnino. Funkcija vrne 2D DRR sliko v spremenljivki `oDRR` in 2D masko DRR slike v spremenljivki `oMask`, ki imata dimenzije 446×446 (enako kot `Xray['img']`). 2D maska DRR slike označuje tiste točke 2D slikovne ravnine, za katere premica do izvora rentgenskih žarkov seče 3D CT sliko. Preizkusite delovanje funkcije s pomočjo spremenljivk `ct` in `Xray` in ustvarite DRR oz. projekcijo 3D CT slike. Preizkusite vpliv dolžine koraka `iStep` na projicirano sliko.

5. Napišite funkcijo za 3D togo preslikavo 3D CT slike okoli njenega središča:

```
def rigidTrans( ct, iPar ):
        return oTrans
```

   kjer vhodna spremenljivka `ct` predstavlja podatke 3D CT slike v obliki slovarja, `iPar` pa šestvrstični vektor $\mathbf{p} = [t_x, t_y, t_z, \alpha, \beta, \gamma]^{\mathrm{T}}$ s parametri 3D toge preslikave, kjer so rotacije definirane glede na center 3D slike. Izhodna spremenljivka `oTrans` predstavlja 4×4 homogeno matriko preslikave koordinat 3D CT slike.

   Razširite funkcijo za 2D projekcijo oz. DRR sliko tako, da bo imela dodatni vhodni parameter `iPar`, npr. `project3DTo2D( ..., iPar )`, kjer je `iPar` šestvrstični vektor s parametri 3D toge preslikave 3D CT slike okoli njenega središča. Ustvarite in prikažite 2D projekcijske slike za različne vrednosti parametrov in ovrednotite pravilnost rešitve.

6. V Python skripti `amslib.py` imate dano funkcijo za izračun vrednosti medsebojne informacije med dvema slikama:

```
def mutualInformation( iImageI, iImageJ, iBins, iSpan ):
        return oMI
```

   kjer sta `iImageI` in `iImageJ` 2D sivinski sliki, `iBins` število intervalov za izračun histograma, `iSpan` pa seznam z območjem sivinskih vrednosti za izračun histograma (min, max). Funkcija vrne skalar `oMI`. Preizkusite delovanje funkcije tako, da izračunate medsebojno informacijo med projicirano sliko in 2D rentgensko sliko, pri tem pa uporabite le sivinske vrednosti na področju maske projicirane slike. Preučite vpliv števila intervalov `iBins` na vrednost medsebojne informacije.

   Izračunajte in izrišite vrednosti `oMI` med 2D projekcijami in 2D rentgensko sliko tako, da spreminjate le parameter α toge preslikave. Ovrednotite povezavo med medsebojno informacijo in dejansko podobnostjo 2D rentgenske in 2D projekcijskimi slikami. Katere vrednosti medsebojne informacije, višje ali nižje, odražajo dejansko podobnost med 2D rentgensko in 2D projekcijskimi oz. DRR slikami?

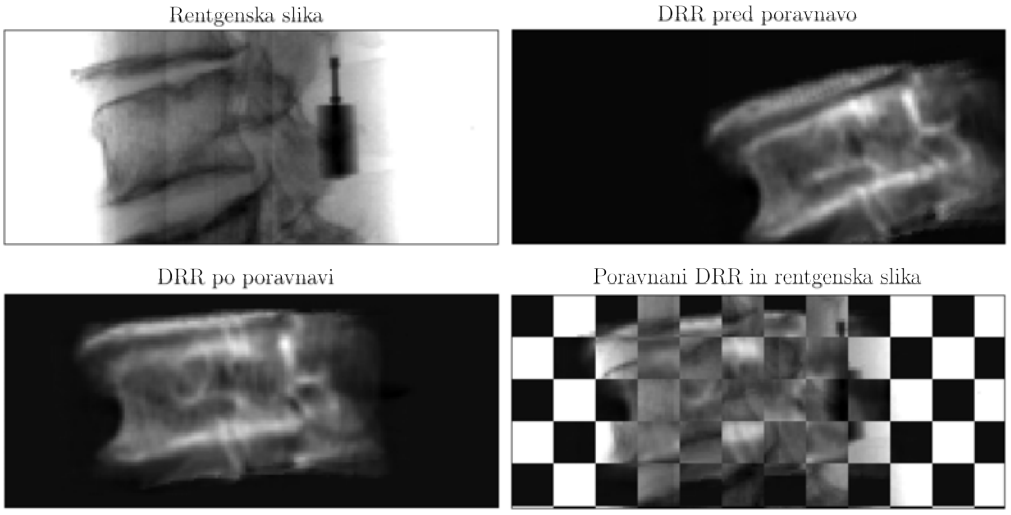
7. Načrtajte avtomatski postopek za poravnavo 3D in 2D slik tako, da ustvarite kriterijsko funkcijo:

```
def criterionFcn( iPar, ct, Xray ):
        return oMP
```

ki pri podanih parametrih 3D toge preslikave `iPar` izračuna vrednost medsebojne informacije MI med DRR in 2D rentgensko sliko. Optimalne vrednosti parametrov $\mathbf{p}^*$ toge preslikave določite s pomočjo Powellove optimizacije, naprimer:

```
# definicija funkcije MP(p)
oMP = lambda iPar : criterionFcn( iPar, ct, Xray )
# klic numerične optimizacije (minimizacije)
from scipy.optimize import minimize
iPar_opt = minimize( fun=oMP, x0=np.zeros(6,1), method='Powell', tol=1e-5)
```

Preizkusite delovanje postopka na podatkih `ct` in `Xray` tako, da optimizirate le parameter α. Uspešnost postopka poravnave lahko preverite kot je prikazano na sliki 5.



Slika 5: Primer kvalitativnega preverjanja uspešnosti 3D-2D poravnave.

Dodatne naloge

Dodatne naloge naj služijo za poglobitev spretnosti programiranja, boljšemu razumevanju snovi in vsebine vaje in spoznavanju dodatnih načinov za obdelavo in analizo medicinskih slik. Opravljanje dodatnih nalog je neobvezno, vendar pa priporočljivo, saj je na nek način to priprava na zagovor laboratorijskih vaj.

1. Izračunajte in izrišite vrednosti `oMI` med 2D projekcijami in 2D rentgensko sliko tako, da spreminjate le po en parameter toge preslikave naenkrat. Translacije t_x, t_y in t_z spreminjajte od -20 do 20 milimetrov s korakom 2 milimetra, rotacije α, β in γ pa od -10 do 10° s korakom 1° in narišite ter ustrezno označite grafe $MI(t_x)$, $MI(t_y)$, $MI(t_z)$, $MI(\alpha)$, $MI(\beta)$, $MI(\gamma)$. Ali imajo poteki jasno izražen optimum pri optimalni poravnavi, tj. pri referenčnih parametrih $\mathbf{p}^* = [0, 0, 0, 0, 0, 0]$?

2. Kolikšni sta teoretično minimalna in maksimalna vrednost medsebojne informacije za dani par slik?

3. Izvedite poravnavo slik z uporabo vseh šestih parametrov toge preslikave. Navedite vrednosti optimalnih parametrov $\mathbf{p}^*$ toge preslikave $T_{ct}*$, ki rezultirajo v uspešni poravnavi 3D in 2D slik, in prikažite pripadajočo DRR sliko. Preizkusite različne vrednosti začetnih parametrov in približno ocenite območje konvergence za vsak posamezen parameter toge preslikave t_x, t_y, t_z, α, β in γ .

4. Glajenje slik pred poravnavo lahko ugodno vpliva na potek mere podobnosti tako, da se poveča območje konvergence. Preverite to hipotezo na dejanskem paru `ct` in `Xray` slike.

5. Parametri referenčne toge preslikave, ki predstavljajo optimalno poravnavo so pri vrednostih $\mathbf{p}^* = [0, 0, 0, 0, 0, 0]$. Za parametere poravnave, ki jih dobite s pomočjo optimizacije izračunajte vrednost

srednje napake med tarčnimi točkami[4] (mTRE), pri čemer naj bodo točke kar oglišča 3D slike. Kaj predstavlja vrednost mTRE in v katerih enotah se izraža? Katero mejno vrednost mTRE bi bilo smiselno uporabiti v kontekstu klinične aplikacije vstavljanja pedikularnega vijaka?

6. Načrtovana vstopna točka in smer pedikularnega vijaka glede na koordinatni sistem 3D CT slike je podana v spremenljivkah $\mathbf{t}_e$ in $\mathbf{v}_e$:

$$\mathbf{t}_e = \begin{bmatrix} 44 \\ 25 \\ 13 \end{bmatrix} \quad \text{in} \quad \mathbf{v}_e = \begin{bmatrix} 0.8638 \\ 0.5039 \\ 0 \end{bmatrix}.$$

   Prenesite trajektorijo pedikularnega vijaka iz prostora 3D CT slike v prostor X-ray slike s pomočjo izhodiščne dane 3D-2D poravnave teh slik. Uporabite vrednosti referenčnih parametrov $\mathbf{p}^*$ toge preslikave in preslikajte načrtovano trajektorijo pedikularnega vijaka v 2D rentgensko sliko. Prikažite 2D rentgensko sliko in vrišite superponirano trajektorijo pedikularnega vijaka, preslikano in 3D CT slike, pri čemer trajektorijo vijaka prikažite kot daljico obarvano zeleno.

7. Nadaljujte prejšnjo nalogo tako, da trajektorijo pedikularnega vijaka vrišete še za primer 3D-2D poravnave, ko parametre preslikave pridobite z optimizacijo. Trajektorijo vrišite npr. z rdečo barvo in primerjate rezultat z referenčno razgradnjo. Je rešitev dovolj natančna?

[4] ang. *mean target registration error (mTRE)*

Univerza v Ljubljani, Fakulteta za elektrotehniko
Analiza medicinskih slik (AMS)

Razgradnja slik z upragovanjem

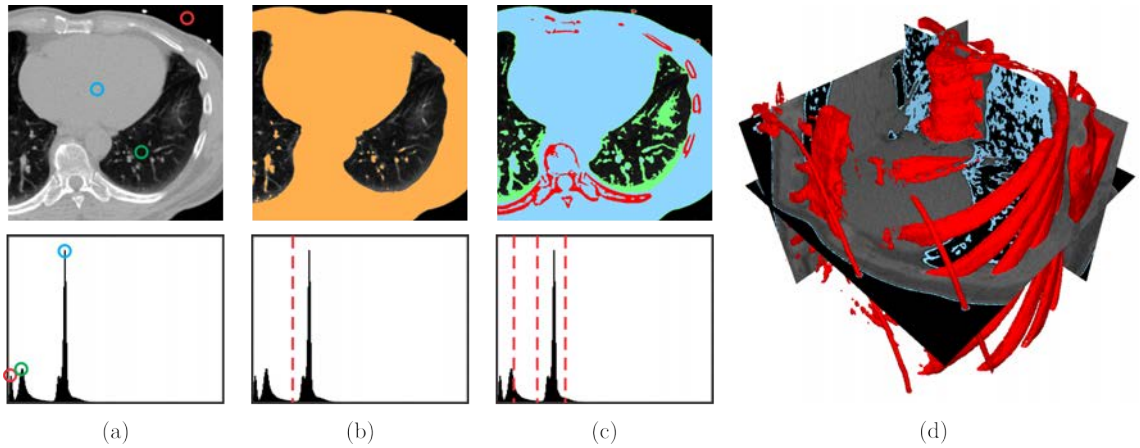
Navodila

Razgradnja[1] združuje postopke, s katerimi sliko razdelimo na osnovna področja oziroma strukture. Razgradnja je eden temeljnih postopkov pri analizi medicinskih slik, saj njena uporaba sega od detekcije zdravih in patoloških struktur, kvantitativnih meritev, analize variacije oblike razgrajenih struktur itn.

Stopnja podrobnosti, ki jih želimo izluščiti iz slike s pomočjo razgradnje, je odvisna od namena uporabe, omejena pa je z vsebino in kakovostjo slik, predvsem z ločljivostjo, ostrino, šumom, kontrastom in sivinskimi nehomogenostmi. Zaradi tega je razgradnja eden najbolj zahtevnih postopkov obdelave in analize slik, ki ga je najtežje popolnoma avtomatizirati.

Če področja v dani sliki ločujejo dovolj velike spremembe sivinskih vrednosti, potem lahko uporabimo razgradnjo z upragovanjem[2]. Sliko lahko razgradimo tudi tako, da so področja glede na določen kriterij oziroma značilnico dovolj homogena, naprimer glede na entropijo ali standardno deviacijo sivinske vrednosti, teksture, itn. Tretja pogosto uporabljana možnost razgradnje slike je geometrijska poravnava z modelom objekta zanimanja, ki predstavlja področja slike.

Slika 1a prikazuje prečno rezino CT[3] slike prsnega koša, sliki 1bc pa rezino razgrajeno v enojnim in trojnim upragovanjem. Vizualizacija razgrajenih struktur v 3D je prikazana v sliki 1d.



