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

Uvod v Python in SimpleITK

Pripravil: Žiga Špiclin

Python in digitalne slike

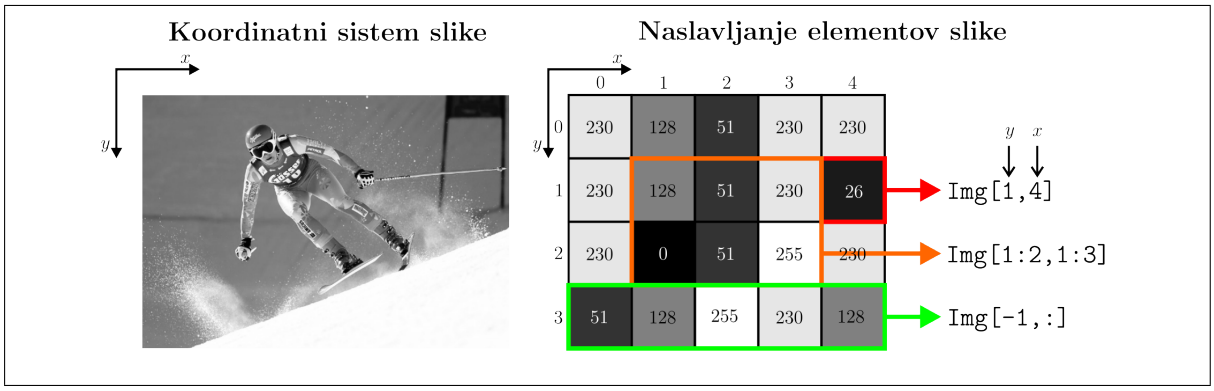
Python je odprtokodni, enostaven in zelo priljubljen interpreterski jezik. Na vajah bomo primere pripravili v programskem okolju WinPython (`http://winpython.github.io/`) z verzijo Pythona 3.x.x, ki ga je mogoče namestiti brez administratorskih pravic, lahko tudi na USB ključek. Vaja služi spoznavanju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

[1] ITK: *Insight Segmentation and Registration Toolkit*, www.itk.org

[2] ang. *edge-preserving smoothing*

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

nehomogenosti. Slika 1a prikazuje rezini mikro CT[3] slike zoba pred in po nelinearnem filtriranju za 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:

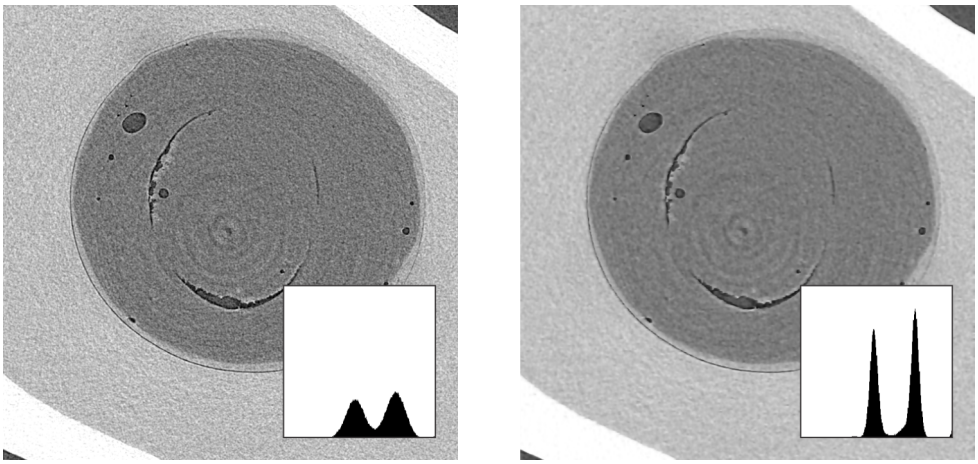
$$E(f(x,y)) = E(g(x,y) * m(x,y)) + E(a(x,y)). \tag{2}$$

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.

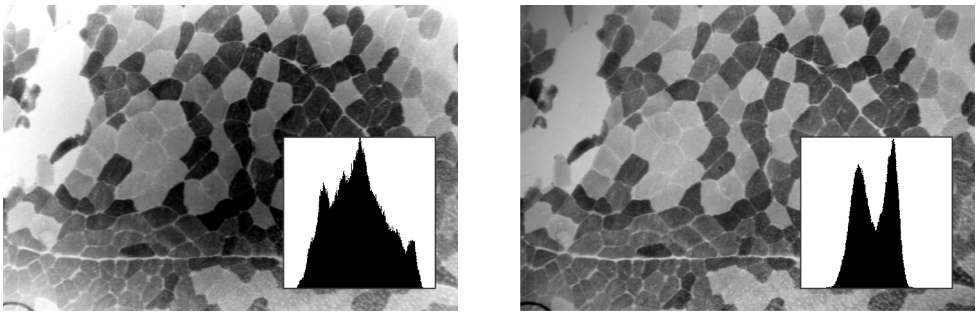
8. Uporabite funkciji za glajenje slike s povprečenjem in z mediano v knjižnici SimpleITK:

[3]CT: ang. *Computed Tomography*

(a)



(b)



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.