Slika 1: (a) Prečna rezina CT slike prsnega koša s pripadajočim sivinskim histogramom. (b) Razgrajena CT slika z avtomatsko določenim pragom s postopkom Otsu. (c) Razgrajena CT slika z avtomatsko določenimi tremi pragovi z rekurzivnim postopkom Otsu. (d) 3D vizualizacija razgrajenih struktur v CT slike.

Upragovanje slike omogoča neposredno določanje področij slike na podlagi sivinskih vrednosti. Predpostavimo, da so sivinske vrednosti slike $f(x, y)$ s svetlejšo strukturo na temnejšem ozadju porazdeljene v dve množici z podobnimi vrednostmi, tako da je histogram bimodalen. V tem primeru lahko objekt ločimo od ozadja s preprostim upragovanjem sivinskih vrednosti slike (slika 1b). Razgrajeno sliko $g(x, y)$ dobimo tako, da ozberemo prag τ in nato sivinske vrednosti slike $f(x, y)$ razdelimo v dve skupini in vsako skupino označimo z različno sivinsko vrednostjo:

$$g(x,y) = \begin{cases} L-1 & \text{ko je } f(x,y) > \tau, \\ 0 & \text{sicer.} \end{cases} \tag{1}$$

Kadar je vrednost praga τ enaka za vse slikovne elemente slike, izvajamo **globalno upragovanje**. V splošnem pa je vrednost praga τ lahko funkcija prostorskih koordinat ($\tau = \tau(x, y)$); v tem primeru govorimo o **prilagodljivem upragovanju slik**. Kadar je na sliki več struktur z različnimi sivinskimi

[1] razgradnja: ang. *image segmentation*
[2] upragovanje: ang. *thresholding*
[3] CT: ang. *Computed Tomography*

vrednostmi je smiselno izvesti **večkratno upragovanje** ter na ta način sliko razgraditi na več struktur (slika 1c)

Razgradnjo z upragovanjem lahko avtomatiziramo tako, da z avtomatskim postopkom poiščemo ustrezen prag. Obstaja veliko različnih postopkov, ki temeljijo na analizi oblike histograma (iskanje vrhov, dolin in velikih sprememb), postopki z rojenjem sivinskih vrednosti oz. modeliranju histograma z mešanico enomodalnih (npr. Gaussovih) porazdelitev, postopki z maksimizacijo entropije področij ali križne entropije med originalno in binarno sliko, itn.

Eden od bolj uveljavljenih postopkov je postopek **iskanja optimalnega praga Otsu**, ki je v osnovi postopek rojenja z minimizacijo variance znotraj področij ali, kar je ekvivalentno, z maksimizacijo variance med področji. Izraze za variance lahko določimo na podlagi analize verjetnostne porazdelitve $p_f(l)$, ki jo dobimo iz pripadajočega normaliziranega histograma $h_f(l)$:

$$p(l) = \frac{h(l)}{N \cdot M}\,, \quad p(l) \geq 0, \sum_{l=L_{min}}^{L_{max}} p(l) = 1\,.$$

N in M predstavljata število elementov vzdolž x in y osi slike $f(x, y)$, L_{min} in L_{max} pa minimalno in maksimalno sivinsko vrednost. Če želimo slik razgraditi v področji A in B na podlagi svetlosti, potem določimo prag τ z minimizacijo variance znotraj področij σ_w^2, tj. z minimizacijo izraza:

$$\sigma_w^2(\tau) = \omega_A(\tau)\sigma_A^2(\tau) + \omega_B(\tau)\sigma_B^2(\tau)\,, \tag{2}$$

pri čemer sta ω_A in ω_B verjetnosti področij A in B, dobljeni na podlagi praga τ, in σ_A^2 in σ_B^2 pripadajoči varianci. Verjetnosti področij dobimo kot:

$$\omega_A(\tau) = \sum_{l=L_{min}}^{\tau-1} p_f(l) \quad \omega_B(\tau) = \sum_{l=\tau}^{L_{max}} p_f(l)\,.$$

Namesto izraza v (2) lahko maksimiziramo varianco med področji:

$$\begin{aligned} \sigma_b^2(\tau) &= \sigma^2 - \sigma_w^2(\tau) \\ &= \omega_A(\tau)(\mu_A(\tau) - \mu)^2 + \omega_B(\tau)(\mu_B(\tau) - \mu)^2 \\ &= \omega_A(\tau)\,\omega_B(\tau)\,(\mu_A(\tau) - \mu_B(\tau))^2\,, \end{aligned} \tag{3}$$

pri čemer smo upoštevali, da za vse vrednosti τ velja $\omega_A(\tau)+\omega_B(\tau) = 1$ in $\omega_A(\tau)\mu_A(\tau)+\omega_B(\tau)\mu_B(\tau) = \mu$. Pripadajoče povprečne vrednosti izračunamo kot:

$$\mu_A(\tau) = \frac{1}{\omega_A(\tau)} \sum_{l=L_{min}}^{\tau-1} l \cdot p_f(l)\,, \quad \mu_B(\tau) = \frac{1}{\omega_B(\tau)} \sum_{l=\tau}^{L_{max}} l \cdot p_f(l)\,, \quad \mu = \sum_{l=L_{min}}^{L_{max}} l \cdot p_f(l)\,.$$

Celoten Otsu postopek avtomatskega določanja praga je naslednji:

1. Izračunaj histogram sivinskih vrednosti $h(l)$ in verjetnost $p(l)$ za vse sivinske vrednosti l,
2. Za vse vrednosti praga $\tau = L_{min}, \ldots, \Delta\tau, \ldots, L_{max}$:
   (a) Izračunaj $\omega_i(\tau)$ in $\mu_i(\tau)$, $i = \{A, B\}$,
   (b) Izračunaj $\sigma_b^2(\tau)$ po enačbi (3),
3. Poišči prag pri maksimumu $\sigma_b^2(\tau)$.

Otsu postopek lahko enostavno razširimo za večkratno upragovanje tako, da dodamo ustrezne člene v izraz (3). Naprimer, razgradnjo v tri področja A, B in C potrebujemo dva pragova τ_1, τ_2, maksimiziramo pa izraz:

$$\sigma_b^2(\tau_1, \tau_2) = \omega_A(\tau_1)(\mu_A(\tau_1) - \mu)^2 + \omega_B(\tau_1, \tau_2)(\mu_B(\tau) - \mu)^2 + \omega_C(\tau_2)(\mu_C(\tau_2) - \mu)^2\,. \tag{4}$$

Večkratno upragovanje lahko naredimo tudi s hierarhično rekurzivno delitvijo slike in podpodročij z osnovnim Otsu algoritmom.

Naloge

Gradivo za vajo vsebuje dve sliki, prva `ct-thorax.nrrd` je 2D slika prečne rezine CT slike prsnega koša, druga `ct-vert.nrrd` pa 3D CT slika prsnih vretenc. Obe sliki sta sivinski v 16-bitnem nepreznačenem zapisu.

1. Napišite funkcijo za določanje praga s postopkom Otsu. Funkcija naj ima podpis:

   ```
   def otsuThreshold(iImage, iBins):
           return oThr
   ```

   kjer je `iImage` vhodna slika dimenzij $X \times Y$, `iBins` pa število predalov za izgradnjo histograma sivinskih vrednosti. Funkcija naj vrne skalarno vrednost `oThr` z vrednostjo praga. Preizkusite delovanje funkcije na 2D prečni rezine CT slike `ct-thorax.nrrd` in prikažite razgrajeno sliko.

2. Napišite funkcijo za večkratno hierarhično upragovanje s postopkom Otsu. Funkcija naj ima podpis:

   ```
   def otsuMultiThresholdRecursive(iImage, iBins, iNumLevels):
           return oThr
   ```

   kjer je `iImage` vhodna slika dimenzij $X \times Y$, `iBins` število predalov za izgradnjo histograma sivinskih vrednosti, `iNumLevels` pa število hiearhičnih delitev slike na podpodročja. Funkcija naj vrne seznam vrednosti tipa `list` v spremenljivki `oThr` z po velikosti urejenimi vrednostmi pragov. Število pragov zavisi od števila nivojev $D > 0$ in znaša $2^D - 1$. Preizkusite delovanje funkcije na 2D prečni rezine CT slike `ct-thorax.nrrd` in prikažite razgrajeno sliko.

Dodatne naloge

Dodatne naloge naj služijo za poglobitev spretnosti programiranja, boljšemu razumevanju snovi in vsebine vaje in spoznavanju dodatnih načinov za obdelavo in analizo medicinskih slik. Opravljanje dodatnih nalog je neobvezno, vendar pa priporočljivo, saj je na nek način to priprava na zagovor laboratorijskih vaj.

1. Razgradite sliko `ct-thorax.nrrd` z implementacijo Otsu postopka v Python knjižnici `SimpleITK`, in sicer:

   ```
   oImageT = itk.OtsuThreshold(iImage)
   ```

   kjer je `iImage` vhodna slika in `oImageT` izhodna razgrajena slika. Obe sliki sta tipa `itk.Image`. Prikažite razgrajeno sliko. Ali dobite enak rezultat z vašo implementacijo pri Nalogi 1?

2. Razgradite sliko `ct-vert.nrrd` z vašo najljubšo implementacijo Otsu postopka. Zaradi vpliva šuma boste opazili, da je razgradnja relativno slaba. Slika 2 prikazuje vpliv predobdelave na razgradnjo.

   Predlagajte in izvedite ustrezno preobdelavo slike. Preverite kakovost razgradnje na prikazu sredinske prečne rezine predobdelane slike ter razgradnjo originalne in predobdelane slike. Kakovost razgradnje lahko preverite tudi s 3D vizualizacijo, npr. z uporabo programa BrainSeg3D kot prikazuje slika 2.

3. Napišite funkcijo za za večkratno upragovanje s postopkom Otsu, pri čemer gleda na zahtevano število pragov ustrezno razširite izraz (3), kot naprimer v (4). Funkcija naj ima podpis:

   ```
   def otsuMultiThreshold(iImage, iBins, iNumThr):
           return oThr
   ```

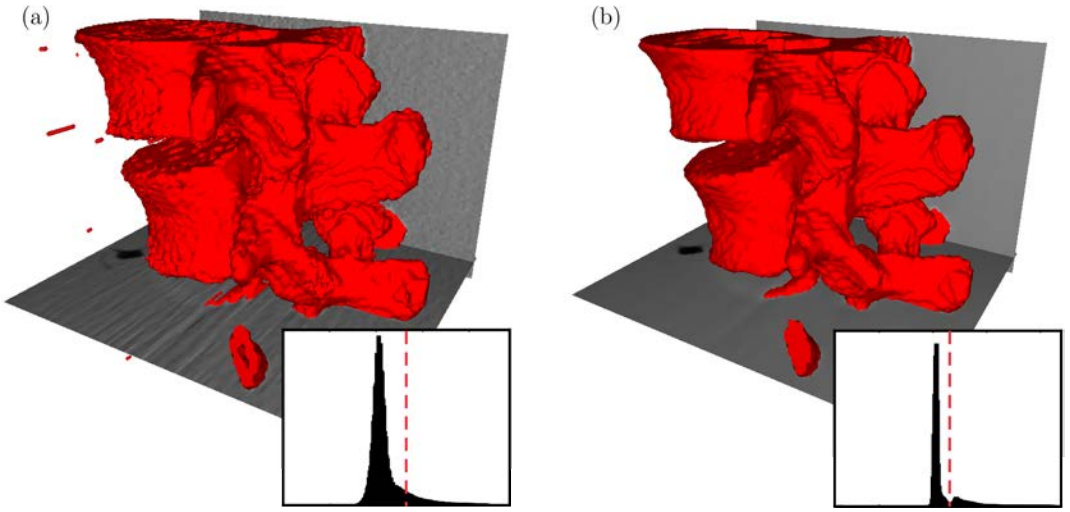
   kjer je `iImage` vhodna slika dimenzij $X \times Y$, `iBins` število predalov za izgradnjo histograma sivinskih vrednosti, `iNumThr` pa število pragov. Funkcija naj vrne seznam vrednosti tipa `list` v spremenljivki `oThr` z po velikosti urejenimi vrednostmi pragov.

   - Preizkusite delovanje funkcije na 2D prečni rezine CT slike `ct-thorax.nrrd` in prikažite sliko razgrajeno z vhodnim parametrom `iNumLevels=3`. Primerjajte in komentirajte kakovost razgradnje z razgradnjo iz Naloge 2.

- Razgradite sliko z ekvivalentno implementacijo Otsu postopka v Python knjižnici `SimpleITK`, in sicer:

  ```
  oImageT = itk.OtsuMultipleThresholds(iImage, iNumThr)
  ```

  kjer je `iImage` vhodna slika, `iNumThr` število pragov in `oImageT` izhodna razgrajena slika. Vhodna in izhodna slika sta tipa `itk.Image`. Prikažite razgrajeno sliko. Ali dobite enak rezultat z vašo implementacijo?



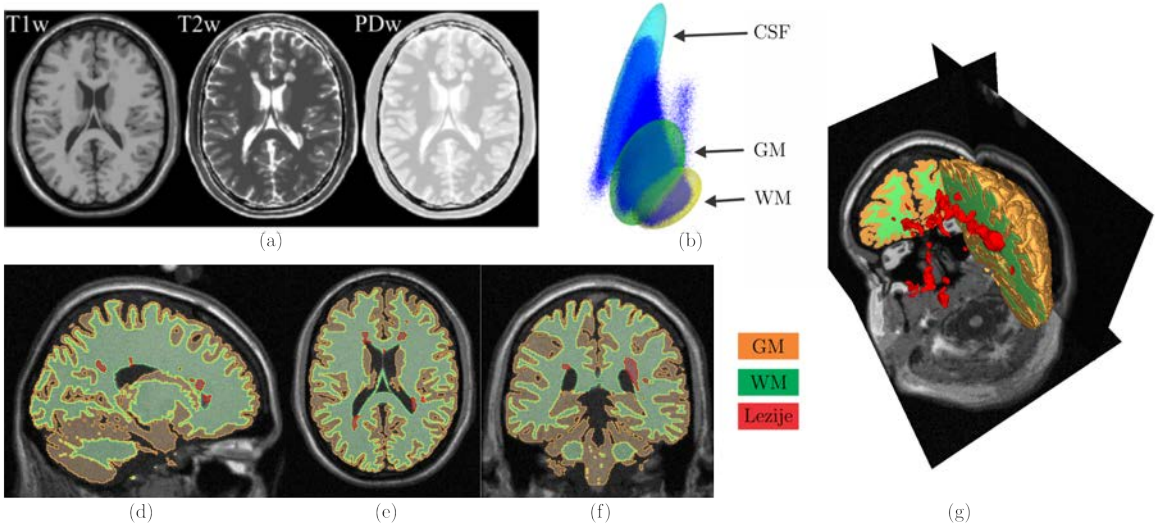
Slika 2: Vizualizacija razgradnje vretenc na (a) originalni CT sliki in (b) CT sliki z obnovo šuma. V pripadajočih sivinskih histogramih je označen prag izračunan z avtomatskim Otsu postopkom.

Univerza v Ljubljani, Fakulteta za elektrotehniko
Analiza medicinskih slik (AMS)

Razgradnja slik z rojenjem

Slikovna diagnostika možganskih bolezni

Tomografsko slikanje z magnetno resonanco (MR) se pogosto uporablja za spremljanje poteka možgansko-žilnih ali nevrodegenerativnih bolezni kot so multipla skleroza (MS), Alzheimerjeva bolezen in druge vrste demence. Potek teh bolezni je namreč povezan z morfološkimi spremembami v možganih, ki se pogosto pojavijo pred drugimi kliničnimi znaki. Morfološke spremembe se najpogosteje pojavijo v beli možganovini (WM[1]), pri čemer gre lahko za reverzibilne ali nereverzibilne poškodbe oz. lezije WM. Kakršnekoli spremembe lezij ali pojav novih lezij so klinično pomemben parameter stanja omenjenih bolezni in se običajno ovrednotijo s pomočjo ročnega obrisovanja in merjenjem števila in volumna lezij. Stanje oz. napredovanje bolezni se odraža tudi v morfoloških spremembah zdravih možganskih struktur (npr. atrofija), zato se merjenje volumna običajno izvaja tudi na strukturah kot so WM, siva možganovina (GM[2]) in likvoju (CSF[3]). Pri posamezni MR preiskavi se zajame več različnih slik, kot so T1-, T2- in PD-uteženе MR slike (slika 1a), za zaznavo, obrisovanje in analizo zdravih možganskih struktur in lezij bele možganovine pa se uporabljajo vse omenjene MR slike. Ker je ročno obrisovanje 3D slik subjektivno, časovno zahtevno in nenatančno je bilo predlaganih kar nekaj avtomatskih postopkov razgradnje MR slik glave. Ti avtomatski postopki MR sliko razgradijo na osnovne možganske strukture (CSF, GM in WM) glede na določen kriterij oz. značilnico, ki je dovolj homogena (sivine, teksture, ...). Zanesljivost in natančnost teh postopkov zavisi predvsem od kakovosti zajetih MR slik, osnovni postopki razgradnje pa temeljijo na odvodih, upragovanju, razvrščanju ali pa na matematičnih, fizičnih ali statističnih modelih (slika 1b). V zadnjem času so se za razgradnjo MR slik uveljavili postopki, ki temeljijo na rojenju sivinskih vrednosti z neparametričnim razvrščanjem. Eden od takih postopkov, ki ga boste za razgradnjo MR slik razvili pri nalogi tej nalogi je postopek k-povprečij[4]. Predno tovrstne postopke lahko uporabimo za potrebe slikovne diagnostike možganskih bolezni jih je potrebno kvantitativno ovrednotiti. To običajno naredimo s primerjavo avtomatsko pridobljenih obrisov z referenčnimi ročnimi obrisi (slika 1defg), ki pa so pridobljeni s statistično analizo in združevanjem večih ročnih obrisov. Pri nalogi bodo dani referečni obrisi, s katerimi boste lahko kvantitativno vrednotili kakovost razgradnje s postopkom k-povprečij.



Slika 1: MR slike glave (a) zajete s T1-, T2- in PD-uteženo MR sekvenco in (b) prostor sivinskih značilnic MR slik z označenimi superponiranimi modeli možganskih struktur (CSF, GM, WM). Razgradnja MR slik z modelom možganskih struktur v (d) stranskem, (e) prečnem, (f) čelnem prerezu 3D MR slike in (g) prikaz v 3D.

[1] WM – ang. White Matter
[2] GM – ang. Gray Matter
[3] CSF – ang. CerebroSpinal Fluid
[4] ang. k-means

Razgradnja 3D MR slik

Za razgradnjo zdravih in bolezenskih možganskih struktur iz MR slik glave se običajno uporablja več sekvenc (npr. T1-, T2- in PD-uteženа, slika 1a), ker se na posameznih sekvencah oz. MR slikah te strukture izražajo z različnimi, a značilnimi sivinskimi vrednostmi. Če so MR slike med seboj prostorsko poravnane, potem sivinske vrednosti več različnih MR slik skupaj tvorijo večdimenzionalen *prostor značilnic*. Večje možganske strukture imajo v MR slikah običajno približno konstantne sivinske vrednosti, zato večja zastopanost teh med seboj podobnih sivinskih vrednosti v prostoru značilnic oblikuje gruče (slika 1b). Analiza gruč[5] v prostoru značilnic MR slik je trenutno eden od uveljavljenih načinov razgradnje MR slik. Eden od osnovnih postopkov za analizo gruč na osnovi katerega lahko izvedemo razgradnjo MR slik z neparametričnim razvrščanjem je postopek k-povprečij.

Naj bodo $I_{T1}(\boldsymbol{x})$, $I_{T2}(\boldsymbol{x})$ in $I_{PD}(\boldsymbol{x})$ 3D sivinske slike, $\boldsymbol{x} = [x_1, x_2, x_3]^{\mathrm{T}}$, ki predstavljajo med seboj prostorsko poravnane T1-, T2- in PD-uteženе MR sekvence. Za neko poljubno točko $\boldsymbol{x}_j$ v MR sekvencah lahko zapišemo vektor sivinskih vrednosti $\boldsymbol{y}_j = [\, I_{T1}(\boldsymbol{x}_j),\, I_{T2}(\boldsymbol{x}_j),\, I_{PD}(\boldsymbol{x}_j)\,]^{\mathrm{T}}$, ki ima dimenzije 3×1. Množice vektorjev $\boldsymbol{y}_j$, $j = 1, \ldots, N$, predstavljajo značilnice na osnovi katerih bomo s postopkom k-povprečij in neparametričnim razvrščanjem načrtali razgradnjo možganskih struktur.

Postopek k-povprečij

Za dano množico značilnic $Y = (\boldsymbol{y_1}, \boldsymbol{y_2}, \ldots, \boldsymbol{y_N})$, kjer je vsaka značilnica d-dimenzionalni vektor, postopek k-povprečij razvrsti teh N značilnic v k različnih množic $\boldsymbol{S} = \{S_1, S_2, \ldots, S_k\}$ $(k \leq N)$ tako, da pri tem minimizira vsoto kvadratov usrediščenih razdalj značilnic znotraj vsake množice:

$$E^2 = \arg\min_{\boldsymbol{S}} \sum_{i=1}^{k} \sum_{\boldsymbol{y_j} \in S_i} \|\boldsymbol{y_j} - \boldsymbol{\mu_i}\|^2 \,,$$

kjer je $\boldsymbol{\mu_i}$ središče oz. povprečje značilnic v množici S_i, $\|\cdot\|$ pa L_2 vektorska norma. Postopek k-povprečij minimizira E^2 na iterativen način tako, da v vsakem koraku t izvede naslednji dve operaciji:

1. Razvrščanje značilnic: Značilnico $\boldsymbol{y}_j$ razvrsti v tisto množico $S_{i^*}^{(t)}$, katere središče $\boldsymbol{\mu}_{i^*}^{(t)}$ je najbližje v smislu Evklidske razdalje:

$$S_{i^*}^{(t)} = \{\boldsymbol{y}_j : \|\boldsymbol{y}_j - \boldsymbol{\mu}_{i^*}^{(t)}\| \leq \|\boldsymbol{y}_j - \boldsymbol{\mu}_i^{(t)}\|,\ \forall i, 1 \leq i \leq k\}\,, \tag{1}$$

kjer vsak $\boldsymbol{y}_j$ pripišemo natanko eni od množic $\boldsymbol{S}^{(t)}$.

2. Posodabljanje središč: Izračunaj nova središča $\boldsymbol{\mu}_i^{(t)}$ kot povprečja značilnic v množici $S_i^{(t)}$:

$$\boldsymbol{\mu}_i^{(t+1)} = \frac{1}{|S_i^{(t)}|} \sum_{\boldsymbol{y}_j \in S_i^{(t)}} \boldsymbol{y}_j \,. \tag{2}$$

Operaciji (1–2) ponavljamo dokler se položaji središč spreminjajo ($\boldsymbol{\mu}_i^{(t+1)} \neq \boldsymbol{\mu}_i^{(t)}, \forall i$) ali do vnaprej določenega maksimalnega števila korakov t_{max}.

Določanje začetnih središč. Pomemben vhodni parameter postopka k-povprečij so začetna središča $\boldsymbol{\mu}_i^{(0)}$, $i = 1, \ldots, k$. Pri nalogi boste uporabili razširjeni postopek k-povprečij++, ki določi tudi začetna središča z naslednjim postopkom:

1. Naključno izberi eno izmed značilnic $\boldsymbol{y}_j$ in jo določi kot prvo središče, $\boldsymbol{\mu}_1^{(0)} = \boldsymbol{y}_j$.
2. Izračunaj Evklidske razdalje vseh značilnic $\boldsymbol{y}_j$ do najbližjih že določenih središč $d_i(\boldsymbol{y}_j) = \|\boldsymbol{y}_j - \boldsymbol{\mu}_i^{(0)}\|, \forall i,\ 1 \leq i \leq i_m$.
3. Določi novo središče $\boldsymbol{\mu}_{i_m+1}^{(0)}$ kot vsoto s kvadratom razdalje $d^2(\boldsymbol{y}_j)$ uteženih značilnic:

$$\boldsymbol{\mu}_{i_m+1}^{(0)} = \frac{\sum_{\boldsymbol{y}_j \in S} d_i^2(\boldsymbol{y}_j) \cdot \boldsymbol{y}_j}{\sum_{\boldsymbol{y}_j \in S} d_i^2(\boldsymbol{y}_j)} \,.$$

4. Ponovi koraka 2 in 3, dokler ne določiš vseh k začetnih središč.

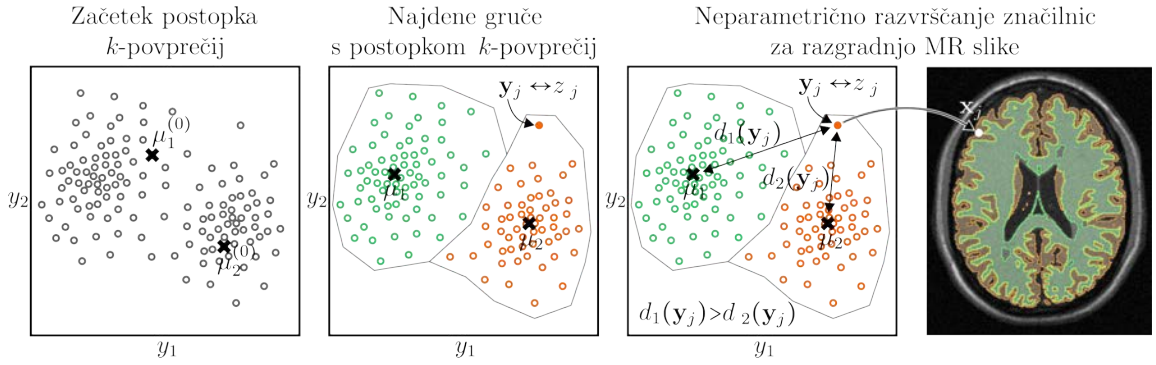
[5] ang. cluster analysis

Razgradnja z neparametričnim razvrščanjem

V nalogi boste uporabili postopek k-povprečij za analizo gruč v prostoru značilnic, določen s sivinskimi vrednostmi 3D slik $I_{T1}(\boldsymbol{x})$, $I_{T2}(\boldsymbol{x})$ in $I_{PD}(\boldsymbol{x})$. V postopku k-povprečij boste določili ustrezno število središč k glede na pričakovano število velikih struktur v sliki – v slikah možganskih tkiv so to WM, GM in CSF. Značilnice $\boldsymbol{y}_j$ boste razvrstili v k razredov glede na najmanjšo oddaljenost od i-tega središča $\boldsymbol{\mu}_i$, tj. minimalno $d_i(\boldsymbol{y}_j) = \|\boldsymbol{y}_j - \boldsymbol{\mu}_i\|$.Vsaka značilnica $\boldsymbol{y}_j$ dobi pripadajočo oznako razreda z_j:

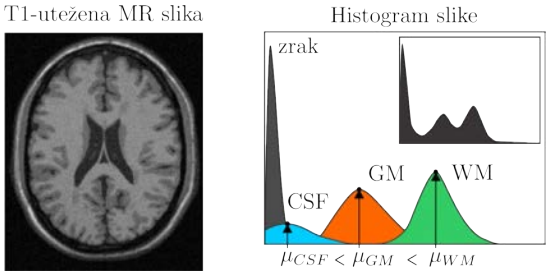
$$z_j = \arg\min_{\forall i} d_i(\boldsymbol{y}_j)\,. \tag{3}$$

Vsaki značilnici $\boldsymbol{y}_j$ v prostoru slike ustreza slikovni element s koordinatami $\boldsymbol{x}_j$, ki ga v sliki razgradnje S zapišemo s pripadajočo oznako $S(\boldsymbol{x}_j) = z_j$. Postopek označevanja slikovnih elementov na osnovi analize gruč s postopkom k-povprečij je prikazan na sliki 2.



Slika 2: Razgradnja z neparametričnim razvrščanjem v prostoru značilnic (*od leve proti desni*): analiza gruč v prostoru značilnic s $\boldsymbol{y}_j$ postopkom k-povprečij in označevanje slikovnih elementov $\boldsymbol{x}_j$ v MR sliki z neparametričnim razvrščanjem značilnic $\boldsymbol{y}_j$ glede na minimalno razdaljo $d_i(\boldsymbol{y}_j)$ do najbližjega središča gruče $\boldsymbol{\mu}_i$.

Ker je začetna lega središč $\boldsymbol{\mu}_1^{(0)}, \boldsymbol{\mu}_2^{(0)}, \ldots, \boldsymbol{\mu}_k^{(0)}$ določena naključno in so zato oznake gruč $Z = \{1, 2, \ldots, k\}$ tudi naključne. Pri razgradnji možganskih struktur bomo uporabili $k = 3$ za tri največje strukture CSF, GM in WM. Po končanju postopka k-povprečij moramo zato določiti konsistentne oznake možganskih struktur CSF, GM in WM (z_{CSF}, z_{GM} in z_{WM}). To lahko naredimo tako, da glede na razgradnjo S izračunamo povprečne sivinske vrednosti μ_1, μ_2 in μ_3 na T1-uteženi MR sliki, ki ima največji kontrast med temi možganskimi strukturami. V splošnem za povprečne sivinske vrednosti struktur CSF, GM in WM velja $\mu_{CSF} < \mu_{GM} < \mu_{WM}$ (slika 3), zato lahko na enak način razvrstimo μ_1, μ_2 in μ_3 in določimo pripadajoče konsistentne oznake z_{CSF}, z_{GM} in z_{WM}.



Slika 3: Prečni prerez T1-uteženе MR slike in histogram slike. Za povprečne sivinske vrednosti struktur CSF, GM in WM katere velja $\mu_{CSF} < \mu_{GM} < \mu_{WM}$, kar lahko izkoristimo za konsistentno označevanje teh struktur.

Vrednotenje razgradnje MR slik

Zmogljivost postopkov razgradnje MR slik običajno ovrednotimo s stopnjo prekrivanja med referenčnimi obrisi možganskih struktur in avtomatsko razgradnjo teh struktur. Kot kvantitativna mera prekrivanja obrisov se najpogosteje uporablja Diceov koeficient podobnosti (DSC[6])

$$DSC_r = \frac{2\,|S_r \cap R_r|}{|S_r| + |R_r|}\,, \quad r = \{CSF, GM, WM\}$$

[6]DSC – ang. Dice Similarity Coefficient

kjer $\cap$ predstavlja presek dveh regij, $|\cdot|$ pa velikost regije. S_r je razgradnja pridobljena z avtomatskim postopkom, R_r pa referenčna razgradnja.

Naloge

Gradivo za vajo vsebuje datoteke `t1.nrrd`, `t2.nrrd`, `pd.nrrd`, ki predstavljajo 3D sivinske slike T1-, T2- in PD-uteženе MR sekvence, datoteka `msk.nrrd` pa podaja pripadajoče referenčne obrise (maske) možganskih struktur. Vse slike in maske imajo dimenzije $217 \times 181 \times 181$, korak vzorčenja je 1 mm vzdolž vseh osi slike. Sivinske vrednosti T1-, T2- in PD-uteženih MR slik so zapisane z nepredznačenimi 8-bitnimi celimi števili. Referenčni obrisi možganskih struktur so v datoteki `msk.nrrd` v nepredznačenem 8-bitnem celoštevilskem zapisu, obrisi posameznih struktur pa imajo standardizirane oznake. Oznake normalnih in patoloških možganskih struktur z_{CSF}, z_{GM} in z_{WM} so dane kot $L = \{CSF = 1, GM = 2, WM = 3, LESIONS = 10\}$.

1. Določite binarno masko `M` področja možganskih struktur iz danih referenčnih obrisov, in sicer kot unijo področij $\Omega_M = \bigcup_{i=1}^{L} \Omega_i$, ki ustrezajo danim oznakam L. Prikažite prečni prerez $x \times y \times 90$ slike maske, tj. prerez pri koordinati $z = 90$.

2. Iz 3D sivinskih slik `t1.nrrd`, `t2.nrrd` in `pd.nrrd` izluščite tiste sivinske vrednosti $I_{T1}(\boldsymbol{x})$, $I_{T2}(\boldsymbol{x})$ in $I_{PD}(\boldsymbol{x})$, ki ležijo na področju maske `M` ($\boldsymbol{x} \in \Omega_M$, $|\Omega_M| = N$). Ustvarite matriko značilnic `Y` kot $Y = [\, I_{T1}(\boldsymbol{x}),\, I_{T2}(\boldsymbol{x}),\, I_{PD}(\boldsymbol{x})\,]$, ki ima dimenzije $N \times 3$. Prikažite značilnice $\boldsymbol{y} \in Y$ kot točke v 3D prostoru značilnic.

3. Napišite funkcijo za določanje začetnih središč $\boldsymbol{\mu}_i$:

   ```
   def kMeansInit( iY, iK ):
           return oMu
   ```

   kjer je vhodni parameter `iY` matrika značilnic dimenzij $N \times d$, `iK` pa število gruč k. Funkcija vrne matriko `oMu`, ki ima dimenzije $d \times k$ in ki predstavlja začetna središča. Določite tri začetna središča $\boldsymbol{\mu}_i^{(0)}$, $i = 1, 2, 3$ na matriki značilnic `Y`, ki ste jo ustvarili pri prejšni nalogi kot $Y = [\, I_{T1}(\boldsymbol{x}),\, I_{T2}(\boldsymbol{x}),\, I_{PD}(\boldsymbol{x})\,]$ in ki ima dimenzije $N \times 3$.

4. Napišite funkcijo za analizo gruč s postopkom k-povprečij++:

   ```
   def kMeansPP( iY, iK, iMaxIter ):
           return oMu
   ```

   kjer je vhodni parameter `iY` matrika značilnic dimenzij $N \times d$, `iK` število gruč k, `iMaxIter` pa maksimalno število korakov postopka k-povprečij++. Uporabite funkcijo `oMu = kMeansInit( iY, iK )`, ki ste jo ustvarili pri prejšnji nalogi, za določanje začetnih središč $\boldsymbol{\mu}_i^{(0)}$, $i = 1, \ldots, k$. Implementirajte operaciji 1) razvrščanja značilnic (enačba 1) in 2) posodabljanja značilnic (enačba 2), ki predstavljata en korak postopka k-povprečij++. Postopek naj se izvaja dokler se položaji središč po koncu koraka spremenijo ($\boldsymbol{\mu}_i^{(t+1)} \neq \boldsymbol{\mu}_i^{(t)}$, $\forall i$) ali do vnaprej določenega maksimalnega števila korakov t_{max}. Funkcija vrne matriko `oMu`, ki ima dimenzije $k \times d$ in ki predstavlja središča gruč, določena s postopkom k-povprečij++.

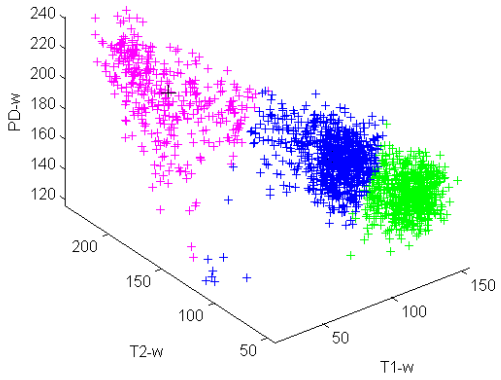
   Določite tri središča gruč $\boldsymbol{\mu}_i$, $i = 1, 2, 3$ na matriki značilnic `Y`, ki ste jo ustvarili pri nalogi 2. kot $Y = [\, I_{T1}(\boldsymbol{x}),\, I_{T2}(\boldsymbol{x}),\, I_{PD}(\boldsymbol{x})\,]$ in ki ima dimenzije $N \times 3$. Izberite primerno število korakov `iMaxIter`.

5. Napišite funkcijo za neparametrično razvrščanje značilnic:

   ```
   def nonparClassification( iY, iMu ):
           return oZ
   ```

   kjer je vhodni parameter `iY` matrika značilnic, ki ima dimenzije $N \times d$, `iMu` pa matrika središč gruč, ki ima dimenzije $k \times d$. Funkcija vrne vektor `oZ` z dimenzijami $N \times 1$, ki predstavlja oznake $z \in \{1, \ldots, k\}$ značilnic v matriki `iY` glede na dana središča gruč `iMu` in ki jih določite po enačbi 3.

   Določite oznake značilnic glede na središča gruč `iMu`, ki ste ji izračunali s postopkom k-povprečij++. Prikažite značilnice $\boldsymbol{y} \in Y$ kot točke v 3D prostoru značilnic in točke obarvajte različno glede na oznako pripadajoče značilnice tako, kot je prikazano na sliki 4.



Slika 4: Neparametrično razvrščanje značilnic $Y = [\, I_{T1}(\boldsymbol{x}),\, I_{T2}(\boldsymbol{x}),\, I_{PD}(\boldsymbol{x}) \,]$ s postopkom k-povprečij++.

6. Napišite funkcijo za razgradnjo MR slik glave z neparametričnim razvrščanjem značilnic s postopkom k-povprečij++:

```
def mrBrainSegmentation( t1, t2, pd, mask ):
        return oS
```

kjer vhodni parametri `t1`, `t2` in `pd` predstavljajo 3D sivinske MR slike, zajete s T1-, T2- in PD-uteženo sekvenco. Vhodni parameter `mask` je binarna 3D slika maske možganskih struktur (naloga 1). Vse vhodne slike imajo enake dimenzije. Uporabite funkciji `kMeansPP()` in `nonparClassification()` na značilnicah Y iz področja maske za analizo gruč in neparametrično razvrščanje za določanje treh možganskih struktur CSF, GM in WM ($k = 3$) na podlagi vseh treh vhodnih slik ($d = 3$).

Funkcija naj vrne 3D sliko oznak `oS`, ki ima enake dimenzije kot vhodne slike. S pomočjo povprečnih T1-uteženih sivinskih vrednosti (slika 3) v posamezni gruči določite standardne oznake L struktur CSF, GM in WM, ki so dane kot $z_{CSF} = 1, z_{GM} = 2$ in $z_{WM} = 3$. Slikovne elemente v `oS`, ki ležijo zunaj maske `mask`, postavite na 0. Prikažite prečni prerez $x \times y \times 90$ izhodne slike oznak `oS` in uporabite barvno kodiranje sivinskih vrednosti (npr. z barvno lestvico `jet`).

Dodatne naloge

Dodatne naloge naj služijo za poglobitev spretnosti programiranja, boljšemu razumevanju snovi in vsebine vaje in spoznavanju dodatnih načinov za obdelavo in analizo medicinskih slik. Opravljanje dodatnih nalog je neobvezno, vendar pa priporočljivo, saj je na nek način to priprava na zagovor laboratorijskih vaj.

1. Primerjajte delovanje funkcije `kMeanPP()` z delovanjem funkcije `ScalarImageKmeansImage()` v knjižnici `SimpleITK`. Ali dobite enak rezultat?

2. Predložite program, ki temelji na algoritmu k-povprečij++ in ki avtomatsko določi prostornino bele in sive možganovine na MR slikah. Prikažite delovanje na danih slikah `t1.nrrd`, `t2.nrrd` in `pd.nrrd`, pri čemer področje možganov izluščite iz maske `msk.nrrd` kot elemente večje on 0. Določite prostornino bele in sive možganovine v enoti mililiter (ml).

3. Napišite funkcijo izračun Diceovega koeficienta med dvema vhodnima slikama oznak `iS` in `iR`:

```
def computeDiceCoeff( iS, iR ):
        return oDSC
```

kjer je `oDSC` skalar. Ovrednotite razgradnjo MR slik glave s postopkom k-povprečij in neparametričnim razvrščanjem tako, da izračunate Diceove koeficiente med dobljenimi oznakami `iS` in referenčnimi oznakami (datoteka `msk.nrrd`) za strukture CSF, GM in WM. Navedite vrednosti DSC_{CSF}, DSC_{GM}, DSC_{WM}.

4. Prilagodite funkcijo `mrBrainSegmentation()` tako, da za razgradnjo MR slik glave v postopku k-povprečij kot značilnice uporabite le sliki `t2` in `pd` ($d = 2$). Razgradite MR sliki in vrednotite dobljeno razgradnjo z Diceovim koeficientom. Ali se kakorkoli spremeni kakovost razgradnje v primerjavi z razgradnjo dobljeno pri prešnji nalogi? Obrazložite odgovor.

5. Prilagodite funkcijo `mrBrainSegmentation()` tako, da za razgradnjo MR slik glave v postopku k-povprečij kot značilnico uporabite le sliko `t1` ($d = 1$). Razgradite MR sliko in vrednotite dobljeno razgradnjo z Diceovim koeficientom. Kakšna je kakovost razgradnje v primerjavi z razgradnjama dobljenima pri prešnjih dveh nalogah? Obrazložite odgovor.

6. Prilagodite funkcijo `mrBrainSegmentation()` tako, da bo poleg razgradnje normalnih možganskih struktur (CSF, GM in WM) omogočala tudi razgradnjo lezij.

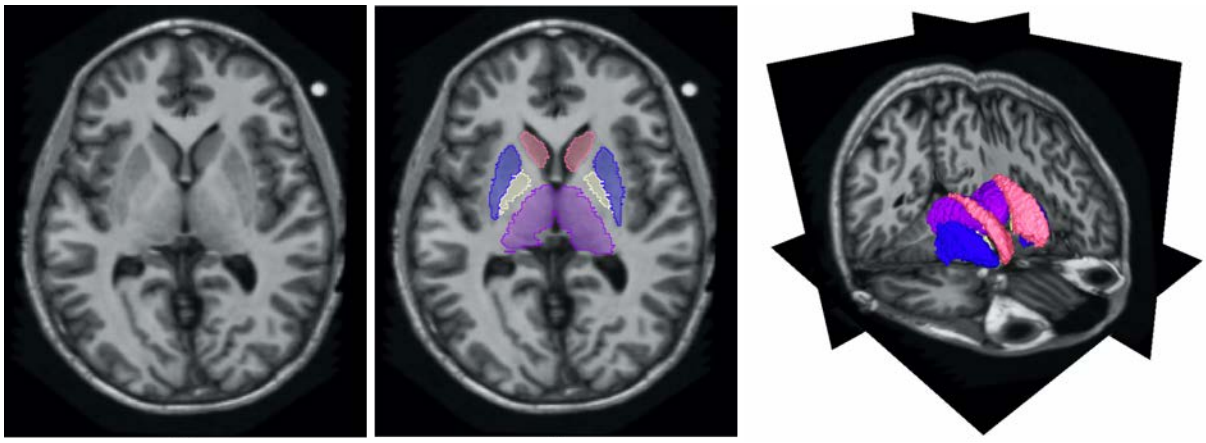
   - Navedite, katere MR slike ste uporabili za razgradnjo in za dobljeno razgradnjo izračunajte Diceove koeficiente za normalne možganske strukture (DSC_{CSF}, DSC_{GM}, DSC_{WM}) in lezije ($DSC_{LESIONS}$). Na kratko komentirajte kakovost razgradnje posameznih struktur.
   - Določite prostornino lezij v enoti mililiter (ml) in število lezij.

Univerza v Ljubljani, Fakulteta za elektrotehniko
Analiza medicinskih slik (AMS)

Razgradnja slik s poravnavo atlasov

Navodila

Avtomatska razgradnja slik temelji na osnovnih lastnostih opazovanih struktur, kot so sivinske vrednosti, tekstura, oblika, itd. Pogosto se pri razgradnji srečamo s strukturami, ki so po sivinski vrednosti ali teksturi povsem enake, lahko imajo tudi enako obliki in se stikajo v prostoru, kar precej otežuje njihovo razgradnjo. Primer tovrstnih struktur v osrednji sivi možganovini je prikazan na sliki 1a. Za razgradnjo teh struktur se lahko zanašamo zanašamo na predznanje o lastnostih struktur zanimanja, kot naprimer anatomska lega in medsebojna povezanost in soodvisnost (lege in/ali oblike) področij zanimanja. Tovrstno predznanje za razgradnjo neke nove slike je lahko podano s podobno sliko, za katero imamo na voljo natančne razgradnje struktur zanimanja, čemur pravimo tudi *topološki atlas* (sliki 1bc). Razgradnjo lahko naredimo s poravnavo topološkega atlasa na novo sliko in preslikovanjem značk[1]. Natančnost tovrstne razgradnje zavisi predvsem od natančnosti poravnave slik. Ker je topološki atlas praviloma pridobljen na drugem subjektu kot nova slika moramo običajno uporabiti netogo poravnavo slik za prileganje atlasa na sliko subjekta. Po drugi strani en sam topološki atlas ne kodira običajno precej visoke biološke variabilnosti med subjekti, zato je smiselno uporabiti več topoloških atlasov za razgradnjo slik. To lahko naredimo s poravnavo vseh topoloških atlasov na novo sliko in zlivanjem značk[2]. Prednost zlivanja značk je tudi v tem, da lahko kompenzira napake zaradi netočne poravnave slik. Razgradnja slik z zlivanjem značk je trenutno ena od najbolj uveljavljenih tehnik razgradnje, razvoj postopkov zlivanja značk pa je zelo aktivno raziskovalno področje, zato bomo tekom vaje podrobneje analizirali nekatere najbolj uveljavljene in učinkovite postopke zlivanja.



(a) (b) (c)

Slika 1: (a) prečna rezina T1-utežene MR slike glave in (b) pripadajoči topološki atlas struktur v osrednji sivi možganovini – kavdatno jedro (*roza*), putamen (*modra*), palidus (*rumena*) in talamus (*vijolična*). (c) Vizualizacija teh struktur v 3D.

Tehnike zlivanja lahko uporabimo tudi na poravnanih slikah in tako zgradimo referenčni oz. *populacijski* atlas – ta lahko predstavlja povprečno anatomsko obliko struktur zanimanja za neko skupino subjektov. S poravnavnimi značkami lahko zgradimo *statistični* atlas, ki predstavlja *a priori* verjetnost posamezne strukture v prostoru. Statistični atlasi se pogosto uporabljajo kot predznanje pri razgradnji z rojenjem, bodisi za inicializacijo postopka rojenja bodisi za vpeljevanje prostorske povezanosti struktur pri razgradnji slik z analizo rojev.

Strategije razgradnje z atlasi

Topološki atlas je dan s sivinsko sliko $\mathcal{I}(\boldsymbol{x})$, $\boldsymbol{x} \in \mathbb{R}^d$, kjer d predstavlja dimenzionalnost slik, in pripadajočo razgradnjo $\mathcal{S} : \boldsymbol{x} \in \mathbb{R}^d \rightarrow \mathcal{S}(\boldsymbol{x}) \in L$, kjer so $L \in \mathbb{Z}^+$ značke struktur v atlasu.

[1] ang. *label propagation*
[2] ang. *label fusion*

Najbolj osnovna strategija razgradnje z atlasom je **preslikovanje značk**, ki jo izvedemo z netogo poravnavo atlasa v prostor nove slike $\mathcal{J}(\boldsymbol{x})$:

$$\mathcal{S}_{\mathcal{J}}(\boldsymbol{x}) = \mathcal{T}^*(\mathcal{S}_{\mathcal{I}}(\boldsymbol{x})), \quad \mathcal{T}^* = \text{argmax}_{\mathcal{T}} MP(\mathcal{T}(\mathcal{I}), \mathcal{J}), \tag{1}$$

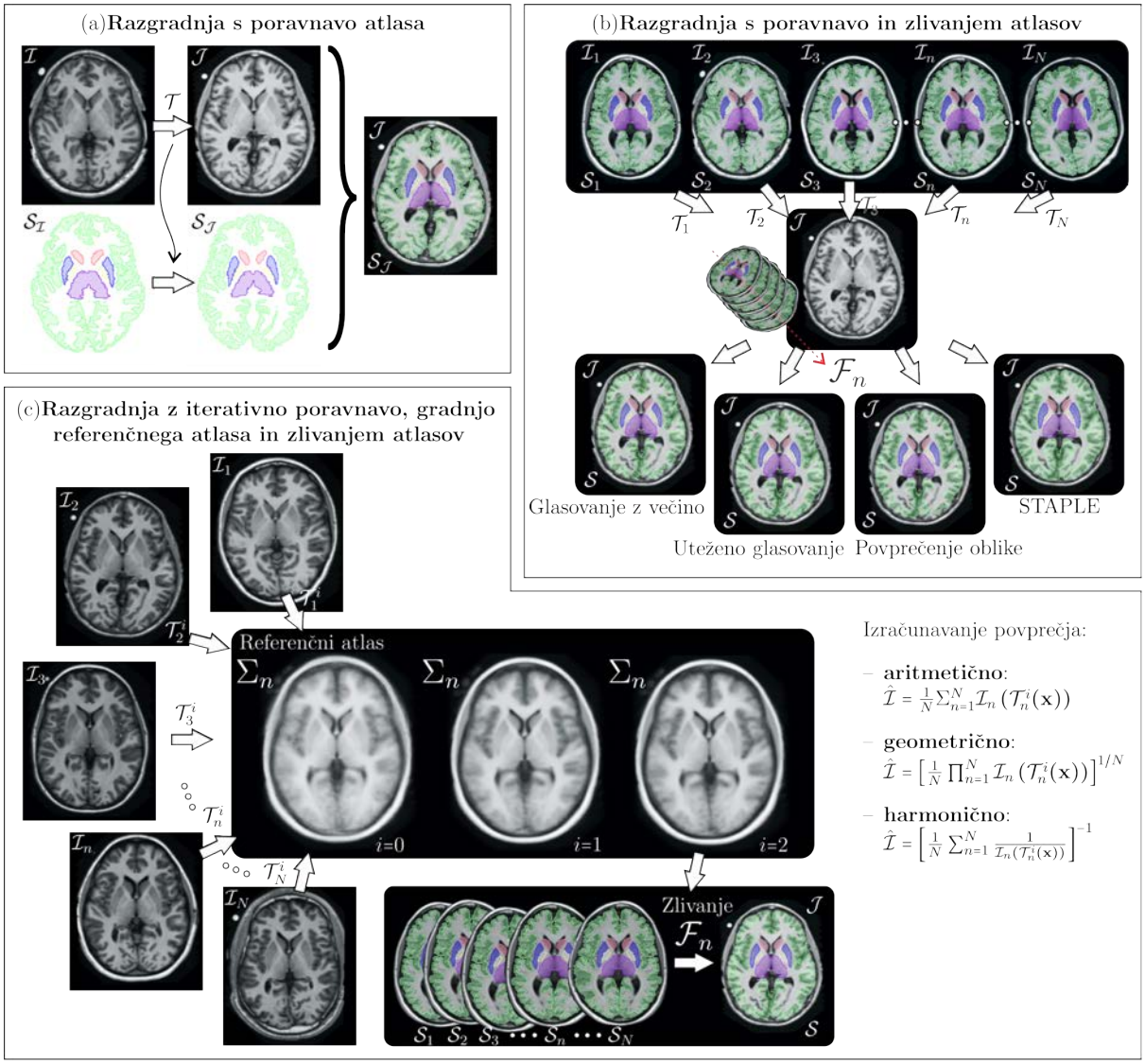
kjer je $\mathcal{T}^*$ optimalna poravnava določena na podlagi maksimizacije mere podobnosti med slikama $\mathcal{I}$ in $\mathcal{J}$. Za netogo poravnavo se najpogosteje uporablja preslikava z B–zlepki z maksimizacijo medsebojne informacije. Primer preslikovanja značk je prikazan na sliki 2a.

Naprednejše strategije uporabljajo več topoloških atlasov $\mathcal{I}_n(\boldsymbol{x}), \mathcal{S}_n(\boldsymbol{x}); n = 1, \ldots, N$ tako, da se vsi atlasi (slike in pripadajoče značke) poravnajo v prostor nove slike $\mathcal{J}(\boldsymbol{x})$, nato pa se značke zlijejo v nove značke, ki predstavljajo razgradnjo. Pogosto se uporabljajo tehnike zlivanja, ki temeljijo na **glasovanju**:

$$\mathcal{S}_{\mathcal{J}}(\boldsymbol{x}) = \text{argmax}_l \sum_{n=1}^{N} w_n(\boldsymbol{x}) \cdot f(\mathcal{T}_n(\mathcal{S}_n(\boldsymbol{x})), l),\ l \in L \tag{2}$$

kjer $\mathcal{T}_n(\cdot)$ predstavlja poravnavo atlas $\mathcal{I}_n$ na sliko $\mathcal{J}$, l predstavlja značko, funkcija $f(\cdot)$ pa je določena kot:

$$f(\mathcal{T}_n(\mathcal{S}_n(\boldsymbol{x})), l) \begin{cases} 1: & \mathcal{T}_n(\mathcal{S}_n(\boldsymbol{x})) = l \\ 0: & \mathcal{T}_n(\mathcal{S}_n(\boldsymbol{x})) \neq l \end{cases} \tag{3}$$



Slika 2: Strategiji razgradnje z atlasi: (a) s poravnavo atlas na novo sliko subjekta; (b) s poravnavo večih atlasov in zlivanjem značk v razgradnjo slike subjekta. (c) Nepristranska poravnava skupine slik izboljoša natančno poravnave v primerjavi s poravnavo sliko po parih in se uporablja za izgradnjo statističnega atlasa.

Pri **glasovanju z večino**[3] so uteži $w_n(\boldsymbol{x})$ določene kot $w_n(\boldsymbol{x}) = 1/N$.

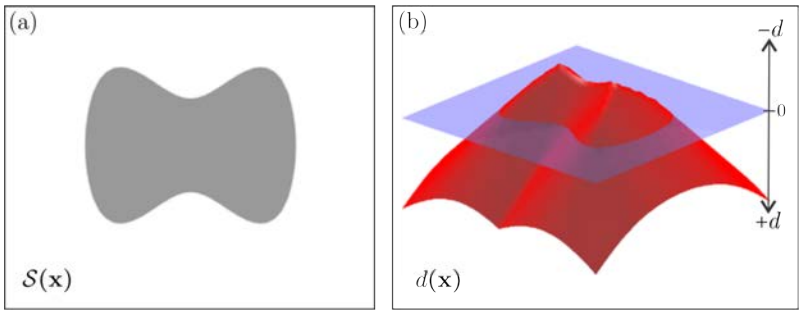
Naprednejša strategija, ki upošteva tudi podobnost struktur, je **glasovanje z lokalnim uteževanjem**[4] vsakega slikovnega elementa z vrednostjo mere podobnosti med lokalno okolico $\mathcal{J}((x)$ in $\mathcal{T}_n(\mathcal{I}_n((x))$. Za mero podobnosti se lahko uporabi križno-korelacijski koeficient (CC):

$$w_n(\boldsymbol{x}) = CC(\mathbf{x}|\mathcal{I}_n, \mathcal{J}) = \frac{\sum_{\mathbf{x}\in\mathcal{R}}(\mathcal{I}_n(\mathbf{x}) - \hat{\mathcal{I}}_n^{\mathcal{R}}) \cdot (\mathcal{J}(\mathbf{x}) - \hat{\mathcal{J}}^{\mathcal{R}})}{\sum_{\mathbf{x}\in\mathcal{R}}(\mathcal{I}_n(\mathbf{x}) - \hat{\mathcal{I}}_n^{\mathcal{R}})^2 \cdot \sum_{\mathbf{x}\in\mathcal{R}}(\mathcal{J}(\mathbf{x}) - \hat{\mathcal{J}}^{\mathcal{R}})^2}\,, \tag{4}$$

kjer je $\mathcal{R}$ kvadratna okolica koordinate $\boldsymbol{x}$, v kateri vrednotimo podobnost, $\hat{\mathcal{I}}_n^{\mathcal{R}}$ in $\hat{\mathcal{J}}^{\mathcal{R}}$ pa povprečni sivinski vrednosti v $\mathcal{R}$.

Skupna slabost strategij z glasovanjem je v tem, da eksplicitno ne upoštevajo povezanost struktur v prostoru. Naprimer, dveh sosednjih elementih $\boldsymbol{x}_i$ in $\boldsymbol{x}_{i+1}$ lahko predvsem v primeru malega števila atlasov N kaj hitro dobimo različne značke l, kar lahko povzroči nazobčan rob oz. prehod med različnimi strukturami. Ta problem naslavlja postopek, ki temelji na **povprečenju oblike**[5]:

1. Za vsak topološki atlas $\mathcal{S}_n(\boldsymbol{x})$ in vsako strukturo z značko l $(l \in L)$ izračunaj polje Evklidskih razdalj $d_{n,l}(\boldsymbol{x})$ do roba strukture $\mathcal{S}_n(\boldsymbol{x}) = l$. Primer polja prikazuje slika 3.



Slika 3: (a) Obris strukture in pripadajoča (b) Evklidska mapa razdalj od roba strukture. Razdalje znotraj strukture imajo negativen, zunaj pa pozitiven predznak.

2. Določi povprečno razdaljo vsake koordinate $\boldsymbol{x}$ do značke l glede na vse topološke atlase:

$$D_l(\boldsymbol{x}) = \frac{1}{N}\sum_{n=1}^{N} d_{n,l}(\boldsymbol{x})\,, \tag{5}$$

3. Končno razgradnjo v vsaki koordinati $\boldsymbol{x}$ dobimo z minimizacijo povprečne razdalje preko vseh značk:

$$\mathcal{S}(\boldsymbol{x}) = \text{argmin}_l D_l(\boldsymbol{x}),\ l \in L\,. \tag{6}$$

Skupinska poravnava slik

Razgradnja s preslikovanjem in zlivanjem značk ter gradnja populacijskih in statističnih atlasov zahteva zelo natančno poravnavo množice slik različnih subjektov. Poravnava slik običajno teče med parom slik, zato glede na izbiro zaporedja parov slik za poravnavo in glede na izbiro referenčne slike lahko dobimo različne rezultate. Zato je zelo aktivno raziskovalno področje tudi razvoj postopkov za *nepristransko* skupinsko poravnavo slik[6]. Enostaven postopek za nepristransko skupinsko poravnavo slik, ki temelji na poravnavo slik po parih:

1. Z linearno preslikavo $\mathcal{T} = \mathcal{T}_{affine}$ poravnaj slike $\mathcal{I}_n$; $n = 1, \ldots, N$ na naključno izbrano referenčno sliko.
2. S povprečenjem poravnanih slik $\mathcal{I}_n$ določi novo referenčno sliko.
3. Z netogo preslikavo $\mathcal{T} = \mathcal{T}_{nonrigid}$ poravnaj slike $\mathcal{I}_n$; $n = 1, \ldots, N$ na referenčno sliko.
4. Ponovi koraka 2 in 3 dokler se referenčna slika spreminja oz. do maksimalnega števila iteracij.

Slika 2c prikazuje potek postopka poravnave in podaja različne načine določanja referenčne slike oz. atlasa, in sicer z aritmetičnim, geometričnim ali harmoničnim povprečenjem.

[3] ang. *majority voting*
[4] ang. *locally weighted voting*
[5] ang. *shape-based averaging*
[6] ang. *unbiased groupwise registration*

Naloge

Gradivo za vajo vsebuje datoteko `t1-images.nii.gz` s 10 prečnimi 2D rezinami T1-uteženih MR sekvenc glave, ki so bile predhodno z afino preslikavo poravnane v referenčni koordinatni sistem. V datoteki `gm-masks.nii.gz` je prav tako v referenčnem koordinatnem sistemu danih 10 pripadajočih mask z značkami podstruktur sive možganovine, ki so standardizirane kot $l = \{CAUDATE = 1, PALLIDUM = 2, PUTAMEN = 3, THALAMUS = 4, CORTEX = 5\}$. Maske podstruktur so bile pridobljene z natančnim ročnim obrisovanjem in predstavljajo referenčne maske, ki jih lahko uporabite za vrednotenje razgradnje.

1. Pripravite naslednji funkciji za netogo poravnavo slik z B–zlepki in preslikovanje poravnane slike tako, da preuredite Python skripto iz Naloge 5 pri Vaji 2 (Netoga poravnava slik), ki temelji na uporabi knjižnice `SimpleITK`. Prva funkcija, ki izvede poravnavo z B–zlepki naj ima podpis:

   ```
   def bsplineRegistration( iFixed, iMoving, iBSplineGridSpacing, iMaxIter ):
           return oTx
   ```

   pri čemer `iFixed` in `iMoving` predstavljata referenčno in premično vhodno 2D sliko za poravnavo. Vhodni sliki naj bosta tipa `numpy.ndarray`, za uporabo v funkciji pa jih pretvorite v tip `itk.Image`. Parameter `iBSplineGridSpacing` določa dolžino koraka v diskretni mreži kontrolnih točk za izračun B–zlepkov, parameter `iMaxIter` pa maksimalno število iteracij. Funkcija vrne izračunano preslikavo v spremenljivki `oTx`.

   Druga funkcija, ki na podlagi izračunane preslikave izvede vzorčenje premične slike, naj ima podpis:

   ```
   def bsplineResample( iFixed, iMoving, iTx, iInterpType ):
           return oImage
   ```

   kjer je `iFixed` referenčna slika, `iMoving` pa slika oz. maska za vzorčenje. Vhodni sliki naj bosta tipa `numpy.ndarray`, za uporabo v funkciji pa jih pretvorite v tip `itk.Image`. Vhodni parameter `iTx` prestavlja preslikavo, parameter `iInterpType` pa način interpolacije slik. Za sivinske slike lahko uporabite interpolacijo prvega reda (`itk.sitkLinear`), za maske pa interpolacijo ničtega reda (`itk.sitkNearestNeighbor`). Funkcija naj vrne preslikano sliko v spremenljivki `oImage`, ki naj bo tipa `numpy.ndarray`.

   Preizkusite delovanje funkcij tako, da poravnate prvi dve prečni rezini v sliki `t1-images.nii.gz`, pri čemer naj prva rezina predstavlja referenčno, druga pa premično 2D sliko. Ustrezno nastavite parametra poravnave `iBSplineGridSpacing` in `iMaxIter` tako, da si prikažete poravnano in neporavnano rezino ter njuno razliko in kvalitativno ocenite ali je bila poravnava uspešna.

2. Napišite funkcijo za razgradnjo na osnovi poravnave in preslikovanja značk:

   ```
   def labelPropagation( iFixed, iMoving, iMovingLabelMap ):
           return oFixedLabelMap
   ```

   pri čemer `iFixed` in `iMoving` predstavljata referenčno in premično vhodno 2D sliko. Premični sliki pripada tudi vhodna maska značk `iMovingLabelMap`, ki jih želimo z netogo poravnavo preslikati na referenčno sliko. Funkcija naj v spremenljivki `oFixedLabelMap` vrne v prostor referenčne slike preslikane značke `iMovingLabelMap`.

   Naj prva prečna rezina v sliki `t1-images.nii.gz` predstavlja referenčno sliko, ki jo želimo razgraditi. Preizkusite delovanje funkcije tako, da maske na vseh ostalih rezinah preslikate v to referenčno rezino. Posamezno premično rezino in pripadajočo rezino z masko v sliki `gm-masks.nii.gz` preslikajte v prostor referenčne slike in prikažite.

   Vrednotite uspešnost razgradnje z Diceovim koeficientom (Vaja 6, funkcija `computeDiceCoeff()`) tako, da maske preslikanih značk primerjate z masko ročno določenih značk na referenčni rezini. Izračunajte povprečno, minimalno in maksimalno vrednost Diceovega koeficienta za vsako od petih GM struktur ($l = \{1, 2, 3, 4, 5\}$) preko vseh preslikanih mask značk.

3. Napišite funkcijo za razgradnjo na osnovi poravnave in zlivanja večih atlasov s pomočjo glasovanja z večino:

   ```
   def fusionMajorityVoting( iMovingLabelMaps ):
           return oFixedLabelMap
   ```

kjer `iMovingLabelMaps` predstavlja seznam (`list`) na referenčno sliko poravnavnih premičnih vhodnih značk, ki jih želimo zliti v novo masko značk. Funkcija naj v spremenljivki `oFixedLabelMap` vrne v prostoru referenčne slike zlito nove masko značk, ki predstavljajo razgradnjo te slike.

Naj prva prečna rezina v sliki `t1-images.nii.gz` predstavlja referenčno sliko, ki jo želimo razgraditi, ostale slike pa naj bodo premične slike in maske, s katerimi bomo naredili razgradnjo s poravnavo in zlivanjem značk na osnovi glasovanja z večino. Maske značk so dane v `gm-masks.nii.gz`. Vrednotite uspešnost razgradnje z Diceovim koeficientom tako, da z zlivanjem pridobljeno masko značk primerjate z masko ročno določenih značk na referenčni rezini.

4. Napišite funkcijo za razgradnjo na osnovi poravnave in zlivanja večih atlasov s pomočjo povprečenja oblik:

```
def fusionShapeBasedAveraging( iMovingLabelMaps ):
        return oFixedLabelMap
```

kjer `iMovingLabelMaps` predstavlja seznam (`list`) na referenčno sliko poravnavnih premičnih vhodnih značk, ki jih želimo zliti v novo masko značk. Funkcija naj v spremenljivki `oFixedLabelMap` vrne v prostoru referenčne slike zlito nove masko značk, ki predstavljajo razgradnjo te slike.

Za izračun polja Evklidskih razdalj za dano masko lahko uporabite funkcijo `SignedMaurerDistanceMap(iMask, squaredDistance=False)` v knjižnici `SimpleITK`.

Naj prva prečna rezina v sliki `t1-images.nii.gz` predstavlja referenčno sliko, ki jo želimo razgraditi, ostale slike pa naj bodo premične slike in maske, s katerimi bomo naredili razgradnjo s poravnavo in zlivanjem značk na osnovi glasovanja z večino. Maske značk so dane v `gm-masks.nii.gz`. Vrednotite uspešnost razgradnje z Diceovim koeficientom tako, da z zlivanjem pridobljeno masko značk primerjate z masko ročno določenih značk na referenčni rezini.

Dodatne naloge

Dodatne naloge naj služijo za poglobitev spretnosti programiranja, boljšemu razumevanju snovi in vsebine vaje in spoznavanju dodatnih načinov za obdelavo in analizo medicinskih slik. Opravljanje dodatnih nalog je neobvezno, vendar pa priporočljivo, saj je na nek način to priprava na zagovor laboratorijskih vaj.

1. Napišite funkcijo za izgradnjo referenčnega atlasa z aritmetičnim povprečenjem in nepristransko skupinsko poravnavo slik:

```
def atlasConstruction( iImages, iLabelMaps, iMaxIter ):
        return oAtlasImage, oImages, oLabelMaps
```

kjer `iImages` predstavlja seznam (`list`) 2D slik, `iLabelMaps` pa seznam (`list`) mask značk, ki pripadajo vhodnim 2D slikam. Funkcija iterativno z aritmetičnim povprečenjem poravnavnih slik zgradi referenčni atlas, na katerega se nato z netogo poravnavo poravnajo vse 2D slike, nato se zgradi nov referenčni atlas, itd. Postopek naj se iterativno ponovi do `iMaxIter` števila iteracij. Funkcija naj po zadnji iteraciji vrne referenčni atlas v spremenljivki `oAtlasImage` in pa seznam poravnanih 2D slik in pripadajočih značk v spremenljivkah `oImages` in `oLabelMaps`.

Uporabite vse prečne rezine v sliki `t1-images.nii.gz` in pripadajoče maske značk `gm-masks.nii.gz` za izgradnjo referenčnega atlasa. Prikažite referenčni atlas in preverite, da so prečne rezine in maske značk v spremenljivkah `oImages` in `oLabelMaps` medsebojno poravnane.

Pridobljene poravnane slike in maske maske značk uporabite za razgradnjo prve rezine podobno kot pri nalogah iz vaj tako, da pri zlivanju izpustite prvo rezino. Preverite delovanje postopka `fusionMajorityVoting()` in `fusionShapeBasedAveraging()`. Vrednotite uspešnost razgradnje z Diceovim koeficientom tako, da z zlivanjem pridobljeno masko značk primerjate z masko ročno določenih značk na prvi rezini.

2. Knjižnica `SimpleITK` vključuje implementacijo postopka STAPLE[7] za statistično zlivanje značk:

```
itk.STAPLE(labelVector, confidenceWeight=1.0, foregroundValue=1.0,
                        maximumIterations=5)
```

[7] STAPLE: simultaneous truth and performance level estimation

Funkcija omogoča zlivanje binarnih mask, torej le mask, ki vsebujejo le značko ospredja in ozadja. Izhod funkcije je polje verjetnosti, iz katere lahko z upragovljanjem vrednosti nad 0.5 pridobimo masko značke ospredja. Zlivanja mask z večimi značkami ta implementacija ne omogoča, vendar jo lahko izvedemo tako, da vsako značko ločeno zlijemo in nato združimo v eno masko značk.

- Preizkusite delovanje funkcije tako, da zlijete poravnane maske značk področja $l = 1$ (kvadratno jedro), ki ste jih izračunali pri prejšnji nalogi. Preverite vpliv števila iteracij na rezultat postopka in določite po vašem optimalno število iteracij. Pri tem si lahko pomagate z izračunom Diceovega koeficienta.
- Napišite funkcijo za zlivanje mask z večimi značkami na osnovi dane implementacije postopka STAPLE:

```
def fusionMultilabelSTAPLE( iLabelMaps ):
        return oLabelMap
```

kjer `iLabelMaps` predstavlja seznam (`list`) mask značk, funkcija pa vrne masko zlitih značk `oLabelMap`. Pri prejšnji nalogi pridobljene poravnane rezine v `t1-images.nii.gz` in maske maske značk `gm-masks.nii.gz` uporabite za razgradnjo prve rezine tako, da pri zlivanju izpustite prvo rezino. Vrednotite uspešnost razgradnje z Diceovim koeficientom tako, da z zlivanjem pridobljeno masko značk primerjate z masko ročno določenih značk na prvi rezini. Kateri od postopkov zlivanja vrne najboljše vrednosti Diceovih koeficientov?

3. Napišite funkcijo za razgradnjo na osnovi poravnave in zlivanja večih atlasov s pomočjo lokalno uteženega glasovanja:

```
def fusionLocallyWeightedVoting( iFixed, iMovingImages, iMovingLabelMaps ):
        return oFixedLabelMap
```

kjer `iFixed` predstavlja referenčno 2D sliko, in `iMovingImages` pa seznam (`list`) premičnih vhodnih 2D slik. Premičnim slikam pripadajo tudi vhodne maske značk `iMovingLabelMaps`, ki jih želimo zliti v novo masko značk. Funkcija naj v spremenljivki `oFixedLabelMap` vrne v prostoru referenčne slike zlite nove značke, ki predstavljajo razgradnjo slike `iFixed`.

Za učinkovito implementacijo križno-korelacijskega koeficienta lahko razvijete števec in imenovalec funkcije v enačbi (4) ter posamezne člene izračunate s pomočjo diskretne 2D konvolucije s kvadratnim *box* jedrom (npr. `np.ones([11, 11])`). Za diskretno konvolucijo lahko uporabite funkcijo `convolve()` v Python knjižnici `scipy.ndimage`.

- Pri prvi dodatni nalogi ste poravnali rezine v `t1-images.nii.gz` in maske maske značk `gm-masks.nii.gz`. Uporabite jih za razgradnjo prve rezine tako, da pri zlivanju izpustite prvo rezino. Vrednotite uspešnost razgradnje z Diceovim koeficientom tako, da z zlivanjem pridobljeno masko značk primerjate z masko ročno določenih značk na prvi rezini. Kateri od postopkov zlivanja vrne najboljše vrednosti Diceovih koeficientov?
- Določite optimalno velikost kvadratne okolice za izračun križno-korelacijskega koeficienta tako, da dobljena razgradnja dosežete maksimalno vrednost Diceovega koeficienta.

4. Razširite funkcijo za izgradnjo referenčnega atlasa z dodatnim vhodnim parametrom `iMeanType`, tj. `atlasConstruction( ..., iMeanType )`, kjer so možne vrednosti tega parametra 'arithmetic', 'geometric' ali 'harmonic', glede na to s kakšnim načinom povprečenja želimo zgraditi referenčni atlas.

- Izgradite referenčni atlas z vsakim od treh načinov povprečenja in primerjajte dobljene slike referenčnih atlasov. Uporabite vse prečne rezine v sliki `t1-images.nii.gz` in pripadajoče maske značk `gm-masks.nii.gz` za izgradnjo referenčnega atlasa. Izvedite tri iteracije za izgradnjo atlasa (`iMaxIter=3`) in shranite povprečne slike v vsaki iteraciji ter slike prikažite.
- Preverite kako izbira načina povprečenja vpliva na razgradnjo slik s poravnavo in zlivanjem značk tako, da podobno kot pri prvi dodatni nalogi izvedete zlivanje ter določite Diceove koeficiente za vsako od petih GM struktur ($l = \{1, 2, 3, 4, 5\}$). Kateri način povprečenja in kateri način zlivanja slik da najboljši rezultat?

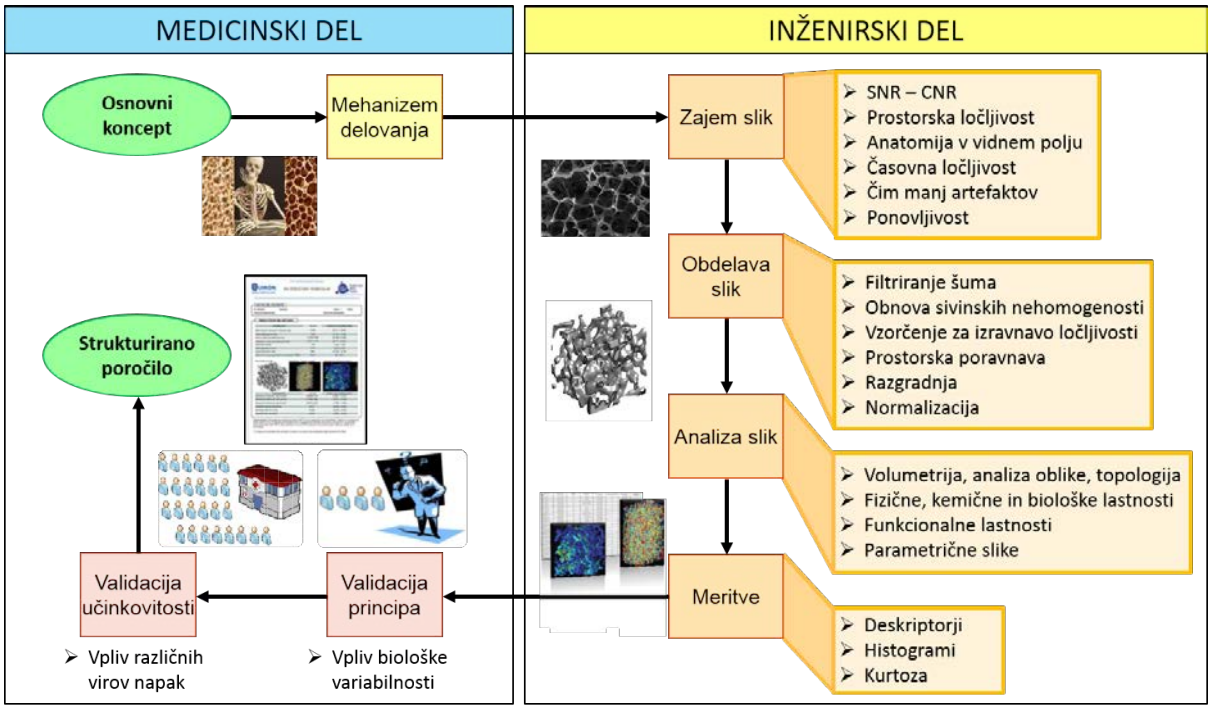
5. Na osnovi optimalno poravnanih slik iz prejšnje domače naloge izgradite statističen atlas za vsako od petih GM struktur ($l = \{1, 2, 3, 4, 5\}$) tako, da uporabite vse maske značk `oLabelMaps`. Statističen atlas bo podan s petimi slikami z vrednostmi od 0 do 1, od katerih vsaka predstavlja prostorsko verjetnost posamezne GM strukture.

Univerza v Ljubljani, Fakulteta za elektrotehniko
Analiza medicinskih slik (AMS)

Analiza slikovnih biomarkerjev

Navodila

Medicinske slikovne tehnike z digitalizacijo postajajo vedno bolj kritično orodje za zgodnjo diagnozo, spremljanje bolezni in odziva na zdravljenje. Omogočajo boljše razumevanje bioloških osnov bolezni, ki skupaj z novimi tehnikami analize teh slik spodbuja tudi uporabo novih parametrov bolezni. Takoimenovani *slikovni biomarker* kodira neko lastnost opazovane anatomije, ki jo lahko objektivno izmerimo na podlagi slik, njegova vrednost pa odraža biološko, funkcionalno ali strukturno organizacijo te anatomije. Ločimo i) napovedni biomarker (ang. *prognosti biomarker*), ki lahko napove potek bolezni in je neodvisen od terapije, ii) pokazalnik zdravljenja (ang. *treatment effect modifier*), ki lahko napove uspešnost terapije, in iii) nadomestni biomarker (ang. *surrogate biomarker*), ki napove potek bolezni glede na izbrano terapijo. Razvoj biomarkerjev zahteva multidisciplinarno sodelovanje, saj so potrebna tako znanja biologije, medicine in klinične prakse, tehnična in metodološka znanja, znanja statistike ter končne inovacije in integracije v kliničnem okolju. Proces razvoja biomarkerjev je prikazan na sliki 1.

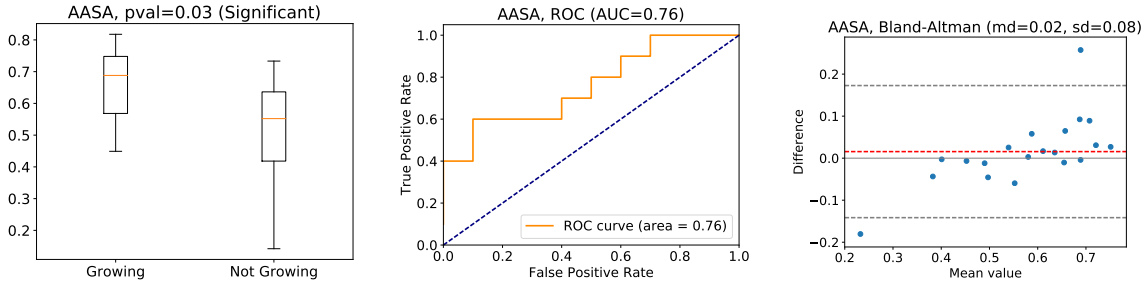


Slika 1: Proces razvoja biomarkerjev.

Pomemben del razvoja slikovnih biomarkerjev predstavlja njihovo vrednotenje, in sicer vrednotenje natančnosti, točnosti, ponovljivost, reprodukcije, itd. V ta namen potrebujemo tudi natančno in zanesljivo referenčno informacijo ali zlati standard, ki je pridobljen na relevantni zbirki slik. Slike za vrednotenje naj bodo zajete tako, da čim bolje odražajo situacijo v klinični praksi. Pri vrednotenju biomarkerjev si pomagamo z orodji vizualizacije porazdelitev vrednosti biomarkerjev s škatelnimi diagrami (slika 2a), ROC[1] krivuljami (slika 2b), Bland–Altman diagrami (slika 2c), s statističnimi testi signifikance (t–test, parni t–test, Wilcoxon rank-sum in signed-rank test, itd.) in merami sposobnosti, kot naprimer AUC[2]. S temi orodji dokazujemo dejansko delovanje in učinkovitost biomarkerjev, ki so odgovor na pomembno klinično vprašanje.

Pri vaji bomo obravnavali slikovne biomarkerje intrakranialnih anevrizem in njihovo vrednost za napovedovanje in spremljanje rasti anevrizme. Intrakranialne anevrizme, ki se pojavljajo v 3,2% svetovne

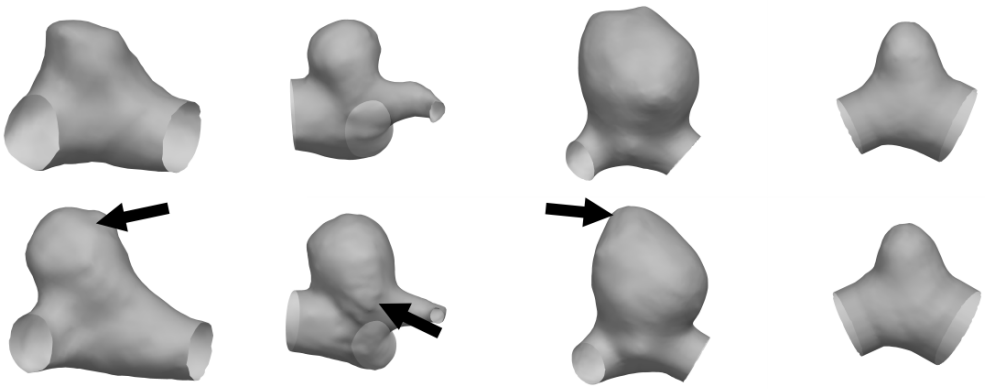
[1] ROC: ang. *receiver operating characteristic*
[2] AUC: ang. *area under ROC curve*



(a) (b) (c)

Slika 2: Primer (a) škatelnega diagrama, (b) ROC krivulje in (c) Bland-Altman diagrama.

populacije (1 na 30 ljudi) se razvijejo po 40. letu starosti. Med bolniki z diagnosticirano anevrizmo je med 10% in 15% takih, ki imajo dve anevrizmi. Večina anevrizem je majhnih, z diametrom od 3 do 20 mm, med 50% do 80% anevrizem pa nikoli ne rupturira. Večje anevrizme (višina kupole >7 mm) se splača zdraviti s kirurškim posegom, in to čim prej, medtem ko za male anevrizme (višina kupole >5 mm) velja, da je tveganje rupture bistveno nižje kot tveganje medoperativnih zapletov. Pri manjših anevrizmah je tveganja rupture majhno, vendar pa je to tveganje bistveno povečano v primeru, da anevrizma raste (slika 3). Zato sta tako zgodnje odkrivanje kot sledenje razvoju majhnih anevrizem zelo pomembna za določanje optimalnega zdravljenja. Na podlagi morfološke analize anevrizem iz CTA (tudi 3D-DSA in MRA) slik lahko razvijemo slikovne prognostične in nadomestne biomarkerje ter pripadajoče klinične smernice za zaznavo in ukrepanje ob potencialno usodnem razraščanju anevrizme.



Slika 3: Površina anevrizme izluščena iz osnovne (*zgoraj*) in sledeče (*spodaj*) CTA preiskave. Puščice označujejo področje rasti, ki predstavlja visoko tveganje za pogosto usodno rupturno anevrizme.

Naloge

Gradivo za vajo vsebuje datoteko `data.p` s kvantitativnimi meritvami možganskih anevrizem za 20 bolnikov. Vsak bolnik ima dve CTA preiskavi, iz vsake pa je bilo z ročnim in avtomatskim postopkom izluščenih šest morfoloških meritev anevrizme (AD, H_{max}, AR, V, $AVSV$, $AASA$). Nevroradiolog je na podlagi analize CTA slik izdelal zlati standard tako, da je za vsako anevrizmo opredelil stanje *ne raste*/*raste* z ustrezno binarno spremenljivko 0/1.

Datoteko `data.p` lahko enostavno naložite v Python z uporabo knjižnice `pickle` z ukazom `load()`, pri čemer boste dobili spremenljivko tipa `dict`, ki je vsebuje na prvem nivoju tri ključe `'biomarkers'`, `'manual'` in `'auto'`, ki podajajo vrstni red in oznake meritev, ročne in avtomatske meritve. Meritve so podane za vsakega bolnika v obliki spremenljike tipa `dict`, ki vsebuje štiri ključe `'bvals'`, `'fvals'`, `'grow'` in `'dyears'`, kjer prva dva ključa vsebujeta meritve prve in sledeče preiskave, tretji binarno vrednost ali anevrizma raste in četrti razliko med preiskavama v letih. Struktura datoteke prikazana v obliki drevesne strukture:

```
datoteka 'data.p'
└─ objekt dict v datoteki s tremi besednimi ključi
```

```
├── 'biomarkers':  imena biomarkerjev z besedo (list)
├── 'manual':  meritve pridobljene z ročno izolacijo anevrizme (dict)
│   ├── 'XJ3252':  identifikacija bolnika (dict)
│   │   ├── 'bvals':  vrednosti biomarkerjev pri prvi preiskavi (list)
│   │   ├── 'fvals':  vrednosti pri naslednji preiskavi (list)
│   │   ├── 'grow':  binarna oznaka raste/ne raste (bool)
│   │   └── 'dyears':  razlika v letih (float)
│   ├── 'WP446':  ...
│   └── 'WC610':  ...
└── 'auto':  meritve pridobljene z avtomatsko izolacijo anevrizme (dict)
    ├── 'XJ3252':  identifikacija bolnika (dict)
    ├── 'WP446':  ...
    └── 'WC610':  ...
```

1. Pripravite funkcijo za nalaganje vrednosti meritev izbranega biomarkerja s podpisom:

```
def loadBiomarkerData( iData, iBiomarker, iWhichValues, iWhichMethod ):
        return oPos, oNeg
```

   pri čemer `iData` predstavlja podatke v spremenljivki tipa `dict`, parametri `iBiomarker`, `iWhichValues`, `iWhichMethod` pa določajo oznako biomarkerja, katerega vrednosti želimo, izbiro vrednosti za prvo ali sledečo preiskavo (`'bvals'`, `'fvals'`) in izbiro postopka (`'manual'`, `'auto'`). Funkcija naj v izhodnih spremeljivkah `oPos` in `oNeg` v obliki seznama `list` vrne vrednosti za anevrizme, ki rastejo in tiste ki ne (pozitivni/negativni vzorci).

   Preverite delovanje funkcije s podatki v dani datoteki `data.p`.

2. Napišite funkcijo za prikaz para škatelnih diagramov ločeno za pozitivne in negativne vzorce in izračun statistične signifikance s t–testom:

```
def analysisBoxplots( iBiomarker, iPos, iNeg, iAxes=None ):
```

   kjer parameter `iBiomarker` podaja oznako biomarkerja, spremeljivki `iPos` in `iNeg` pa vrednosti pozitivnih in negativnih vzorcev v obliki seznama `list`. Parameter `iAxes` naj ima privzeto vrednost `None` oz. naj podaja indeks osi za risanje škatelnih diagramov. Slednje narišete s klicem funkcije `boxplot()` v Python knjižnici `matplotlib.pyplot`. Statistično signifikanco oz. p–vrednost t–testa lahko izračunate s funkcijo `ttest_ind` v Python knjižnici `scipy.stats`.

   Preizkusite delovanje funkcije s podatki v dani datoteki `data.p` tako, da izrišete škatelne diagrame in izračunate p–vrednost za prvotne vrednosti biomarkerja AR in $AASA$, določene z avtomatskim postopkom.

   Preverite tudi statistično signifikanco oz. p–vrednost tudi z neparametričnim Mann-Whitney U-testom, ki jo dobite z uporabo funkcije `mannwhitneyu` v Python knjižnici `scipy.stats`.

3. Napišite funkcijo za prikaz krivulje ROC in izračun površine pod krivuljo (AUC):

```
def analysisROC( iBiomarker, iPos, iNeg, iAxes=None ):
```

   kjer parameter `iBiomarker` podaja oznako biomarkerja, spremeljivki `iPos` in `iNeg` pa vrednosti pozitivnih in negativnih vzorcev v obliki seznama `list`. Parameter `iAxes` naj ima privzeto vrednost `None` oz. naj podaja indeks osi za risanje ROC krivulje. Slednjo izračunate s klicem funkcije `roc_curve()` v Python knjižnici `sklearn.metrics`, AUC mero sposobnosti pa z funkcijo `auc()` v isti knjižnici.

   Preizkusite delovanje funkcije s podatki v dani datoteki `data.p` tako, da izrišete ROC krivuljo in izračunate AUC vrednost za prvotne vrednosti biomarkerja AR in $AASA$, določene z avtomatskim postopkom.

4. Napišite funkcijo za prikaz Bland–Altmanovega diagrama (ang. *bias–variance plot*) med pripadajočimi meritvami z dvema različnima postopkoma:

```
def analysisBlandAltman( iBiomarker, iData1, iData2, iAxes=None ):
```

kjer parameter `iBiomarker` podaja oznako biomarkerja, spremeljivki `iData1` in `iData2` v obliki seznama `list` pa vrednosti vzorcev, pridobljene s postopkom '1' in '2'. Seznama morata imeti enako dolžino, saj podajata pripadajoče meritve. Parameter `iAxes` naj ima privzeto vrednost `None` oz. naj podaja indeks osi za risanje Bland–Altman diagrama. Slednjega izrišete v obliki razsevnega diagrama s funkcijo `scatter()` iz Python knjižnice `matplotlib.pyplot`, pri čemer na horizontalno os nanesete povprečno vrednost pripadajočih meritev, na vertikalno pa razliko. V diagram vrišite horizontalne črte pri vrednosti razlike 0 in pri $+1,96 \cdot SD$ in $-1,96 \cdot SD$, kjer SD predstavlja oceno standardne deviacije razlik.

Preizkusite delovanje funkcije s podatki v dani datoteki `data.p` tako, da izrišete Bland–Altmanov diagram za prvotne vrednosti biomarkerja AR in $AASA$, določene z ročnim in avtomatskim postopkom.

Dodatne naloge

Z uporabo orodij za vrednotenje biomarkerjev odgovorite na naslednji vprašanji:

1. Izmed vseh šestih morfoloških meritev anevrizem določite tisto, ki je najbolj primerna kot prognostični biomarker rasti anevrizme. Ovrednotite tudi sposobnost reprodukcije biomarkerja.

2. Na podlagi analize sprememb posameznih morfoloških meritev anevrizem določite biomarker, ki je najbolj primeren kot nadomestni biomarker rasti anevrizme. Preverite ali je smiselno upoštevati razliko v času med preiskavama. Ovrednotite tudi sposobnost reprodukcije biomarkerja.